

A Taxonomy of Software Bots:
Towards a Deeper Understanding of Software Bot Characteristics

by

Carlene R. Lebeuf
B.Sc., University of Victoria, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Carlene Lebeuf, 2018
University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by
photocopying or other means, without the permission of the author.

A Taxonomy of Software Bots:
Towards a Deeper Understanding of Software Bot Characteristics

by

Carlene R. Lebeuf
B.Sc., University of Victoria, 2013

Supervisory Committee

Dr. Margaret-Anne Storey, Supervisor
(Department of Computer Science)

Dr. Neil Ernst, Departmental Member
(Department of Computer Science)

Dr. Hausi A. Müller, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Margaret-Anne Storey, Supervisor
(Department of Computer Science)

Dr. Neil Ernst, Departmental Member
(Department of Computer Science)

Dr. Hausi A. Müller, Departmental Member
(Department of Computer Science)

ABSTRACT

Software bots are becoming increasingly pervasive in our everyday lives. While bots have been around for many decades, recent technological advancements and the adoption of language-based platforms have led to a surge of new ubiquitous software bots. Although many new bots are being built, the terminology used to describe them and their properties are vast, diverse, and often inconsistent. This hinders our ability to study, understand, and classify bots, and restricts our ability to help practitioners design and evaluate their bots.

The overarching goal of this thesis is to provide a deeper understanding of the complexities of modern software bots. To achieve this, I reflect on a multitude of existing software bot definitions and classifications. Moreover, I propose an updated definition for bots and compare them to other bot-like technologies. As my main contribution, I formally define a set of consistent terminology for describing and classifying software bots, through the development of a faceted taxonomy of software bots. The taxonomy focuses on the observable properties and behaviours of software bots, abstracting details pertaining to their structure and implementation, to help safeguard against technological change. To bridge the gap between existing research and the proposed taxonomy, I map the terminology used in previous literature to the terminology used in the software bot taxonomy. Lastly, to make my contributions actionable, I provide guidelines to illustrate how the proposed taxonomy can be leveraged by researchers, practitioners, and users.

Contents

Supervisory Committee	ii
Abstract	iii
Table of Contents	iv
List of Tables	ix
List of Figures	x
Acknowledgements	xii
Dedication	xiii
1 Introduction	1
1.1 Research Questions	2
1.2 Contributions	3
1.3 Structure	3
2 Software Bot Background	5
2.1 The Evolution of Software Bots	5
2.1.1 Daemons	6
2.1.2 Automata	6
2.1.3 Robots	7
2.1.4 Chatbots	7
2.1.5 Expert Systems	8
2.1.6 Software Agents	9
2.2 The AI Winter	9
2.3 Why Now? The Re-Emergence of Software Bots	11
2.3.1 Technological Advancements	11
2.3.2 Mainstream Adoption of Messaging Platforms	11
2.3.3 Emergence of Voice-Only Platforms	12

2.3.4	Transition to Service Oriented Architectures	12
2.3.5	Abundance of Public APIs and Datasets	13
2.3.6	Industry Support	13
2.4	Summary	14
3	Defining Software Bots	16
3.1	What Is a Software Bot?	16
3.1.1	Proposed Definition of Software Bots	18
3.2	Comparing Software Bots and Related Technologies	21
3.2.1	Software Bots vs. Robots	21
3.2.2	Software Bots vs. Scripts	21
3.2.3	Software Bots vs. Programs	22
3.2.4	Software Bots vs. Agents	23
3.2.5	Software Bots vs. Chatbots	24
3.3	Existing Classifications of Software Bots	25
3.3.1	High-Level Classifications (Subtypes)	25
3.3.2	Role-Based Classifications	26
3.3.3	Other Classifications	27
3.4	Shortcomings of Existing Classifications	28
3.5	Summary	28
4	Developing a Software Bot Taxonomy	30
4.1	Taxonomy Generation Methodology	30
4.2	Planning (Phase 1)	33
4.2.1	Knowledge Area	33
4.2.2	Objectives & Scope	34
4.2.3	Subject Matter	34
4.2.4	Classification Structure	35
4.2.5	Classification Procedure	35
4.3	Data Collection (Phase 2)	36
4.3.1	Systematic Literature Search	36
4.3.2	Backwards Snowballing	40
4.3.3	Online Search	41
4.4	Identification & Extraction (Phase 3)	41
4.4.1	Term Identification & Extraction	41
4.4.2	Term Reduction	42
4.5	Design & Construction (Phase 4)	43
4.5.1	Card Sorting	44
4.5.2	Dimensions, Facets, and Relationships	45

4.5.3	Drafting Definitions & Refining Dimensions	46
4.6	Summary	48
5	A Taxonomy of Software Bot	49
5.1	Reader's Guide	50
5.1.1	General Usage Guidelines	52
5.2	Environment Dimensions	53
5.2.1	Environment Type	53
5.2.2	Scope	54
5.2.3	Closure	55
5.2.4	Dynamism	55
5.2.5	Predictability	55
5.2.6	Permanence	56
5.2.7	Population	57
5.3	Intrinsic Dimensions	58
5.3.1	Knowledge	58
5.3.2	Reasoning	60
5.3.3	Adaptability	63
5.3.4	Goals	65
5.3.5	Delegation	67
5.3.6	Specialization	68
5.3.7	Anthropomorphism	68
5.3.8	Life Cycle	72
5.4	Interaction Dimensions	74
5.4.1	Access	74
5.4.2	Sense	74
5.4.3	Act	74
5.4.4	Communicate	76
5.4.5	Initiative	79
5.4.6	Robustness	80
5.4.7	Mobility	81
5.5	Summary	82
6	Taxonomy Validation	83
6.1	Benchmarking	83
6.2	Subject Matter Tagging	84
6.3	Domain Expert Tagging	85
6.4	Summary	86

7	Discussion, Limitations, and Future Work	88
7.1	Why Another Software Bot Taxonomy?	88
7.2	Limitations	90
7.3	Operationalizing the Taxonomy	93
7.3.1	Researchers	93
7.3.2	Practitioners	94
7.3.3	End Users	95
8	Conclusions	96
8.1	Summary of Research	96
8.2	Final Remarks	97
	Appendices	98
A	Collection Queries	99
A.1	ACM Digital Library	99
A.2	IEEE Xplore	99
A.3	ScienceDirect (Computer Science section)	99
A.4	SpringerLink	100
A.5	Wiley Online (Computer Science section)	100
B	Data Extracted from Article Selection	101
B.1	Excluded Systematic Search Articles	101
B.2	Included Systematic Search Articles	105
B.3	Included Snowballed Articles	107
B.4	Included Online Articles	108
C	Excluded Cards	110
D	Restructuring the Taxonomy	111
E	Mapped Terminology	113
E.1	Environment Dimension	113
E.2	Intrinsic Dimension	114
E.3	Interaction Dimension	117
F	Benchmarking Validation	119
F.1	Literature Search Articles:	119
F.2	Snowballing Articles:	122
F.3	Online Articles:	123

G Validation: Subject Matter Tagging	126
H Study Questions	129
H.1 Participant Background & Experience	129
H.2 Software Bot Taxonomy Feedback	130
I H.R.E.B. Ethics Approval	131
J Expert Tagging Session	132
Bibliography	135

List of Tables

Table 4.1	Breakdown of search results by database.	38
Table C.1	Cards excluded during taxonomy generation	110
Table E.1	Terminology mappings for the <i>environment</i> dimension	113
Table E.2	Terminology mappings for the <i>intrinsic</i> dimension	114
Table E.3	Terminology mappings for the <i>interaction</i> dimension	117
Table J.1	Software bot tagging on the <i>environment</i> dimension and facets.	126
Table J.2	Software bot tagging on the <i>intrinsic</i> dimension and facets.	126
Table J.3	Software bot tagging on the <i>interaction</i> dimension and facets.	126
Table J.1	Expert tagging on the <i>environment</i> dimension and facets.	132
Table J.2	Expert tagging on the <i>intrinsic</i> dimension and facets.	132
Table J.3	Expert tagging on the <i>interaction</i> dimension and facets.	132

List of Figures

Figure 3.1	The relationship between software bot interfaces and software services.	19
Figure 3.2	The relationship between software bot interfaces and software services: (a) software bot with external services, (b) software bot with internal services, and (c) software bot with both internal and external services.	19
Figure 3.3	A time-line of the emergence and mainstream adoption of new user interface paradigms, adapted from Ryan Block ¹ .	20
Figure 3.4	The relationship between (a) software bots, (b) software services, (c) software bots with internal services, and (d) software programs/scripts.	23
Figure 3.5	The relationship between (a) software bots, (b) software services, (c) software bots with internal services, (d) software programs/scripts, (e) software agents, and (f) chatbots.	25
Figure 3.6	Socio-Technical Model for Collaborative Software Development in- cluding the (a) society's social system, (b) team's social system, and three categories of friction that bots can help reduce: (b) team's inter- actions, (c) individual's interactions with technology, and (d) team's interactions with technology.	27
Figure 4.1	Usman et al.'s methodology for taxonomy generation [1]	31
Figure 4.2	Methodology for creating the updated taxonomy, adapted from Usman et al. [1]. The underlined/strikeout text depicts steps that were added or removed, respectively.	32
Figure 4.3	Comparison of the structure of (a) hierarchical taxonomies and (b) faceted taxonomies. The green boxes reflect the how a sample entity would be classified in each taxonomy.	36
Figure 4.4	An overview of the methodology followed for the systematic literature search data collection process.	37
Figure 4.5	The taxonomy construction methodology followed.	43
Figure 4.6	The content for the card sorting process.	44

Figure 4.7	The software bot taxonomy at various stages of creation: (a) shuffled cards ready to be sorted; (b) beginning to group similar terms; (c) groups have been created; (d) assigning the groups dimension labels; (e) labeling the dimensions, facets, and sub-facets; (f) the completed initial version of the taxonomy.	46
Figure 4.8	A high level overview of the preliminary version of the Software Bot Taxonomy using FreeMind ⁹⁵	47
Figure 5.1	A high-level view of the Software Bot Taxonomy's structure.	50
Figure 5.2	Example of the structure of the proposed taxonomy.	50
Figure 5.3	Examples of the three possible facet value types: (a) boolean sub-facets, (b) exclusive states, and (c) ranges.	51
Figure 5.4	The Software Bot Taxonomy's Environment Dimensions. The dimensions/facets/sub-facets and the range of possible values which the facet can take on are shown in the solid and dashed boxes, respectively.	54
Figure 5.5	The software bot taxonomy's intrinsic dimensions. The dimensions/facets/sub-facets as well as the range of possible values which the facet can take on are shown in the solid and dashed boxes, respectively.	59
Figure 5.6	The software bot taxonomy's interaction dimensions. The dimensions/facets/sub-facets, as well as the range of possible values which the facet can take on, are shown in the solid and dashed boxes, respectively.	75
Figure D.1	Version #1 of the re-ordering of dimensions, facets, and sub-facets.	111
Figure D.2	Version #2 of the re-ordering of dimensions, facets, and sub-facets.	111
Figure D.3	Version #3 of the re-ordering of dimensions, facets, and sub-facets.	112
Figure D.4	Version #4 of the re-ordering of dimensions, facets, and sub-facets.	112

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor **Dr. Margaret-Anne Storey** for her mentorship, encouragement, enthusiasm, and patience. The insightful feedback and guidance you provided were invaluable, and not only helped improve my research but also shape the way I approach problems to this day. Thank you for helping me grow both personally and professionally.

I would also like to thank my **thesis committee** for their insights and helping push me to make my research better.

All of the **members of the CHISEL lab** (both past and present) for their support, feedback, and laughs along the way. It wouldn't have been the same without each and every one of you! A special thanks to **Courtney Bornholdt**, **Elena Voyloshnikova**, **Eirini Kalliamvakou**, **Maryi Arciniegas-Mendez** for their friendship, support, and chats whenever I needed it the most; **Alexey Zagalsky** for his patience, thoughtful comments on whatever I was working on, and suffering through many long rants about bots (and many other things); **Matthieu Foucault** for his time and support in helping me generating and refine the taxonomy presented in this thesis; and **Cassandra Petrachenko** for her sharp wit, seemingly endless help and knowledge whenever I was in a pinch, and thoughtful editing of not only this thesis but also many other projects.

Lastly, I would like to thank my **parents, sister, friends, and loved ones** for believing in me, sharing in my high points, and being there for me in the low ones. I couldn't have done it without you!

DEDICATION

For my family.

Chapter 1

Introduction

From the earliest days of computer programs, people have dreamed about creating programs that could think and behave like humans. Such programs could not only automate tasks that humans perform, but they could also work with humans to solve intellectual tasks that cannot be entirely automated. The term “*bot*” was used to describe a realization of this vision quite early on. In just the past few decades, we have witnessed a rebirth of the next generation of software bot technologies.

Although bots have been around for years, the ease with which software bots can be integrated into modern communication tools has created an explosion of new bots. There has been a widespread adoption of conversational applications, both for personal use (e.g., Facebook Messenger,¹ WhatsApp,² Telegram,³ or Skype⁴) and within the software development domain (e.g., Slack,⁵ Teams,⁶ or Stride⁷). Messaging applications are being used not only to chat with friends, but also to connect users with companies, browse products, and consume a variety of media. These once simple, text-based applications that allowed users to send messages and share pictures, videos, and GIFs with friends have evolved into complex ecosystems that allow developers to build custom integrations using the platforms’ APIs. These platforms now serve as a breeding ground for new, conversation-based tools and integrations, often in the form of software bots. Although software bot technologies are still relatively new to these platforms—most emerging within the past few years—the use of bots continues to grow at a rapid pace. Nowadays, there exists a huge range of bots, from small and simple apps, to huge and complex entities. They can help with almost anything you need to get done: from reading a bedtime story,⁸ to booking a flight,⁹ to calling a restaurant and making a reservation.¹⁰ For almost anything you can do with an app, there is also now *a bot for that*.¹¹

The use of the buzzword “*bot*” is continuing to grow in popularity, both in research and industry. Despite its popularity, however, there is little consensus on what exactly constitutes a bot. To date, there is a lack of a generally accepted definition of software

bots. The word “*bot*” is vague, but it is commonly used to describe a wide range of software programs. Research into software bots is also quite diverse and spans many disciplines, making it even more difficult to pinpoint one single definition for software bots. This has led researchers and practitioners to define software bots in accordance with their unique applications, if they define them at all.

The knowledge field of software bots, like many other areas in software engineering research, is still relatively immature. Providing some form of consistent organization and description of software bots can help *researchers* disseminate knowledge more easily, better understand the relationships between the different properties/behaviours of software bots, and aid in identifying gaps in the knowledge area [1]. This would also help *practitioners* and *end-users* better understand the software bot technologies they are building and using.

1.1 Research Questions

The overarching goal of this research is to provide a deeper understanding of modern software bots. Although software bots are widely used, there remains a lot about them that is not well understood. More precisely, the research presented in this thesis is motivated by three main research questions.

RQ 1: What is a software bot? With the recent explosion of new software bots, the boundaries of what constitutes a bot continue to be blurred. As Socrates stated, “[t]he beginning of wisdom is the definition of terms”. Thus, without first defining software bots, you cannot meaningfully discuss them.

RQ 2: What properties of software bots can be used to classify them? Like software systems, software bots are complex entities composed of a “*purposeful collection of interrelated components that work together to achieve some objective*” [2]. Software bots are more than simply a sum of their parts. To better understand software bots, they can be classified through a combination of the properties or behaviours that emerge from the system as a whole, rather than a single high-level category.

RQ 3: What existing terminology, used in previous software bot research, can be mapped onto the new classification scheme? Since software bots are still an emerging field of research, the terminology used to describe them is vast, diverse, and often inconsistent. Research into them spans across and borrows from many disciplines, both in industry and academia. This diversity, however, is an obstacle that may lead to further division of software bot research, as well as confusion for those building or using bots. Providing a set of consistent terminology that previous research can be mapped to would

be a solid step towards untangling the various terminology used to describe similar concepts across the many domains of software bot research.

1.2 Contributions

Stemming from the research questions identified above, the work presented in this thesis provides three main contributions to software bot research.

Defining a Software Bot: I reflect on previous descriptions of software bots and propose an updated definition. I also examine the relationship between software bots and other related technologies. Lastly, I reflect on existing classifications of software bots, highlighting the shortcomings of the current ways we are trying to describe them.

Taxonomy of Software Bots: Researchers have long acknowledged the value of taxonomies and formally describing knowledge areas [1]. Inspired by existing classifications of various software bot subtypes, as well as software bot technologies themselves, I produced an updated classification of software bots in the form of a multi-faceted taxonomy. This classification focuses on the observable properties of software bots. The proposed taxonomy was generated through the merging of numerous existing classifications of software bots. This taxonomy helps to connect software bot research across multiple domains by providing a set controlled vocabulary for discussing software bots in the future.

Software Bot Terminology Mappings: Lastly, I provide an updated mapping between the terminology used in previous software bot research and the terminology used in the proposed taxonomy. This mapping helps connect the different domains of software bot research by mapping the many terms used to describe the equivalent concepts to the controlled vocabulary.

1.3 Structure

The remainder of this thesis is structured as follows:

Chapter 2: I provide a brief history of the ancestors of the modern software bot. I also discuss the factors that may have led to the recent re-emergence and explosion of software bot technologies.

Chapter 3: I explore some of the existing ways that software bots have been described, provide an updated definition of software bots, and compare software bots to related technologies. Lastly, I describe some existing classifications of software bots.

Chapter 4: I present the methodology used to generate the new faceted taxonomy of the observable properties and behaviours of software bots. I also discuss how the taxonomy was validated and any implications of my approach.

Chapter 5: I introduce the proposed taxonomy of software bots and describe each of the properties and behaviours that it includes. I also provide a guide to help readers interpret and use the proposed software bot taxonomy.

Chapter 6: I describe the methodology used to validate the proposed taxonomy and discuss my findings.

Chapter 7: I discuss the importance of the work presented in this thesis, reflect on its limitations, and provide suggestions on how it can be used in the future.

Chapter 8: I conclude this thesis by reflecting on the three research questions that sparked my exploration into software bots.

Additional documents can be found in the Appendices, including lists of the articles used to generate the taxonomy, initial drafts of the taxonomy's structure, materials used in the validation of the taxonomy, and the terminology mappings.

¹<https://www.messenger.com/>

²<https://www.whatsapp.com/>

³<https://telegram.org/>

⁴<https://www.skype.com/en/>

⁵<https://www.slack.com/>

⁶<https://products.office.com/en-ca/microsoft-teams/>

⁷<https://www.stride.com/>

⁸<https://www.amazon.ca/Webguild-Short-Bedtime-Story/dp/B01DJCJTZ2>

⁹<https://www.hipmunk.com/tailwind/hello-hipmunk-bot-for-skype/>

¹⁰<https://www.androidcentral.com/google-duplex>

¹¹<https://www.thereisabotforthat.com/>

Chapter 2

Software Bot Background

This chapter provides an overview of the history of software bots and helps to ground the re-emergence of software bots within the larger landscape of bot-like technologies. The chapter is divided into three sections: a brief history of the technologies from which, I believe, software bots descended; the artificial intelligence winter, which threatened research into software bot-like technologies; and the factors contributing to the recent re-emergence of software bots.

2.1 The Evolution of Software Bots

Humans have always been interested in intelligence: understanding it, measuring it, even attempting to artificially create it. The field of artificial intelligence, commonly referred to as AI, tries to do just that: build intelligent entities. Software bots have proven to be great test-beds for researchers to experiment with many of the advancements stemming from AI research [3]. Although the field of AI has received a lot of attention, many other software bot-like technologies are rarely mentioned. In this section, I briefly reflect on AI and other influential technologies that set the stage for modern software bots.

The field of artificial intelligence is still relatively new, the term itself being coined by John McCarthy in 1956 [4]. According to Russell and Norvig [5], AI research is concerned with creating entities that *think* and *act* humanly and rationally. Efforts towards creating entities that *think* humanly and rationally focus on developing algorithmic solutions to problems like knowledge reasoning, planning, machine learning, input processing, etc. These problems have spurred research into techniques such as search and optimization, logic, probabilistic methods, classifiers and statistical learning methods, and neural networks.

Efforts into creating entities that *act* humanly and rationally have produced many of the technologies that I believe to be the *ancestors* of the modern software bot. In the following sections, I present a brief history of many of these software bot-like technologies

in a roughly chronological order. Although there is some overlap between many of these research areas and technologies, I call out each field individually as I believe they have all contributed (in their own way) to the modern software bot ecosystem.

2.1.1 Daemons

The first known appearance of a bot-like helper was Socrates’ *daimonion* in 399 BC [3]. This invisible, intelligent helper offered Socrates advice and served as a voice of reason by warning him of possible mistakes. Although Socrates’ *daimonion* provided him with guidance, it never explicitly told him what to do and it allowed him to make his own decisions. Socrates’ *daimonion* closely resembled the daemons found in Greek mythology: supernatural beings (either good or evil) working in the background.

The concept of helpful daemons re-entered the scene in 1871 when physicist James Clerk Maxwell imagined that a *demon* could (in theory) be used to sort molecules in order to violate the second law of thermodynamics [6]. Maxwell claimed that the demon could monitor a small door between two gas chambers and open/close it to allow only the fast molecules to pass into the second chamber. This would divide the fast and slow molecules and allow for the second chamber to warm up as the first chamber cooled down. Since his original proposal, Maxwell’s work has been highly criticized since a “*purely mechanical Maxwell’s demon is impossible*” [7]. However, the idea of having a daemon that could silently automate tasks in the background resonated.

The first real *digital daemons* were created for the Multics operating system by programmers at Massachusetts Institute of Technology’s Project MAC in 1963 [3]. They adopted the term of daemon, which is still used today, to describe small programs running unobtrusively as background processes instead of being directly controlled by users on Unix-like operating systems.

2.1.2 Automata

One of the first efforts towards building *physical* life-like entities, automata are self-operating machines that follow a predetermined sequence of operations. Although automata have taken a variety of forms, many were designed to look like they operate on their own accord, often taking a human-like form. The idea of automata stems as far back as ancient Greek mythology [8]: in his workshop, Hephaestus created Talos, an artificial man of bronze [8].

In the Hellenistic period, many automata were created as art, toys, tools, and even scientific prototypes [8]. The manufacturing of small automata continued well into the 15th century, when some of the first mechanical clocks and measuring instruments began to appear in European towns [9]. Many other forms of automata quickly gained in popularity. Although not re-discovered until the 1950s, Leonardo da Vinci’s 1495 mechanical man (if

successfully built) would have been able to move its arms, twist its head, and sit upright. Other automaton, such as drummers and flute players, would perform in public concerts [9]. In 1738, Jacques de Vaucanson presented the digesting duck, an automaton duck that could eat and dispose of its waste [9, 10]. Although it was eventually revealed as a hoax, the Mechanical Turk toured around Europe in the 18th century posing as an intelligent, chess playing automaton [10]. A more recent application of automaton was NASA’s Automaton Rover for Extreme Environments.¹² Since the harsh climates on Venus were unfavourable to modern robotics, NASA developed a wind-powered automaton to explore the planet.

2.1.3 Robots

The first appearance of the term robot is credited to a 1921 science fiction play entitled *Rossums Universal Robots* [11] (RUR), where the author replaced the term automata with robot. Although not exactly robots by our current standards, these RUR robots were living creatures rather than machinery. The term robot was adapted from the Slavic word *robota*, meaning “*forced labourer*”, and RUR’s robots were just that: synthetic, organic, human-like beings that could work around the clock.[11]

Although the term *bot* originated as an abbreviation of robot, unlike software bots which are digital, robots are mechanical. Robots are used in the *physical world* much in the same way software bots are used in the *digital world*; They have tangible, mechanical bodies that perform tasks by manipulating the physical world, often helping automate repetitive tasks.

The world’s first intelligent, human-like robot was created by Japan’s Waseda University in 1972: WABOT-1 could walk around, extend its arm, and grip objects [12]. It used its sensors (artificial eyes and ears) to gather data about its environment, and it used its artificial mouth to speak to users. Also in the early 1970s, researchers at the University of Edinburgh in Scotland developed Freddy,[13] a non-verbal robot capable of assembling simple objects without intervention. Today, robots can be found automating a variety of real-world tasks, from assembly lines, to automatically vacuuming floors and driving cars, to industry 4.0 robots [14].

2.1.4 Chatbots

In 1950, computer scientist and mathematician Alan Turing developed the Turing Test, also known as the Imitation Game [15]. The game required three players: two humans and a machine. One of the humans serves as the interrogator, and the identities of the other human and the machine are concealed from the interrogator. By asking only text-based questions, the interrogator must determine which of them is human. If the machine successfully tricked the interrogator into believing it is human, then it passed the test.

The Turing Test sparked the development of chatbots, computer programs designed to *act humanly* by talking to users [5]. Created in 1966 by MIT professor Joseph Weizenbaum, Eliza was the first computer program that could have a conversation with humans. Eliza attempted to cover her limited vocabulary by simulating a psychotherapist. Eliza searched for keywords in the user’s speech and responded with preprogrammed questions, shifting the focus of the conversation back onto the user.

Eliza inspired a variety of notable chatbots, some of which include: Perry (1972), the paranoid schizophrenic [16]; Alice (1995), the natural language processing bot with lots of personality [17]; and SmarterChild (2000) [18]. While these earlier chatbots’ interactions were purely text based, advances to natural language processing allowed chatbots to begin using spoken language or a combination of text and speech. The personal assistant chatbot Julia (1994) was the first verbal chatbot [19]. A couple of years later, Sylvie (1997) became the first “virtual human” with its own animated face and voice [20]. Among the thousands of chatbots that exist today, some popular mainstream examples include: Apple’s Siri,¹³ Microsoft’s Cortana,³⁷ Amazon’s Alexa,¹⁴ Google Assistant,¹⁵ Microsoft’s Tay,¹⁶ Cleverbot,¹⁷ and IBM’s Watson.¹⁸

2.1.5 Expert Systems

Expert systems are often considered one of the first successful applications of AI [21]. The first expert systems emerged in the 1970s and became increasingly popular during the 1980s. An expert system is a computer system designed to emulate the decision-making capabilities of human experts in a complex but narrow domain. Expert systems rely on domain experts to provide a predefined knowledge base, which the systems analyse with their inference engines to arrive at a solution to a domain-specific problem. These systems leverage certain AI techniques, such as rules and cognitive models, to solve complex problems at the same level as, or sometimes even better than, human experts.

Ted Shortliffe’s 1974 PhD thesis proposed the first use of an expert system [9]. Shortliffe’s system, MYCIN, was used to help identify disease in patients by asking them diagnostic questions and comparing answers to a knowledge base compiled by experts. Throughout the 80s and 90s, expert systems were successfully applied in agriculture, education, law, manufacturing, etc. However, due to their symbolic nature, many expert systems failed to scale. As a result, expert systems quickly developed a reputation of being too brittle and too ill-suited for more complex problem-solving tasks. Expert systems were also only as good as their knowledge bases and the fact that data storage was expensive in the 1980s. The systems’ knowledge bases also required continual maintenance and updating. As a result, many expert systems were abandoned since they were too costly to maintain.

Today, expert systems can still be found in many financial institutions, manufacturing companies, and within the medical domain. A modern example of an expert medical system

is DXplain [22], a system that uses clinical findings (i.e., signs, symptoms, laboratory data) to produce ranked lists of possible diagnoses. However, since expert systems present a strategic advantage to the companies that own and use them, they are not often discussed outside of the companies that deploy them.

2.1.6 Software Agents

The word agent originates from the Latin word *agere*, meaning “to do” or “to act on someone’s behalf” [5, 23]. The first software agents can be traced back to Hewitt’s Actor Model [23]. In his model, Hewitt describes a software *actor* that is in control of its own state and can respond to messages from similar systems:

“A self-contained, interactive and concurrently-executing object, possessing internal state and communication capability.” —Hewitt, 1977 [24]

Much of the early work on software agents, from the late 1970s to early 1990s, evolved from multi-agent systems. Nwana [23] highlighted that much of the early work from this period focused on the “*macro aspects*” of agent systems: how to design systems composed of multiple collaborative agents [25, 26, 27], and classical theories of agent architectures and languages [28, 29]. Research into multi-agent systems (and in turn software agents themselves) inherited many properties from distributed AI, distributed problem solving, and parallel AI. Software agents also inherited the benefits of modularity, speed (i.e., influenced by parallelism), and reliability (i.e., redundancy) [23].

Software agent research began to diversify in the 1990s and a variety of agents emerged to support a broad range of tasks across many domains. Bringing agents into the public eye, the famous “*Knowledge Navigator*” video portrayed the interaction between a software agent and its user [30]. Software agents also began to take on various names, based on either some significant property (e.g., collaborative, interface, mobile, internet, reactive, or smart) or their purpose (e.g., personal assistants [31], guides [32], buying/selling [33], or entertainment [34]).

2.2 The AI Winter

The early days of the AI boom, however, were fueled with unrealistic expectations. Many AI researchers dramatically overestimated the future capabilities of AI systems, and inevitably they under delivered. Although AI overhype is typically cited as the main cause of the AI winter, it most likely stemmed from a “*perfect storm*” of factors [21].

For one, in the early days of AI, researchers faced issues with computing capacity. As a result, many of the potential technological advancements (e.g., neural networks) were never fully realized, as they required computing power far beyond the hardware capabilities

of the time. In fact, many of the recent advances in AI stem from decades-old research that was never fully realized.¹⁹ Also, AI research is often multidisciplinary and faces many of the problems that interdisciplinary fields face: unclear funding; being secondary to the primary research; and when one discipline faces budget cuts, the entire project can suffer. Similarly, when institutions were required to make cuts, risky non-essential ventures like AI were usually the first to go.

One of the first major cuts to AI funding came in 1974 when Sir James Lighthill produced a report that convinced the British government to dismantle most of their AI research objectives. The report, which came to be known as the Lighthill report, concluded that nothing could be discovered through AI research that could not come from other scientific disciplines. Lighthill claimed that AI research utterly failed to achieve its *“grandiose objectives”* and that *“in no part of the field have discoveries made so far produced the major impact that was then promised”* [35]. Since then, the Lighthill report has been strongly criticized, and many now view Lighthill’s pessimism as unfounded [21]. *“The prior opinion of many informed observers, based on decades of disappointing experimental results, was that the problems were so hard that they would remain unsolved for many decades yet”*, Hans Moravec, a robotics researcher in the 1970s, *“[b]ut now everyone knows differently”*.

²⁰ But at the time, the Lighthill report had a devastating effect on AI research, leading to major cuts in AI research at nearly all British universities. Many of Lighthill’s claims were also echoed in a follow-up Defense Advanced Research Projects Agency (DARPA) report, leading to many AI-related cuts in the United States in the late 1970s [36]. Luckily, some of the programs that had begun prior to these reports were allowed to continue. In the late 1970s came the first wave of commercialized AI technologies in the form of *expert systems*. By the mid-1980s, expert systems had begun to see promising results, and were deployed in a variety of industry settings. However, many believed that popularity of the new expert systems meant the end for traditional AI: many AI researchers disowned expert systems, claiming that their rule-based logic was not true AI [21]. Though much of the popularity that expert systems had been gaining began to fade as they failed to scale to larger problems, the field of AI began to slip back into a second winter.

After a few decades of winter, the *“AI spring”* bloomed in the the early 2000s.²⁰ The next section discusses some of the factors commonly contributed with ending the AI winter. There were, however, some lasting effects.

John Markoff wrote that, *“[a]t its low point, some computer scientists and software engineers avoided the term artificial intelligence for fear of being viewed as wild-eyed dreamers”*.

²⁰ Many avenues of research that were traditionally viewed as part of the AI space broke off into a variety of sub-fields (e.g., computer vision, robotics, natural language processing), and researchers continued to make incremental improvements on the technologies that already existed [9].

2.3 Why Now? The Re-Emergence of Software Bots

What sparked the re-emergence and mainstream adoption of software bot technologies? In the following sections, I discuss many of the factors that may have led to the recent explosion of new software bot technologies.

2.3.1 Technological Advancements

In just the past few decades alone, there have been numerous technological breakthroughs. When Eliza was first introduced in 1966, interactive computing was a new thing. Since then, the invention of personal computers, the internet, smartphones, and internet of things enabled devices (just to name a few) have completely changed the way that people interact with technology. People have also been using these technological breakthroughs in new and exciting ways, e.g., using the internet to perform automation through strategies such as web-tasking [37].

We have also seen dramatic increases in the computing power available. Personal computers are growing more powerful by the day, and services such as Amazon Web Services,²¹ Microsoft Azure,²² Google Cloud,²³ and IBM Cloud²⁴ offer affordable access to powerful computing capabilities. This abundance of affordable computing power has led to numerous advances in many fields, including machine learning, natural language processing, speech recognition, and computer vision. These advancements helped lay the groundwork for many of the technologies that modern software bots rely on to be successful.

2.3.2 Mainstream Adoption of Messaging Platforms

Another factor that contributed to the sudden popularity of bots was the mainstream adoption of messaging platforms. When many of the early bot-like technologies were first introduced, they were built into platforms that were not easily accessible by the public. Today, people are spending more time on messaging platforms than any other type of application. They are entirely comfortable communicating via short typed interactions and carrying out several asynchronous conversations at the same time [38]. Software bots provide users with quick ways to get things done inside the platforms they are already using. Furthermore, Nir Eyal pointed out that “[t]he power of the conversational interface is that it shields the end user from having to learn anything new. We already know how to chat, so making requests is easy”.²⁵ Interacting with bots through messaging platforms also removes the need to download, install, and switch between multiple applications to get things done – helping to reduce “app fatigue” [39].

Messaging platforms are pervasive in both our personal lives and at work. Major social messaging platforms such as Apple’s Messenger,²⁶ Microsoft’s Skype,²⁷ WhatsApp,²⁸

Telegram,²⁹ WeChat,³⁰ and Kik³¹ all not only support software bots but encourage them. Workplace communication platforms like Slack,³² Microsoft Teams,³³ Stride,³⁴ and Flock³⁵ also have thriving communities of helpful ready-to-integrate bots and offer software development kits for building bots.

2.3.3 Emergence of Voice-Only Platforms

One of the members of the new wave of software bots that has been receiving a lot of attention lately is voice-driven personal assistants. One of the first mainstream voice-driven technologies to be made publicly available was Siri,¹³ a virtual assistant that lives inside Apple’s operating systems. Originally a standalone iOS application, Siri was quickly acquired by Apple, integrated into the iPhone, and then eventually added to almost all of Apple’s products. Since then, many companies have released their own voice assistants, a few prominent examples being Google Now³⁶ (2012), Microsoft Cortana³⁷ (2015), and Google Assistant (2016).¹⁵

Voice technologies are also making their way into our homes. Since the introduction of Amazon’s Alexa³⁸ in 2014, voice-only technologies for the home have only been increasing in popularity. Alexa serves as a central voice-operated hub for smart devices and home automation. Since then, Google Home³⁹ (2016) and Apple Homepod⁴⁰ (2018) have also entered the voice-driven, home automation scene. Today, an estimated 16% of Americans (i.e., approximately 39 million people) own or use smart speakers [40]. Typically, these devices help users by answering simple questions or performing small tasks on their behalf. Some of the most popular uses of smart speakers include checking the weather or traffic reports, adding things to shopping lists, finding restaurants or businesses, looking for recipes, ordering food, and even reading bedtime stories[40].

2.3.4 Transition to Service Oriented Architectures

Another factor possibly contributing to the increasing number of bots was an overall shift in the way that developers were thinking about and building software. Service-oriented architecture (SOA), a style of software design which promotes loose coupling between services, began gaining popularity in the early 2000s. SOA organizes software so that logic is divided into discrete units of functionality (i.e., services) that communicate with each other via set protocols. By isolating core functionality into services, it becomes easier to reason about what each component does. Many of the basic principles of SOA are echoed in the designs of modern software bots.

2.3.5 Abundance of Public APIs and Datasets

Application programming interfaces (API) embrace essentially the same goals as SOA systems but are more open. Internal APIs have been a key part of software development for many years, providing a way to develop for specific platforms (e.g., Windows operating system APIs). Salesforce is often credited as being the first public web API, releasing an “*internet as a service*”-style API in the early 2000s.⁴¹ Over the next few years, many websites, such as Ebay (2000), Del.icio.us (2003), Flickr (2004), Facebook (2006), Twitter (2006), and Google Maps (2006), began releasing public APIs to allow developers to access their data and/or services.⁴¹

Today, almost every online platform offers some form of web API, whether to build integrations for their site or to grant access to the data they possess. Currently, the ProgrammableWeb⁴² directory lists over 19,000 public APIs. These APIs are easily consumable, relatively amendable, and often support human-readable formats (e.g., JSON). API architectural styles, such as REST [41], also provide for standards for creating web services. Since APIs can also be marketed as products themselves, third-party API companies are fundamentally changing the way we build and sell software. Developers no longer need to re-invent the wheel, but instead rely on APIs to provide a set of basic building blocks for creating software, giving them more time to focus on their core features.

In the early 2000s, there was a shift of focus in AI research, from developing/optimizing algorithms to improving the scale/quality of the data being used [5]. In recent years, there has been a push to release large, well-curated datasets from governing bodies, research institutions, and industry. In 2011, the Canadian federal government launched *Canada’s National Open Data* website.⁴³ Since then, Alberta,⁴⁴ British Columbia,⁴⁵ Ontario,⁴⁶ and Quebec⁴⁷ have all launched their own provincial open data portals. In 2016, Yahoo released a 13.5TB machine learning dataset filled with interactions from 20 million users for academic research purposes through the Webscope⁴⁸ platform. Since then, many large companies as well as academic institutions have released a range of new public datasets for developers to leverage. Today, Kaggle,⁴⁹ a popular platform that hosts data analytic competitions, offers over 9,000 public datasets. These public datasets (and APIs) provide much of the content, services, and building blocks used by many of today’s popular software bots.

2.3.6 Industry Support

The re-emergence of software bot technologies has been very much fueled by industry. For years, bots have been seen as a way for developers to “*scratch their own itch*” and improve their development workflows.⁵⁰ Many software bots started as small personified scripts used to help automate part of software developers’ processes. The content was there, developers just needed a better way to access it... bots!

Recently, major software companies have begun investing heavily in software bot-like technologies. In 2011, GitHub released a chatbot building framework, Hubot, which has since been translated into multiple programming languages and forked over 3,000 times.⁵¹ GitHub now also offers support documentation for bot developers and even offers a unique user type to support bot-style integrations.⁵² Shortly after Alexa’s release, Amazon announced that they would be investing \$100 million into voice-related technologies to help developers both build and use Alexa’s skills.⁵³ In early 2016, Microsoft launched the Microsoft Bot Framework, which allows developers to create cross-platform chatbots that leverage Microsoft’s AI services. Microsoft’s Satya Nadella even referred to chatbots as the next big thing.⁵⁴ Later that same year, Facebook’s Mark Zuckerberg claimed that chatbots would be the solution to app overload.⁵⁵ “*Bots for Messenger*” was introduced at F8 (replacing the codename “*Agents for Messenger*”), then Facebook launched their own bot building service.⁵⁶ IBM also released Watson Conversation, a cloud-based system that allows developers to build custom chatbots.⁵⁷

Other companies also offer a variety of bot-building services to support the increased demand for bots [42]. These creation platforms support the design and development of bots and provide a range of software foundations, frameworks, toolkits, APIs, and other advanced features (e.g., natural language processing, search, or image processing). Many vibrant online communities have also emerged to connect bot developers with expertise in the form of tutorials, articles, discussions, and support [42].

2.4 Summary

In this chapter, I described a variety of technologies from which modern software bots have descended and examined possible factors contributing to their sudden re-emergence and proliferation. In doing so, it became obvious that the modern software bot has evolved from a complex landscape of bot-like technologies. As Leonard puts it, “*[t]he bot family tree is a confused and contradictory plant, a warped and twisted structure as unlike Darwins great Tree of Life as a blackberry bush is unlike a weeping willow*” [3].

Although understanding the complex history of software bot technologies can help us to better understand today’s software bot ecosystem, it provides little insight into the bots themselves. *What is a bot? What properties can bots have? How do bots behave?* In the remainder of this thesis, I attempt to answer these questions by exploring existing definitions of software bots, providing an updated definition (cf. Chapter 3), and examining their multitude of properties and behaviours as I create a taxonomy of software bots (cf. Chapters 4-5).

-
- ¹²<https://www.nasa.gov/feature/automaton-rover-for-extreme-environments-aree>
- ¹³<https://www.apple.com/ca/ios/siri/>
- ¹⁴<https://developer.amazon.com/alexa?cid=a>
- ¹⁵<https://assistant.google.com/>
- ¹⁶<https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/>
- ¹⁷<http://www.cleverbot.com/>
- ¹⁸<https://www.ibm.com/watson>
- ¹⁹<https://www.technologyreview.com/s/608911/is-ai-riding-a-one-trick-pony/>
- ²⁰<https://www.nytimes.com/2005/10/14/technology/behind-artificial-intelligence-a-squadron-of-bright-real-people.html>
- ²¹<https://aws.amazon.com>
- ²²<https://azure.microsoft.com>
- ²³<https://cloud.google.com>
- ²⁴<https://www.ibm.com/cloud/>
- ²⁵<https://www.nirandfar.com/2015/07/the-message-is-the-medium-3-reasons-apps-as-assistants-work.html>
- ²⁶<https://www.messenger.com/>
- ²⁷<https://www.skype.com/en/>
- ²⁸<https://www.whatsapp.com/>
- ²⁹<https://telegram.org/>
- ³⁰<https://web.wechat.com/>
- ³¹<https://www.kik.com/>
- ³²<https://www.slack.com/>
- ³³<https://products.office.com/en-ca/microsoft-teams/>
- ³⁴<https://www.stride.com/>
- ³⁵<https://www.flock.com/>
- ³⁶<https://www.google.com/search/about/>
- ³⁷<https://www.microsoft.com/en-ca/windows/cortana>
- ³⁸<https://www.amazon.ca/echofamily>
- ³⁹<https://store.google.com/home>
- ⁴⁰<https://www.apple.com/ca/homepod/>
- ⁴¹<https://history.apievangelist.com/#Understanding>
- ⁴²<https://www.programmableweb.com/about>
- ⁴³<https://open.canada.ca/>
- ⁴⁴<https://open.alberta.ca/>
- ⁴⁵<https://data.gov.bc.ca/>
- ⁴⁶<https://www.ontario.ca/search/data-catalogue>
- ⁴⁷<https://www.donneesquebec.ca/>
- ⁴⁸<https://webscope.sandbox.yahoo.com/>
- ⁴⁹<https://www.kaggle.com/datasets>
- ⁵⁰<https://www.atlassian.com/atlascamp/2013/archives/thursday/scratch-your-own-itch>
- ⁵¹<https://hubot.github.com/>
- ⁵²<https://developer.github.com/v4/reference/object/bot/>
- ⁵³<https://developer.amazon.com/alexa-fund>
- ⁵⁴<https://www.businessinsider.com.au/microsoft-to-announce-chatbots-2016-3>
- ⁵⁵<https://www.wsj.com/articles/facebook-hopes-chatbots-can-solve-app-overload-1460930220>
- ⁵⁶<https://techcrunch.com/2016/04/12/agents-on-messenger/>
- ⁵⁷<https://www.ibm.com/watson/services/conversation/>

Chapter 3

Defining Software Bots

In this section, I introduce some basic terminology used to describe software bots, and provide a formal definition of software bots, which I reference and build upon for the remainder of this thesis. I also reflect on common subtypes of software bots and compare them to related technologies. I then explore some existing approaches for classifying software bots.

3.1 What Is a Software Bot?

Despite their increasing popularity, there is no generally accepted definition of software bots. Research into software bot technologies spans across multiple computer science areas (e.g., AI, software engineering, or systems) as well as other disciplines, in particular decision science, biology, sociology, and many others [43]. This serves to further increase the difficulty of pinpointing what exactly constitutes a bot.

“A bot is a software version of a mechanical robot. Like a mechanical robot, it is guided by algorithmic rules of behaviour [...] But there is no consensus on what particular sequence of encoded ones and zeros truly classifies a bot. Bot genetic structures remain inadequately mapped. The word bot describes everything from a simple logon script (like one that might save a user the trouble of typing a phone number, a password, and a user identification code every time the user wants to go online) to complex programs written in the latest, most-advanced programming languages and designed to execute tasks that most humans would find impossible.”

—Andrew Leonard, 1997 [3]

In response to this ambiguity, researchers and practitioners have defined bots in accordance with their specific applications of software bot technologies. Below, I highlight some key trends in how people are describing software bots.

Bots as Automation: One of the more common ways of defining bots is as software programs that **automate tasks** [44, 45]. The Merriam-Webster dictionary defines bots as “a computer program that performs automatic repetitive tasks”. Similarly, “[b]ots are software programs that perform automated, repetitive, predefined tasks. These tasks can include almost any interaction with software that has an API.” [46] Some have taken it a step further and defined bots by their **autonomy** to perform these automated tasks: “A bot is a type of automated technology that’s programmed to execute certain tasks without human intervention. That is, it might be prompted by a human to perform an action, but it can carry it out on its own.”⁵⁸ Others define bots by their ability to perform actions on behalf of others: “A bot is a program that operates automatically as an agent for a user or another program.” [47]

Bots as Malicious: Unfortunately, many people view these behaviours as being riddled with the **mal-intent** of the bot’s creator [46]. The Merriam-Webster dictionary emphasizes the term bot refers “especially” to computer programs “designed to perform a malicious action”.⁵⁹ Bruce Schneier, the founder of Counterpane Internet Security, believes that bots “deserve special scrutiny because their risks are different than normal risks. Bots are risky because they do what they do automatically, and lots of them can work in tandem. So the relatively minor damage they can do—spam, worms, and so on—becomes nasty because a lot of it happens.” [48]

Bots as Human-Like: Luckily, as bots continue to become more pervasive in everyday life, people have started seeing bots as more than simply malicious scripts. People have begun to enjoy interacting with bots, especially ones that have been given life-like personalities. The Oxford dictionary defines bots by their ability to act and be perceived **humanly**: “An autonomous program on a network (especially the Internet) which can interact with systems or users, especially one designed to behave like a player in some video games.”⁶⁰ Similarly, Maus [49] who defines bots as “automated or largely automated programs that interface with online platforms in largely the same way that a typical human would be expected to: they hold normal accounts, make connections, and post content”. Slack goes as far as claiming that “[b]ots are like having a virtual team member—they can help you manage tasks, run your team standup, poll the office, and more!”⁶¹

Bots as Conversationalists: Many large software companies, like IBM, Google, and Microsoft, have also been pushing to make their bots sound more human.⁶² It is not surprising that the definition of bots is often coupled with their ability to **communicate using human language**. Microsoft defines bots as “an app that users interact with in a conversational way using text, graphics (cards), or speech”.⁶³ Similarly, Dale describes

the human-bot relationship as “*achiev[ing] some result by conversing with a machine in a dialogic fashion, using natural language*” [38].

Bots as Interfaces: In an earlier blog post, Amir Shevat described bots as “*digital users within a messaging product. Unlike most users, they are powered by software rather than by a human, and they bring a product or service into a given messaging product via the conversational interface.*”⁶⁴ Since then, Shevat removed the requirement of conversation from his definition of software bots: “[T]he bot itself is only an interface into the service, in the same way a website can be used for booking a flight...exposing a service.” [39] This approaches bots as an **interface** paradigm, or as Roy [50] describes them, “*the bridge between data and action*”.

While a variety of definitions of software bots, there are several key features that are consistent across many of the approaches. In the next section, I explore the similarities between the various interpretations of software bots and propose an updated definition of software bot-hood.

3.1.1 Proposed Definition of Software Bots

My proposed definition of software bots builds upon many of the existing definitions of software bots described in the previous section. First and foremost, I view software bots as a new **interface paradigm**. Bots connect **users** with **software services**. While bot users are often humans, they are not required to be. Users can be programs, systems, or even other bots. The bot is the interface that *provides* the services to the user, e.g., the bot is everything required to *present* the service to the user. However, the bot and the service can and should be decoupled from each other.

Software services are “*a mechanism to enable access to one or more capabilities*”.⁶⁵ Software bots utilize software services for the raw value they provide. Services provide software functionality (or a set of software functionalities) in a format that can be reused by multiple clients for a variety of purposes.⁶⁵ Today, services can come in many forms, ranging from the retrieval of information to the execution of a set of operations. Often the bot performs tasks that rely on these services repetitively, saving the user time through automation.

Software services can be **external** or **internal** to the bot. Figure 3.1 shows the domain of (a) software bots, (b) software services, and (c) the relationship between them. When services are external to the bot, the bot has no ownership of the software services it requires to perform its tasks. External services can be thought of as online services, i.e., they are connected to something external (e.g., network or Internet). The bot must access the external service to provide its functionality. Figure 3.1(a) shows the domain of software bots that access (represented by the dashed arrow) external software services. When services

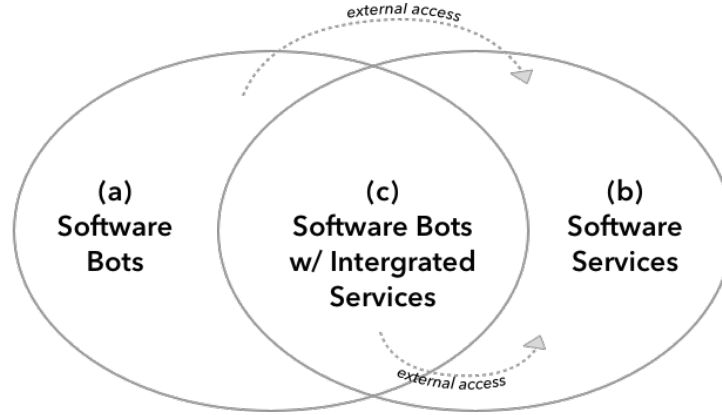


Figure 3.1: The relationship between software bot interfaces and software services.

are internal to the bot, the bot itself owns the software services that it requires to perform its tasks. Figure 3.1(c) shows the domain of software bots that have integrated software services. Internal services can be thought of as offline services; if disconnected from the outside world, these bots would still be able to deliver their services. Lastly, a bot can have a combination of internal and externally accessed services (represented by the second dashed arrow). These bots would be able to deliver some (but not all) of their functionality if they were disconnected from their outside services.

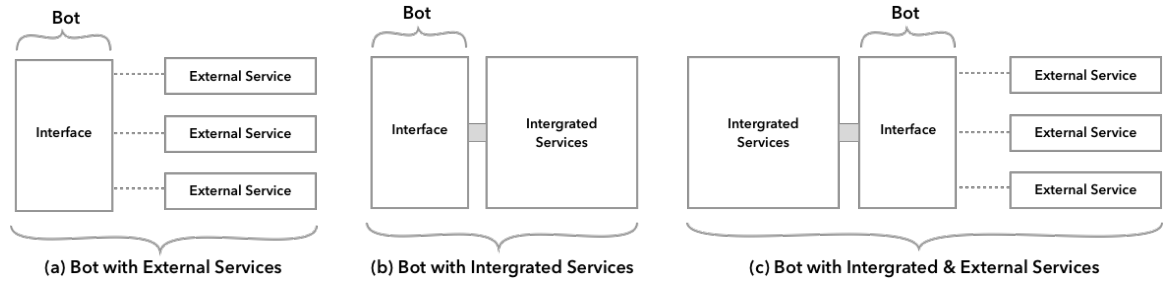


Figure 3.2: The relationship between software bot interfaces and software services: (a) software bot with external services, (b) software bot with internal services, and (c) software bot with both internal and external services.

Figure 3.2 provides another view of the same relationship between bots and services. In this figure, we can clearly see the distinction between the software bot's interface and the services it provides. Figure 3.2(a) shows a software bot interface that accesses set of external software services. An example of this type of bot is the github⁶⁶ bot which accesses GitHub's API. Figure 3.2(b) shows a software bot with a set of integrated software services. An example of this type of bot is Eliza as its conversational logic is internal to the bot. Lastly, a bot can have a combination of external and internal services. Figure 3.2(c) shows a software bot with integrated software services and a set of connected external services. An

example of this type of bot is poncho⁷³, a bot that accesses weather reports (i.e., external services) and offers a variety of games (i.e., internal services).

If software bots provide an alternative interface to services, then what exactly does the software bot **interface** entail? Simply put, an interface is where two things meet [51]. The software bot interface is where the user and the bot’s services meet. The software bot interface also usually provides some form of **additional value** on top of its services. This additional value can come in many forms, from lowering the barrier of access to consolidating multiple services to providing automation. Bots often leverage the recent advances in user interfaces to provide additional value, usually through changing the interaction style. Figure 3.3 shows the mainstream adoption of new human-computer interface paradigms over the past few decades. Today, users can interact with software bots via the command line, graphical interfaces, touch interfaces, spoken/written language, or a combination of interaction paradigms. It should be noted, however, that these interfaces are not required to be interfaces that humans use; the software bot’s interface can be used by other bots or other types of software systems. Another common way that software bots are providing additional value is through *anthropomorphism*—making the user’s interactions with the software services more enjoyable by making it more human. There are many ways in which people anthropomorphize software bots; giving them names, personalities, emotions, etc.

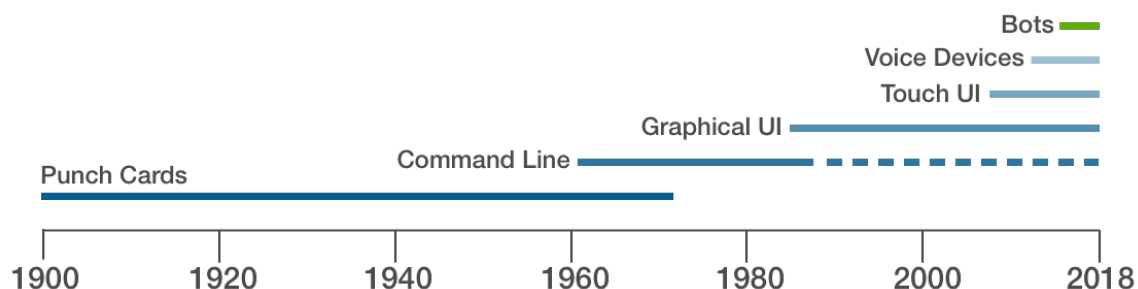


Figure 3.3: A time-line of the emergence and mainstream adoption of new user interface paradigms, adapted from Ryan Block¹.

So, what is a bot? In summary, I define a software bot as an *interface* that connects users to *services*. These services can be *internalized* in the bot’s code and/or accessed *externally*. The bot also provides some sort of *additional value* (in the form of interaction style, automation, anthropomorphism, etc.) on top of the software service’s basic capabilities.

¹<https://medium.com/@ryan/bots-messaging-and-the-interface-visibility-scale-c77ce56f1401>

3.2 Comparing Software Bots and Related Technologies

In the previous section, I proposed an updated definition of software bots, however, it is important to understand how software bots relate to other bot-like technologies to truly understand what makes a bot, a “bot”.

“The semantics of botness are confused and have yet to be satisfactorily sorted out [...] But whatever you call them—agents or bots or scripts or plain old programs—they are a genus, a class of species, of their own.”⁶⁷

—Andrew Leonard, 1996

It should be noted that the purpose of this section is not to universally define these bot-like technologies themselves, but instead use these definitions to gain a better understanding of the complexities of defining software bots. I also acknowledge that there will likely be some technologies that fall between the definitions that I propose, which I will not be addressing.

3.2.1 Software Bots vs. Robots

As discussed in Chapter 2, traditionally software bots and robots have been distinguished from each other primarily by where they act: robots act in the physical world and bots act in the digital world. However, as robots grow more sophisticated (e.g., using machine learning or other algorithmic techniques) and bots start connecting to physical devices (e.g., internet of things enabled devices) the difference between robots and bots becomes more subtle.

Both robots and software bots can perform tasks such as unlocking your front door. To complete this task, the robot would physically make its way over to the door and turn the deadbolt. The software bot, on the other hand, would send an unlock request to the API of the smart-lock. Thus, the robot performed the task physically itself, while the software bot sent a software request to another device that performed the physical action.

I imagine that as robots and software bots continue to grow more advanced, the difference between them may continue to grow more blurred. However, for the scope of this thesis, I treat robots and software bots as two distinct technologies.

3.2.2 Software Bots vs. Scripts

Though there is no universally accepted definition of a script, they tend to be described as small pieces of software (usually just a small set of commands) that do not perform any significant computing on their own.⁶⁸ Scripts are commonly associated with code written in

a scripting language (which are typically interpreted not compiled) although some exceptions do apply. Others define scripts by the way they behave: “A ‘script’ is code that acts upon some system in an external or independent manner and can be removed or disabled without disabling the system itself.”⁶⁸ These types of scripts are generally associated with automation or logging, and are typically run via the command line (i.e., shell scripts).

This ambiguity only adds to the difficulty of trying to understand the relationship between scripts and software bots. Although domain specific scripting languages (e.g., IFTTT⁶⁹) have been developed to help users build bots, to the best of my knowledge, the difference between scripts and bots has never concisely been defined. I propose that *some scripts are bots*, and conversely *some bots are scripts*. Some bots clearly do not fit within the requirements of a script; they are large complex programs that are written in a compiled language.

However, the difference between pure scripts and bots that are scripts is often more subtle. I believe that the main difference stems from the intent of the developer. For example, providing a script with some additional value in the form of anthropomorphism (e.g., giving it a name or personality) can be the difference between a bot and a non-bot script. There are many other small differences between scripts and bots that can also help to distinguish between them. For example, most scripts are triggered from the command line, however, this is less common for bots. While scripts tend to be re-executed each time they are needed, bots tend to persist and perform tasks over a longer period of time. While these differences are highly subjective and often quite subtle, they provide some guidance in distinguishing between scripts and software bots.

3.2.3 Software Bots vs. Programs

Today the line between scripts and programs is blurred. Some view scripts as a small, special subtype of computer programs.⁷⁰ Others view scripts and computer programs as two separate entities.⁶⁸ Although there is no universally accepted definition of a program, they tend to be described as larger pieces of software, written in programming languages (i.e., compiled), that perform significant computing.⁶⁸

If we consider scripts and programs to be two mutually exclusive types of software, then *some programs are bots*, and conversely *some bots are programs*. Some types of programs are clearly not bots (e.g., word processing software or other large desktop applications). Figure 3.4 shows the relationship between software bots, programs, and scripts.

However, the difference between bots and programs is often more subtle. There are two main factors that distinguish bots from non-bot software programs. The first factor that distinguishes them is that bots must have some level of autonomy, i.e., bots must be able to do something without the user having to use direct manipulation to perform the tasks themselves. With the most basic of bots, autonomy is realized through automation. On the

other end of the spectrum, highly autonomous bots are free to act according to their own will. Another essence of bot-hood is the anthropomorphizing of the bot interface. People interact with bots differently than they interact with other computer systems, and bots are seen as almost pseudo-life-like entities. While these differences are highly subjective and often quite subtle, they provide some guidance in distinguishing between programs and software bots.

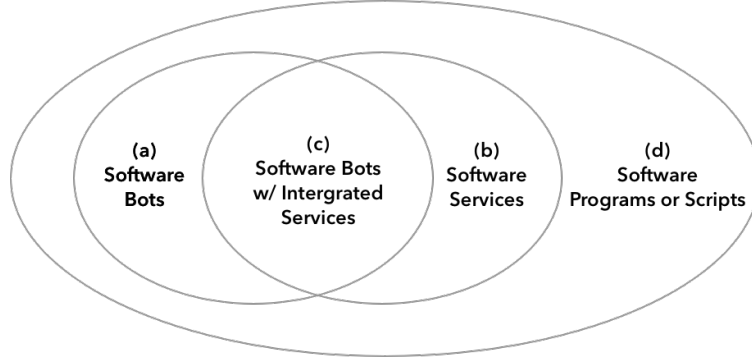


Figure 3.4: The relationship between (a) software bots, (b) software services, (c) software bots with internal services, and (d) software programs/scripts.

It is worth noting that computer applications (apps) are commonly seen as a subtype of programs, and are distinguished from the larger domain of programs through being specifically designed for a human end user (e.g., they possess things like a user interfaces) and being operating system dependant.⁷¹ However, for the scope of this thesis, I do not formally distinguish between apps and programs.

3.2.4 Software Bots vs. Agents

Some researchers do not distinguish between bots and agents. Wooldridge and Jennings, for example, believe that *“bot [is] another term for agent, usually one implemented in software”* [52]. Others think of software bots and software agents as two distinct software technologies. I proposed, however, that software agents are a subtype of software bots. That is, *all agents are bots*, but *not all bots are agents*. Then what distinguishes software agents from software bots?

Like software bots, there is little consensus on what exactly constitutes an agent as *“[t]he term ‘agent’ has been picked up, widely appropriated, and in many cases misappropriated, by technical publications, lay publications, and many researchers in computer science”* [53]. Although there are numerous definitions of agent-hood in the previous literature, I focus my attention on some key definitions of software agents.

Perhaps the loosest definition of agents is that of Russell and Norvig [5], who describe an agent as any program whose outputs are determined by its inputs: *“An agent is any-*

thing that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.” [5] Others, like Hayes-Roth, equate agents with intelligence and reasoning abilities: “*Agents continuously perform three functions: perception of conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions.*” [54] Wooldridge and Jennings attributed the following properties to agent-hood: autonomy, social ability, reactivity, pro-activeness [28]. Tasic and Agha distinguish between *weak agency* (i.e., control of own state, reactivity, and persistence) and *strong agency* (i.e., weak agency, goal-orientation, and pro-activeness) [55].

Since the scope of this thesis is not about defining *agent-hood*, I will focus on the relationship between software agents and software bots. Software agents are a subtype of software bots, that fulfill a set of minimum *requirements for agency*. These requirements usually include *sensing, acting, intelligence, autonomy, social-ability*, however, they can vary slightly based on the definition of software agency you choose to adopt. To realize these requirements (specifically autonomy), software agents typically have some form of internal services, as shown in Figure 3.5(e). An example of this type of bots are the agents that made-up early mutli-agent systems [52]. However, with the latest advances in AI, more and more software bots are trending towards agent-hood.

3.2.5 Software Bots vs. Chatbots

Many people commonly (yet incorrectly) use the term chatbot to refer to all software bot technologies, and vice versa. Although many of the popular mainstream bots do in fact have some form of language capability, possessing the ability to understand language is not a requirement for software bots. That is, *all chatbots are bots*, but *not all bots are chatbots*. More specifically, chatbots are distinguished from the greater domain of software bots by their ability to use language to communicate with their users. Chatbots are also commonly referred to as *chatterbots*, *talkbots*, or *artificial conversational entities*.

Not surprisingly, there is also a lack of agreement on what constitutes a chatbot. Broadly, a chatbot can be described as any software bot with a conversational interface [46]. This definition includes most of the common mainstream chatbots we see today, e.g., Slack’s original helper bot, Hubot.⁷² More narrowly, a chatbot can be described as a software bot that was designed purely to hold conversations with humans [46], e.g., Eliza, the original chatbot [56]. For the scope of this thesis, I will be adopting the more broad definition of chatbots as it is more widely used.

As shown in Figure 3.5, chatbots can either have their own internal services (e.g., Eliza [56]) or access to external services (e.g., Poncho⁷³). Some chatbots may also be agents (cf. Figure 3.5e/f) when they possess the minimum requirements for agency as well as the

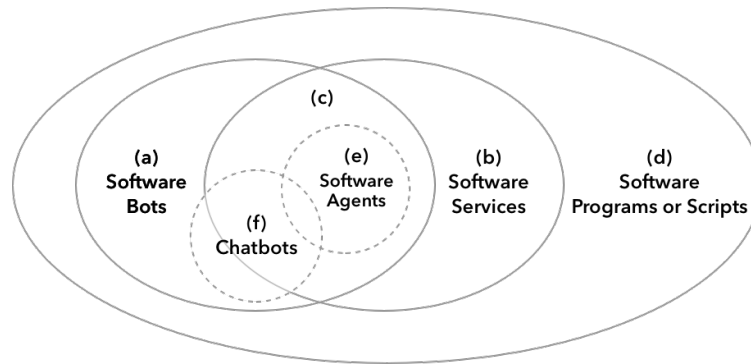


Figure 3.5: The relationship between (a) software bots, (b) software services, (c) software bots with internal services, (d) software programs/scripts, (e) software agents, and (f) chatbots.

ability to communicate with some form of human-understandable language, e.g., personal assistant bots such Apple’s Siri⁷⁴ or Amazon’s Alexa.⁷⁵

3.3 Existing Classifications of Software Bots

The lack of consistent terminology surrounding software bots has not stopped people from trying to make sense of them. In this section, I identify and describe some common ways of classifying software bots.

3.3.1 High-Level Classifications (Subtypes)

One of the most common ways that people try to classify software bots is through their set of specialized behaviours. These classifications build upon the basic requirements of software bot-hood in one or more ways. Throughout the remainder of the thesis, I will refer to this type of classification as software bot *subtypes*. **Software agents** and **chatbots**, as discussed earlier, are good examples of enhanced bot behaviours that have earned their own classifications. Examples of other types of high-level classifications of software bots include: **web bots**, software bots that perform their tasks over the internet; **embodied agents**, agents that have a life-like digital form; **conversational agents**, chatbots that satisfy the requirements of agency; **intelligent agents**, software agents that typically include more advanced AI capabilities; **sapient agents**, software agents that make informed decisions; **rational agents**, software agents that do the *right* thing [5]; and various combinations of these behaviours.

3.3.2 Role-Based Classifications

People have also tried to classify software bots through their purpose or role [57]. Many common bot catalogues (e.g., BotList,⁷⁶ Chatbottle,⁷⁷ or Discord⁷⁸) include some form of role-based classification to help users browse and discover new software bots.

One common way of examining software bots' roles is through the lens of *good vs. bad* bots [46]. I introduce some common role-based classifications for both good and bad bots below. It should be noted that these categories are not intended to be mutually exclusive, nor are they an exhaustive list of all possible bot roles.

There are many examples of bots that work cooperatively with humans to help them complete tasks [46]. An example of such bots are **crawlers**, bots which run continuously in the background to fetch and often store data from websites, APIs, etc. Search engines employ them to crawl the web, collecting the documents required to build a searchable index by following the hyperlinks on each page. GoogleBot⁷⁹ and BingBot⁸⁰ are two prominent examples of search engine bots. **Information bots**, often referred to as **news bots**, help bring information to users, often in the form of notifications. **Transactional bots** work on the users behalf, interacting with external systems to automatically execute transactions when a condition is met. **Productivity bots** work to improve user or team productivity by automating simple office tasks. **Collaboration bots** help users communicate, coordinate, and collaborate. Other bots, such as **art bots** and **game bots**, are designed to keep users entertained.

Unfortunately, a software bot's role can also be bad [46]. **Hacker bots** were developed to distribute malware, exploit security vulnerabilities, and organize botnets. Although not always malicious, **scrapers** are used to download data from the web, sometimes with the intention to republish the stolen content elsewhere. **Spammers** post promotional content in attempts to drive traffic to a specific site. Bots can also act as impersonators, either masking their identities to look like network traffic (e.g., for DDoS attacks) or pretending to be someone else.

Storey and Zagalsky proposed a role-based classification to discuss how bots are being used in software development [45]. **Testing bots** help detect bugs and code quality issues in the project. **Code bots** help make coding activities more efficient and effective. **Documentation bots** produce documentation from developer artifacts. **Translation bots** can generate documentation in many languages. **Support bots** interact with the user to answer simple questions by consulting a knowledge base. **Operations bots** help automate deployment and operations tasks. The use of bots in a development operations context is often referred to as ChatOps.

In previous research, I explored the roles that bots can play in reducing the *types of friction* software developers face when working collaboratively [58]. Using a socio-technical

model for collaborative work (cf. Figure 3.6, we identified three main roles that software bots can play: reducing (a) a team’s interaction friction, (c) an individual’s interactions with technology friction, and (d) a team’s interactions with technology friction.

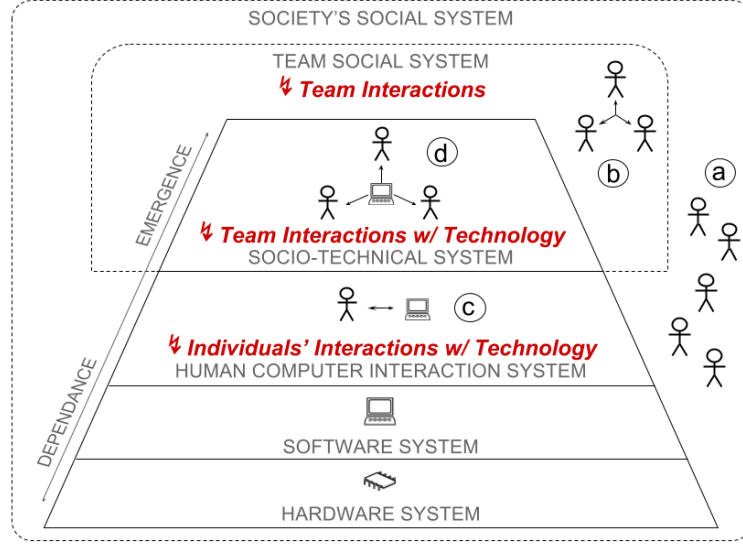


Figure 3.6: Socio-Technical Model for Collaborative Software Development including the (a) society’s social system, (b) team’s social system, and three categories of friction that bots can help reduce: (b) team’s interactions, (c) individual’s interactions with technology, and (d) team’s interactions with technology.

3.3.3 Other Classifications

Bots can also be classified by where they live (i.e., their hosting platforms) [42]. These platforms dictate where and how the bots are accessed by end users. Common types of platforms include social networking (e.g., Facebook Messenger, Microsoft’s Skype, or WeChat) or domain-specific channels (e.g., Slack,⁸¹ Microsoft Teams, or Hipchat). Many common bot catalogues also include some form of hosting platform-based classification.^{82, 78} Similarly, bots can also be described by the platforms in which they were created [42]. These creation platforms (e.g., Microsoft Bot Framework,⁸³ BotKit,⁸⁴ PandoraBots,⁸⁵ or ProBot⁸⁶) support the design and development of bots.

One step towards a multi-faceted classification of bots is the dimensions proposed by Storey and Zagalsky who identified that software bots can differ across five dimensions: how *autonomous*, how *intelligent*, how do they *interact*, how are they *created*, where do they *reside*, and what do they *do* [59]. They also stressed the importance for examining how the use of bots by developers benefits developers by studying bots on two main dimensions: *efficiency*, supporting developers doing things faster; and *effectiveness*, helping developers move towards meaningful goals.

3.4 Shortcomings of Existing Classifications

Much of the recent effort towards understanding software bots has been focused on classifying them by a single aspect (e.g., their role or platform). However, as software bots continue to grow more complex, high-level classifications are likely to overlook many of the properties that make each bot unique. For example, what features distinguish one chatbot from another? Some classifications focus on technologies used to build bots (e.g., the algorithms or communication protocols used). However, in a field that is developing rapidly, these technology-based classifications quickly become out of date.

Existing classifications that explore a variety of properties and behaviours of software bots do exist, which are discussed in detail in Chapters 4–7. Much of this research, however, focuses only on a specific subtype of bot (e.g., software agents) or a narrow set of properties/behaviours. When examining only a subset of software bots, it is easy to miss the wide range of behaviours bots can have. Although Storey and Zagalsky [59] provide a step towards a holistic view of software bot behaviours, their proposed classification is still very high level.

3.5 Summary

In this chapter, I explored a variety of existing trends for describing bots. Building on these previous descriptions, I derived an updated definition of software bots and provided a comparison of modern software bots to a variety of bot-like technologies (i.e., scripts, programs, applications, agents, and chatbots), highlighting both their similarities and differences.

To further understand *what is a software bot*, I also explored previous ways of classifying bots. The remainder of this thesis describes the creation of a new taxonomy of software bots, aimed at exploring the emergent behaviours and properties of software bots as a whole. The work presented in this thesis builds upon these existing classifications of software bot sub-types to develop a more holistic classification of modern software bots.

⁵⁸<https://blog.hubspot.com/marketing/where-do-bots-come-from>

⁵⁹<https://www.merriam-webster.com/dictionary/bot>

⁶⁰<https://en.oxforddictionaries.com/definition/bot>

⁶¹<https://slack.com/apps/category/At0MQP5BEF-bots>

⁶²<https://www.theverge.com/2018/5/21/17375482/microsoft-semantic-machines-acquisition-bots-cortana-human>

⁶³<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-3.0>

⁶⁴<https://venturebeat.com/2018/05/21/researchers-use-vr-to-train-ai-drones-cutting-autonomous-vehicle-crashes/>

⁶⁵<https://www.oasis-open.org/committees/soa-rm>

⁶⁶<https://www.slack.com/apps/A8GBNUWU8-github>

⁶⁷<https://www.wired.com/1996/04/netbots/>

- ⁶⁸<http://preserve.mactech.com/articles/mactech/Vol.15/15.09/ScriptingLanguages/index.html>
- ⁶⁹<https://ifttt.com/>
- ⁷⁰<https://whatis.techtarget.com/definition/script>
- ⁷¹<https://www.pcmag.com/encyclopedia/term/37919/application-program>
- ⁷²<https://hubot.github.com/>
- ⁷³<http://www.poncho.is>
- ⁷⁴<https://www.apple.com/ca/ios/siri/>
- ⁷⁵<https://developer.amazon.com/alexa?cid=a>
- ⁷⁶<https://botlist.co/>
- ⁷⁷<https://chatbottle.co/>
- ⁷⁸<https://discordbots.org/tags>
- ⁷⁹<https://www.google.com/search/howsearchworks/crawling-indexing/>
- ⁸⁰<https://www.bing.com/webmaster/help/which-crawlers-does-bing-use-8c184ec0>
- ⁸¹<http://www.slack.com>
- ⁸²<https://botlist.co/>
- ⁸³<https://dev.botframework.com/>
- ⁸⁴<https://www.botkit.ai/>
- ⁸⁵<https://www.pandorabots.com/>
- ⁸⁶<https://github.com/probot>

Chapter 4

Developing a Software Bot Taxonomy

Classification is a powerful tool. Through the development of a software bot taxonomy, I aim to provide a set of consistent terminology for understanding the observable properties and behaviours of software bots. This taxonomy views software bots from multiple perspectives, rather than classifying them by a single property (e.g., their role or purpose). This allows for the flexible comparison of bots, as well a taxonomy that can easily be updated/extended in a rapidly changing field.

This chapter describes the generation of the software bot taxonomy. Throughout this process, I reflected and built upon past research on software bots and their predecessors. Although the taxonomy I generated provides a set of consistent terminology for future work in the field of software bots, I also include a mapping of alternate terminology for understanding past work.

4.1 Taxonomy Generation Methodology

This section provides a high-level overview of the methodology I used to generate the taxonomy of software bots that I propose in this thesis. I adapted Usman et al.'s taxonomy development methodology [1]. As shown in Figure 4.1, their suggested methodology is comprised of four phases: (a) planning, (b) term identification and extraction, (c) design and construction, and (d) validation.

The taxonomy generation methodology I followed deviates slightly from the methodology proposed by Usman et al. [1], as shown in Figure 4.2. My adapted methodology expands the original four into six distinct phases: (a) Planning (b) Data collection (c) Term identification and extraction (d) Design and construction (e) Validation (d) Usage guidelines A description

of the changes made to the methodology, as well as the reasons for the adaptations, are described below.

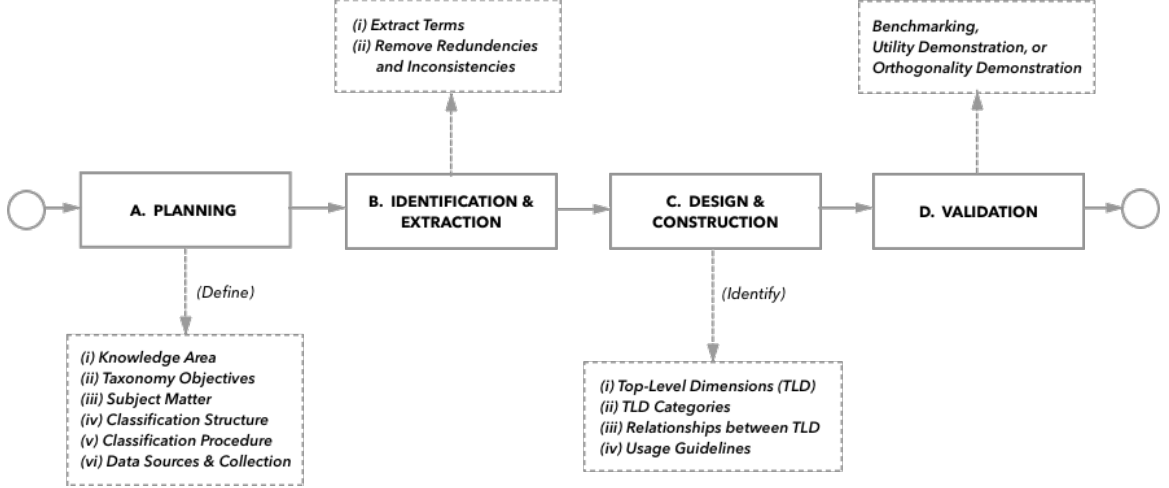


Figure 4.1: Usman et al.'s methodology for taxonomy generation [1]

The first phase was the **planning** phase (Figure 4.2A). Similar to Usman et al.'s original methodology, this phase included identifying the taxonomy's (i) *software engineering knowledge area*, (ii) *scope and objectives*, (iii) *subject matter*, (iv) *classification structure*, and (v) *classification procedure*. However, in Usman et al.'s original methodology, data identification and collection is a single step in the planning phase. Since data collection was a complex, multi-stage process in my taxonomy generation, I divided it into its own distinct **data collection** phase (Figure 4.2B). To get a robust set of articles from which to extract terms, I conducted a (i) *systematic literature search*, (ii) *search results snowballing*, and an (iii) *online search* for recent bot-related articles.

This was followed by the **identification and extraction** phase (Figure 4.2C). Similar to Usman et al.'s original methodology, the first step in this phase was also (i) *term extraction*. However, in their original methodology, redundancy and inconsistency removal were performed in the same step. In my methodology, these are broken into two distinct steps. First, (ii) *redundancies were removed* by grouping like terms or synonyms together. Then, any (iii) *inconsistencies or terms not satisfying the goals of the taxonomy were removed*. It should be noted that steps (ii) and (iii) were performed iteratively and continued throughout the remainder of the taxonomy generation process.

The fourth phase was **design and construction** (Figure 4.2D). Usman et al.'s original methodology suggests a top-down approach for taxonomy generation. Since I had few insights on how the taxonomy should look, I used a bottom-up approach to help reveal which higher level categories would best capture the range of software bot behaviours. For this reason, I included an additional (i) *card sorting step* which allowed the basic structure

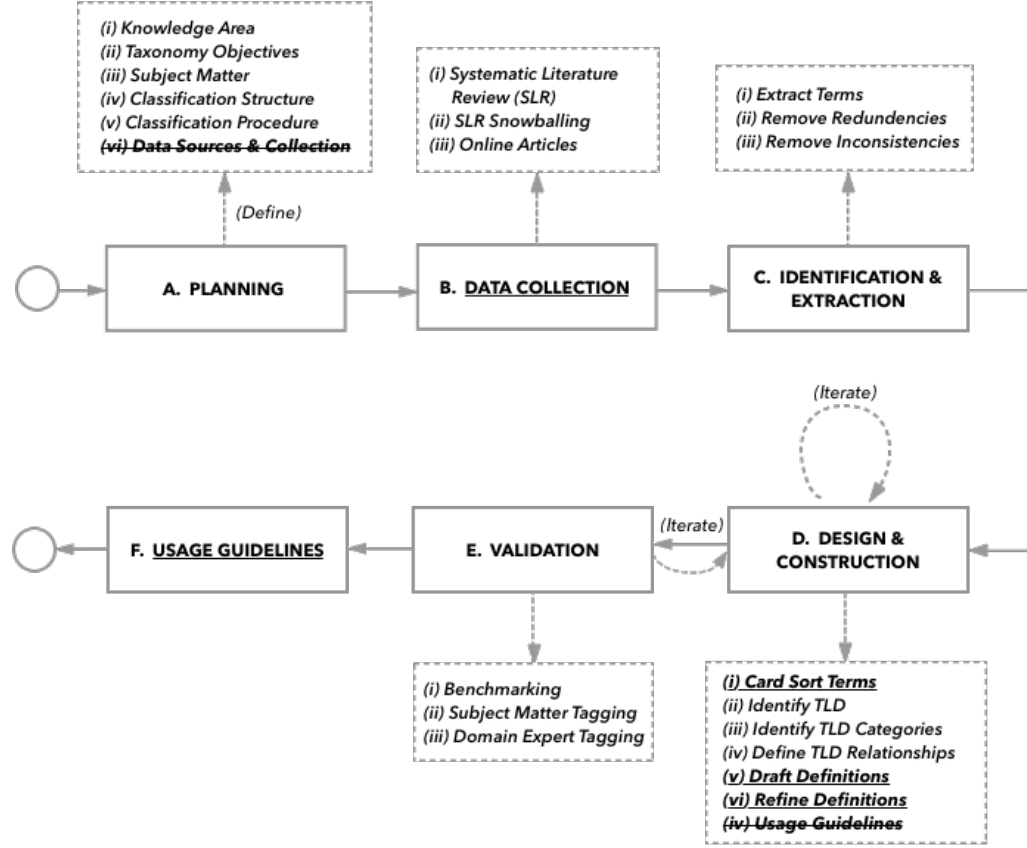


Figure 4.2: Methodology for creating the updated taxonomy, adapted from Usman et al. [1]. The underlined/strikeout text depicts steps that were added or removed, respectively.

of the taxonomy to emerge. Next, I identified and described the *(ii) top-level dimensions*, the *(iii) categories for each dimensions*, and any *(iv) relationships between dimensions*. I also included an additional iterative *(v) refining the taxonomy* step. At this point, Usman et al. included a step for generating usage guidelines; however, I decided to move this step later in the process. This allowed me to tailor the usage guidelines to better fit the needs of potential users based on the results of the validation phase.

The fifth phase, **validation** (Figure 4.2E), had three steps. First, I performed *(i) benchmarking* by comparing the proposed taxonomy against existing software bot taxonomies. Next, I conducted an *(ii) expert tagging session* where a software bot developer used the proposed taxonomy to tag their own bot. Lastly, I performed *(iii) subject matter tagging* with three software bots. Throughout the validation phase, I iterated on and updated the taxonomy as required.

The final phase was the creation of **usage guidelines** (Figure 4.2F). The usage guidelines describe the structure of the proposed taxonomy and identify how it can be used.

In the following sections, I describe each of the six taxonomy generation phases in greater detail.

4.2 Planning (Phase 1)

Prior to beginning the taxonomy construction, the planning phase helped define the context of the new taxonomy [1]. In this phase, I describe the basic taxonomy design decisions: the knowledge area (Section 4.2.1); taxonomy objectives (Section 4.2.2); subject matter (Section 4.2.3); classification structure (Section 4.2.4); and classification procedure (Section 4.2.5).

4.2.1 Knowledge Area

Identifying the knowledge area helps describe the context of the taxonomy [1]. I used the Software Engineering Body of Knowledge (SWEBOK) to determine the knowledge area which the taxonomy contributes to [2]. The SWEBOK divides the discipline of software engineering into fifteen distinct knowledge areas [2], each of which is further broken down into multiple subareas and sub-subareas.

Software bots, like any software system, fall within the scope of the *Computing Foundations* knowledge area in SWEBOK-V3 [2].

13: “The scope of the Computing Foundations knowledge area (KA) encompasses the development and operational environment in which software evolves and executes. [2]”

Specifically, software bots can be viewed as a set of components that, when combined together, achieve some purpose or goal. I identified the sub-knowledge area for the software bot taxonomy as being the SWEBOK’s *Basic Concept of a System*.

13.8: “A system is a purposeful collection of interrelated components that work together to achieve some objective. A system can be very simple and include only a few components, like an ink pen, or rather complex, like an aircraft. [2]”

Software bots, like many other software systems, are also more than simply a sum of their parts and exhibit specific behaviours and properties as a result of the entire system working together as a whole. Based on this, I further refined the knowledge area to be the *Emergent System Properties* of the software bots systems:

13.8.1: “A system is more than simply the sum of its parts. Thus, the properties of a system are not simply the sum of the properties of its components. Instead, a system often exhibits properties that are properties of the system as

a whole. These properties are called emergent properties because they develop only after the integration of constituent parts in the system. Emergent system properties can be either functional or non-functional. Functional properties describe the things that a system does. Non-functional properties describe how the system behaves in its operational environment. [2]”

Thus, the taxonomy proposed in this thesis contributes to the SWEBOKs *Computing Foundations: Basic Concept of a System: Emergent System Properties*, as it describes emergent properties (both functional and non-functional) of software bots [2].

4.2.2 Objectives & Scope

Since previous literature on software bots failed to adequately define modern software bots, the main objective of this taxonomy was to provide an up-to-date, non-prescriptive description of the range of possible software bot properties. Thus, the main objectives of the software bot taxonomy were to provide:

1. a set of consistent, **updated terminology** to describe the properties of bots;
2. the range of **values** for each of the identified properties; and
3. the **mappings** between the updated terminology and the terminology used in existing literature.

The taxonomy was designed with researchers, practitioners, and end-users in mind. By organizing and describing software bots, the taxonomy can help *researchers* advance the knowledge area by providing a set of a common terminology to ease the sharing of knowledge. This enables a better understanding of the interrelationships between the various properties of bots, and aids in identifying gaps in the knowledge area [1]. *Practitioners* can use the properties defined in the taxonomy to better understand the variety of design choices when building software bots so they can make informed decisions when developing bots. Software bot *end-users* can also leverage this taxonomy to better understand the technologies they are interacting with, the possible risks associated with bots, and make informed decisions on which bots to adopt.

4.2.3 Subject Matter

While the knowledge area describes the broad area, the subject matter describes what is being classified. The subject matter for this taxonomy is the observable properties and behaviours of software bots. The taxonomy treats software bots as systems that can be understood by their inputs (i.e., what they understand) and outputs (i.e., behaviours, properties, characteristics), without requiring any knowledge of their internal workings. More

specifically, the subject matter for this taxonomy can be classified through observing and interacting with the bots.

I purposefully excluded implementation details to allow the taxonomy to be more easily maintained and expanded upon. To understand software bots, one does not require a detailed understanding of how the bots are built. In addition, the technologies and algorithms used to build software bots are constantly changing and being updated. Thus, including implementation details would more quickly render the taxonomy out of date.

4.2.4 Classification Structure

Taxonomies are most commonly structured as hierarchies; however, taxonomies can also be trees, paradigms, or facet based [1]. Hierarchical taxonomies have a single top-level dimension and a set of inheriting subclasses (i.e., has an “*is-a*” relationship with the parent class). Hierarchical taxonomies ensure mutual exclusivity, so each entity can only belong to a single subclass, as shown in Figure 4.3(a).

Software bots, however, are complex entities that can be viewed and classified by more than one perspective. For this reason, the software bot taxonomy employs faceted classification. As shown in Figure 4.3(b), faceted taxonomies allow for the subject matter, in this case software bots, to be independently classified across multiple properties, called facets. Each of the facets may be either decomposed into further sub-facets, or contain a set of mutually exclusive values for classifying the software bot. The facets can be either independent, or relationships can be defined between them.

Whereas hierarchical classifications are fixed and unchanging, faceted taxonomies allow for greater flexibility and expandability. Faceted taxonomies are also well suited for emerging or evolving areas, such as software bots, since they do not require complete knowledge of the area beforehand [1]. Additional facets, sub-facets, or facet values may be easily added, allowing them to evolve smoothly over time.

An example of a faceted taxonomy is the Art & Architecture Thesaurus (AAT)⁸⁷, which classifies art based on seven facets: concept, physical attributes, styles/period, agents, activities, materials, and objects. Faceted taxonomies are also commonly used for organizing web content. For example, an online marketplace that allows customers to filter based on price, colour, material, etc.

4.2.5 Classification Procedure

Since each facet or sub-facet in a taxonomy has a set of possible values, the classification procedure describes the way in which the subject matter will be assigned to a facet value. In the proposed taxonomy, the properties software bots are classified either nominally or ordinally. Nominal facets have two or more categories, but there is no clear ordering between

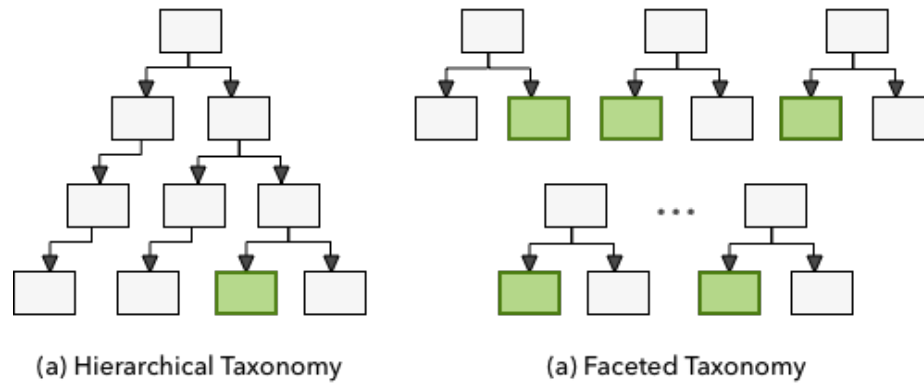


Figure 4.3: Comparison of the structure of (a) hierarchical taxonomies and (b) faceted taxonomies. The green boxes reflect the how a sample entity would be classified in each taxonomy.

them. Ordinal facets have two or more options, but there is a clear ordering between the options. However, both nominal and ordinal require the user to somewhat subjectively assign bots to the appropriate value for the facet.

4.3 Data Collection (Phase 2)

Due to the variety of research, articles, and other media on software bots, the primary data I used to generate the taxonomy was previous literature that classified software bots. To retrieve the articles, I employed three main data collection methods: a **systematic literature** search (Section 4.3.1), backwards **snowballing** (Section 4.3.2), and an **online search** (Section 4.3.3).

In the following sections, I describe each of these three data collection methods and the articles collected.

4.3.1 Systematic Literature Search

One of the main goals for the creation of the software bot taxonomy was to consolidate the terminology used in existing classifications of software bots. To find these existing classifications schemes, I conducted a systematic literature search for taxonomies that focused on the observable behaviours of software bots, following Kitchenham et al. [60] methodology. As shown in Figure 4.4, the systematic literature search had five main phases, as described below.

4.3.1.1 Selecting Terms

I identified two search terms, **software bots** and **taxonomy**, and a series of synonyms to guide the initial article collection process.

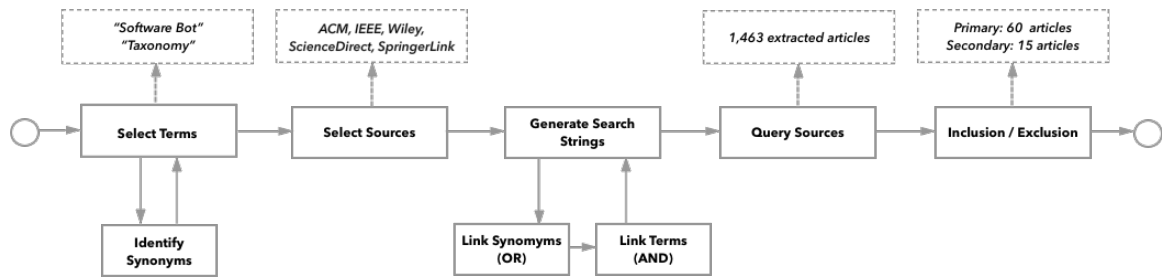


Figure 4.4: An overview of the methodology followed for the systematic literature search data collection process.

Software Bots: The subject matter of interest, software bots, are commonly referred to by many different names: chatbots, chatterbots, agents, artificial agents, autonomous agents, etc. For this reason I included the synonyms `agent*`, `chatbot*`, and `chatterbot*`, along with `bot*` in the initial search.

Taxonomy: The classification of knowledge can be described in many different ways: taxonomy, ontology, classification, characterization, etc. For this reason I included the synonyms `classif*` and `character*`, alongside `taxonom*` in the initial search. I opted to exclude the term `ontology` from the search as it returned an excessive number of unrelated results (e.g., bots using ontologies in their decision making processes) that would have been infeasible to handle.

4.3.1.2 Select Sources

Five prominent digital libraries were selected for the systematic literature search: ACM Digital Library,⁸⁸ IEEE Xplore,⁸⁹ Science Direct,⁹⁰ SpringerLink,⁹¹ and Wiley Online.⁹² These databases were selected since they are widely used, well established in the domain of software engineering and computer science, and cover a broad range of article types (e.g., journals, proceedings, magazines, books, courses, or reference work).

4.3.1.3 Generate Search Strings

Search strings were generated for each of the five databases selected. The two search terms were linked with their synonyms (OR) and then joined together (AND) to generate the search strings. The search strings were applied only to the titles of the papers (due to the volume of papers), and no article publication date restrictions were used. A copy of the search strings used for each of the databases is provided in Appendix A.

4.3.1.4 Query Sources

Each of the five databases were queried in early April 2018, returning a total of 1,462 articles. At this point, Google Scholar,⁹³ a popular meta-search engine, was used to check the thoroughness of the search results and retrieve any articles not included the databases. I repeated the same queries and examined the first two pages of results for each term 1 (agent*, chatbot*, chatterbot*, bot*) and term 2 (classif*, character*, taxonom*) pairing. These queries returned one grey paper that was not included in our previous search results [61]. An overview of the distribution of articles returned from each of the digital libraries is shown in Table 4.1.

Table 4.1: Breakdown of search results by database.

Database	Search Results
ACM Digital Library	567
IEEE Xplore	198
ScienceDirect	39
SpringerLink	91
Wiley Online	577
Total	1,462 (databases) + 1 (meta-search) = 1,463

4.3.1.5 Article Selection

Querying the databases returned a total of 1,463 articles that satisfied the search strings. Due to the number of articles returned and the fact that many of them were unrelated to the software bots or did not focus on their observable properties, I generated a set of selection criteria to determine which articles should be used to create the taxonomy.

Selection Criteria: The selection criteria took two forms: inclusion and exclusion criteria. Articles had to satisfy the following **inclusion criteria** to be included:

- I1:** The article’s title contains the keywords defined in the search string: (agent or bot or chatbot) and (taxonomy or classification or characterization);
- I2:** the article’s main subject matter is software bots or bot subtypes, evidenced by being mentioned in title or abstract of the publication;
- I3:** the article examines or provides some classification, characterization, or taxonomy of the properties, dimensions, or functionality of bots;
- I4:** the article falls within SWEBOK (IEEE) Computing Foundations: Basic Concept of a System: Emergent Properties (13.8.1) [2];

- I5:** the article is published in a peer-reviewed journal, conference proceeding, workshop, book, magazine, or as grey literature (e.g., technical reports, graduate thesis); and
- I6:** the article is written in English.

Inclusion criteria 1 & 2 checked that the database queries returned the correct articles based on our search terms. Criteria 3 & 4 ensured that the articles that fit our knowledge area and subject matter, as described in Section 4.2. Lastly, criteria 5 & 6 ensured that high-quality articles were selected.

The **exclusion criteria** determined if the articles contained anything that should disqualify them from being used. The following exclusion criteria helped determine which articles should not be included:

- E1:** The article only focuses on a high-level role or behaviour-based classification of bots (e.g., the article doesn't examine any lower-level properties of bots);
- E2:** the article only focuses on the algorithmic classification of bots;
- E3:** the article only focuses on the application of bots in other domains (e.g., biology, chemistry, or medicine);
- E4:** the article only focuses on the classification of a very specific subtype of bots, which could not be applied to the domain of bots as a whole (e.g., botnets); or
- E5:** the full article was not available online (e.g., only the abstract or slides were available).

Exclusion criteria 1 ensured that the author's classification scheme was not too broad/high-level. Criteria 2, 3, & 4 checked that the articles were not too specific to be applied to the entire field of software bot research. Lastly, criteria 5 ensured that the article could be accessed.

Since articles that failed to fulfill all of the inclusion criteria or that satisfied any of the exclusion criteria were not included in the term extraction phase, each of the articles was put through a primary and secondary selection process.

Preliminary Selection (Title and Abstract): In the preliminary selection phase, the titles and abstracts (when available) for each of the articles were carefully read. Papers that failed to meet all of the selection criteria (i.e., they violated an inclusion criterion or satisfied an exclusion criterion) were removed immediately. If it was unclear whether the article would satisfy the selection criteria based on the title and abstract alone, I erred on the side of caution and tentatively included the article in the secondary selection phase. Duplicate papers were also removed, leaving a total of 60 potential articles to be analyzed in the secondary selection phase.

Secondary Selection (Full Paper): In the secondary selection phase, each of the remaining articles was read in full to determine if they satisfied all of the selection criteria and to extract the following meta-data:

- **Title:** The full title of the article.
- **Author(s):** The article author’s full names.
- **Publication Venue:** The type of venue that the article was published.
- **Publication Date:** The year article was published.
- **Exclusion Reason:** The selection criteria the paper failed to meet, if any.

45 articles failed to satisfy at least one of the selection criteria and were excluded. A list of these articles, including their extracted meta-data and reason for exclusion, is provided in Appendix B.1.

15 articles fulfilled all of the selection criteria and were included in the next phase of the taxonomy generation. For each of these articles, the following additional meta-data was extracted:

- **Subject Matter:** The type subject matter (e.g., bots, agents, agent subtype).
- **Classification:** The classification structure (e.g., taxonomy).
- **Methodology:** The methodology used to derive the classification.
- **Validation:** The classification validation method, if any.

A list of the articles that satisfied all of the selection criteria, including their extracted meta-data, is provided in Appendix B.2.

4.3.2 Backwards Snowballing

A round of backwards snowballing was also conducted to identify articles that were not returned in the database queries but still satisfied the remaining selection criteria. Snowballing helped identify papers that still focused on classifying the properties of software bots, but may have used slightly different terminology. Therefore, the snowballed articles still had to pass all of the selection criteria except for inclusion criteria I3 (i.e., keywords in the title).

I performed a single iteration of backwards snowballing using the methodology outlined by Wohlin [62]. The “*start set*” for the snowballing procedure was the 15 articles collected in systematic literature search phase, see Appendix B.2. While the list of articles cited by the start set (i.e., backwards snowballing) was manageable (100s), the list of papers citing the start set (i.e., forwards snowballing) was massive (1000s). Furthermore, examining a subset of the forward snowballing papers showed that most of the papers were focused on designing software bots, not classifying or understanding them. The papers that the

start set of articles referenced were often used to build upon or inspire the classifications, making them much more relevant. For these reasons, I opted to conduct only backwards snowballing.

I examined the title of each article in the reference lists and the place that the article was referenced in the paper. If it appeared as though it may pass the selection criteria, I read the full paper to determine its eligibility. I identified nine additional articles using this methodology and extracted the following meta-data from each: **Title Author(s) Publication venue Publication date Subject matter Classification Methodology Validation** A list of these articles and their extracted meta-data is provided in Appendix B.3.

4.3.3 Online Search

Since research into modern software bot technologies is still relatively new, the majority of the articles collected from the systematic literature search and backwards snowballing phases focused on software agents. For this reason, I conducted an informal online search for recent work (e.g., papers, blog posts, books) on software bots to get a fresh perspective on the latest trends. I selected high-quality articles that focused on the observable properties and behaviours of software bots. For this phase, I did not employ a rigorous methodology but instead relied on my personal expertise to identify 16 articles to be included in the next phase of the taxonomy generation. The following meta data was extracted from each of the articles: **Title Author(s) Publication venue Publication date Subject matter** A list of these articles, their meta-data, and reason for inclusion is provided in Appendix B.4.

4.4 Identification & Extraction (Phase 3)

The identification and extraction phase involved extracting and controlling the terminology that would be used to generate the new taxonomy. This phase had two main stages, **term identification and extraction** (Section 4.4.1) and **term reduction** (Section 4.4.2), which are described in the following sections.

4.4.1 Term Identification & Extraction

Terms that described an observable property or behaviour of software bots were extracted from the articles identified in the data collection phase, see Section 4.3. Since the terms that I extracted from the articles had to fall within the taxonomy’s knowledge area, objectives, and subject matter of the software bot taxonomy (see Section 4.2), I derived a set of extraction requirements. The terms needed to be included as part of the article author’s classification of bots or clearly stated as being a property that the authors associated with bots (e.g., not simply in the background or related work sections). The terms extracted

also had to describe the observable properties or behaviours of software bots and not purely an implementation detail. To ensure these requirements were satisfied by the terms to be extracted, I identified the following term **extraction criteria**:

R1 The author(s) clearly defines the term as a property of bots.

R2 The term describes an observable property or behaviour of bots.

Each of the articles was re-read and any terms that satisfied these requirements were extracted. If the author provided a simple definition for the term in the article, that was extracted in addition to the term itself. This resulted in a list of extracted terms for each of the articles identified in the data collection phase. All of the extracted terms were collected in an anonymized spreadsheet containing a list of paper IDs, extracted terms, and definitions (when available). Approximately 400 individual *articleId-term-definition* tuples were extracted during this stage.

It should be noted that, during this step, the main goal was to extract terms for generating the new taxonomy rather than evaluating the taxonomies presented in the articles. For this reason, I attempted to ignore the structure of the taxonomies while still extracting meaningful terms. This was done in an attempt to reduce biasing the structure of the new taxonomy.

4.4.2 Term Reduction

Since taxonomies are useful for organizing knowledge areas that lack common terminology, it is expected that many terms may be used to describe the same property, or that some terms may have multiple meanings [1]. For this reason, it is critical to reduce redundancies and remove inconsistencies when generating a new taxonomy.

I used two strategies to reduce redundancy throughout the taxonomy generation process: merging and mapping. *Merging* taxonomies is the process of bringing together different taxonomies on the same subject.⁹⁴ If multiple terms are used to describe the same concept, redundancies are merged together under a *preferred term* [63]. The result of merging taxonomies is a new, improved taxonomy that is a combination of the concepts from all of the taxonomies. *Mapping* involves identifying the alternate terms, often called *variant terms* [63], used to describe identical concepts across different taxonomies.⁹⁴

Digital Reduction: Once the initial term extraction was complete, I checked the spreadsheet for exact terminology matches and *merged* them together. It should be noted that this step did not remove any terms, but instead grouped together exact matches. After this initial digital reduction, approximately 350 unique terms remained.

Sorting Reduction: Term reduction continued throughout the remainder of the taxonomy generation process. Any term *merges* that were missed in the digital reduction step, for example “[64] *age*” and “[39] *age*”, were stapled together. Potential terminology *mappings* and/or *mergings*, for example “[65] *Learning (adaptive): changes its behavior based on its previous experience*”, “[66] *Amenability: ability to adapt behavior to optimize performance*”, and “[44] *Flexible*”, were tentatively clipped together and revisited later in the taxonomy generation process.

Inconsistency Removal: If I was unsure if a term satisfied the extraction criteria during the identification step, I erred on the side of caution and over-extracted terms. I did this since the terms would be revisited in the term reduction step, as well as throughout the remainder of the taxonomy generation process. Thus, if I found any cards that did not satisfy the term extraction requirements or the goals of the taxonomy, they were removed from the taxonomy generation process. A full list of excluded cards and their reasons for exclusion is provided in Appendix C.

4.5 Design & Construction (Phase 4)

This section describes how the taxonomy’s top-level dimensions, facets, sub-facets, and facet values were selected in the design and construction phase. As shown in Figure 4.5, I employed a highly iterative process of taxonomy generation which included six main steps: card sorting (Section 4.5.1), identifying top-level dimensions (Section 4.5.2), identifying facets/sub-facets (Section 4.5.2), describing relationships (Section 4.5.2), refining dimensions (Section 4.5.3), and validation (Chapter 6).

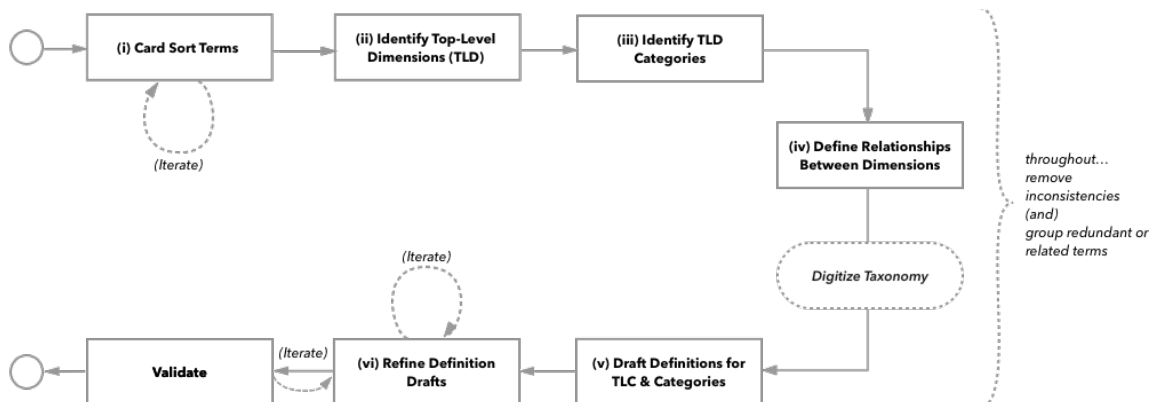


Figure 4.5: The taxonomy construction methodology followed.

4.5.1 Card Sorting

Card sorting is a fast, inexpensive, and reliable method to derive an initial foundation for structuring vast amounts of data [67, 68]. For these reasons, I employed card sorting to identify patterns and create a rough taxonomy structure from the extracted terms. The two common card sorting methodologies are open sorting, where subjects can define their own top-level categories, and closed sorting, where top-level categories are already defined [67, 68]. Since I had few insights on how the taxonomy should be structured, I used open card sorting to help reveal the categories that would best capture the range of properties and behaviours of software bots. Below, I briefly describe the card sorting process that was followed.

Content: Good content for the card sort should come from a variety of sources [67]. The content used for the sort were the terms extracted (cf. Section 4.4) from the articles collected during the systematic literature search, results snowballing, and online search (cf. Section 4.3). The extracted terms were printed on 11cm by 2.5cm cards, as shown in Figure 4.6. Each of the cards contained a single term, its definition (if available), and the article ID(s).

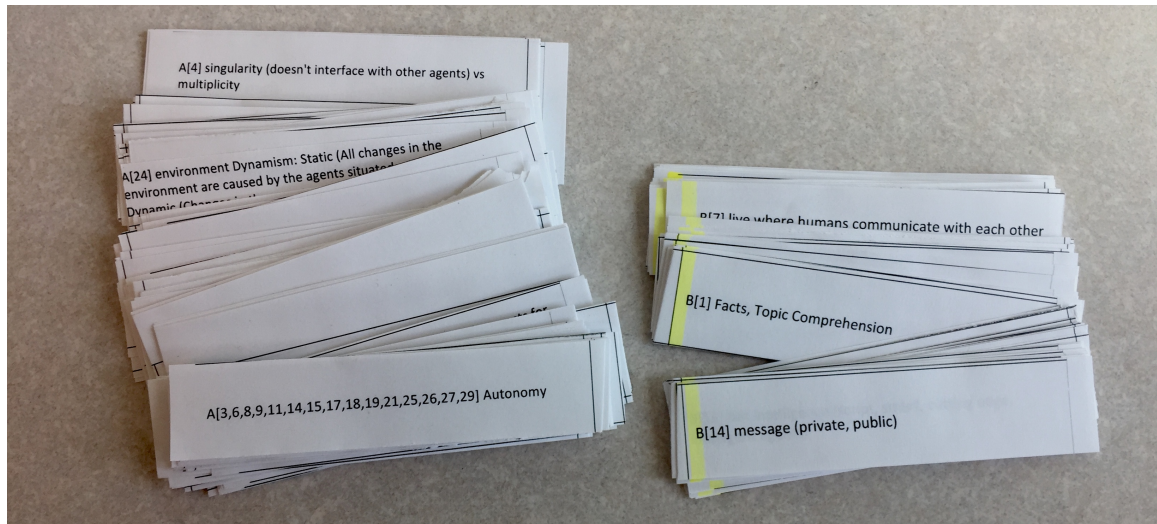


Figure 4.6: The content for the card sorting process.

Number of Cards: A total of 338 cards were sorted. While we were well over Spencer and Warfel [68]’s recommended maximum number of cards (30-200), these guidelines are only a recommendation since it can be time consuming and tiresome for the participants [67]. Since our card sorting was completed over multiple sessions and we were familiar with the subject matter, we chose to take on a larger sort.

Duration: The card sorting process was not limited by time, but rather until consensus was reached. The preliminary card sorting was conducted in a series of short sessions which spanned over a consecutive two-week period. Each of the sessions ranged from one to four hours in length.

Participants: The card sorting was performed by myself and a collaborator. Although it is generally recommended to have more than two participants [67], due to the number of cards and the time requirements for the sort, it was not feasible to perform the sort with more than two participants. Although I believe that the thoroughness of the card sort, completing multiple iterations of the taxonomy, and validating it makes up for only having two participants. All of the sorting was completed with both of us present.

Prior to the card sorting process, my collaborator was familiar with bots but had not studied them directly. They were not involved in the article selection or extraction processes, had not read the articles from which the terms were extracted, and had no prior experience with software bot classifications.

Card Sorting Process: Prior to beginning the card sort, all of the term extraction cards were shuffled and stacked in a pile on the table (Figure 4.7a). My collaborator and I examined each of the cards and worked together to group similar terms (Figure 4.7b).

The card sorting process was iterative. Each of the groupings were revisited multiple times and cards were re-assigned to different groups as required. Any terms that were found to be duplicates, redundancies, or synonyms were reduced using the methodology described in Section 4.4.2. If we were unsure of the meaning of any terms, I referred to the original article and updated the card as required.

We continued the sort until the structure of the groups remained relatively stable upon re-examination (Figure 4.7c). The cards were laid out according to their groupings and we identified the rough dimensions, facets, and the relationships between the dimensions with sticky notes (Figure 4.7d).

4.5.2 Dimensions, Facets, and Relationships

Faceted taxonomies have a set of dimensions at the top level that represent the main categories under which the bots can be classified [1]. The initial version of the taxonomy's dimensions emerged during the card sorting process, as described above.

Once the initial top-level dimensions were identified, the cards corresponding to each dimension were re-examined to determine the appropriate facets, sub-facets, and facet values. When facets began to emerge from the terms in the dimension card grouping, we identified them as possible facets and gave them a label.

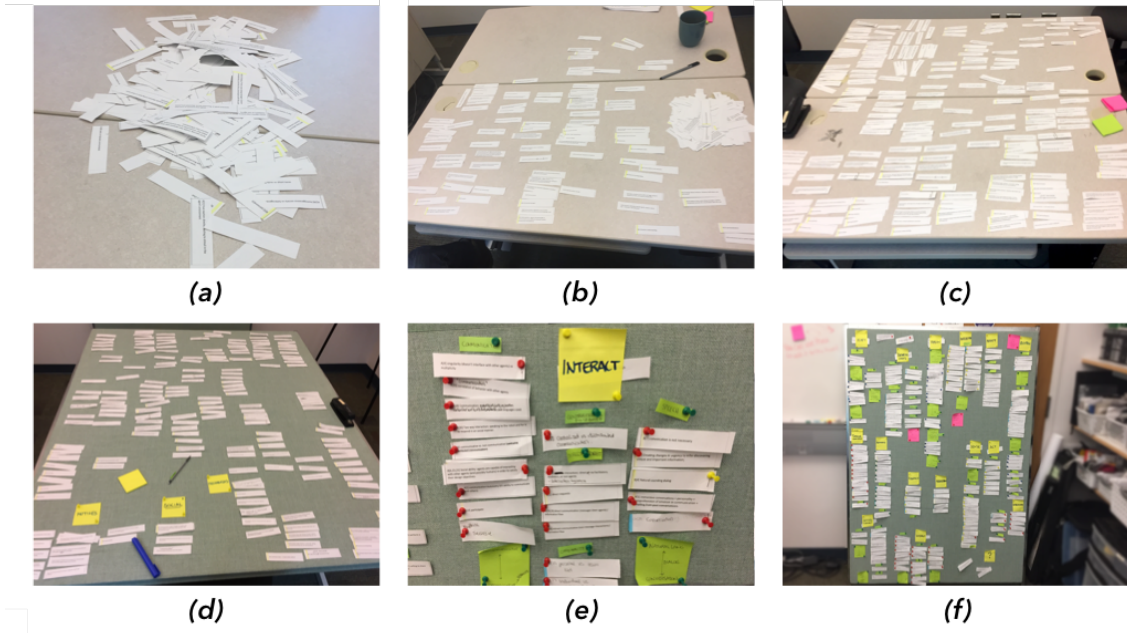


Figure 4.7: The software bot taxonomy at various stages of creation: (a) shuffled cards ready to be sorted; (b) beginning to group similar terms; (c) groups have been created; (d) assigning the groups dimension labels; (e) labeling the dimensions, facets, and sub-facets; (f) the completed initial version of the taxonomy.

Since the taxonomy focused on the emergent properties of software bots, it is likely that all of the properties had some form of relationship between each other. Thus, for the scope of the taxonomy, we focused only on the most prominent relationships. When we identified key relationships between facets during the card sorting process, we recorded them on sticky notes and pinned them next to the dimensions.

4.5.3 Drafting Definitions & Refining Dimensions

A digital version of the card sorted taxonomy was created using the FreeMind⁹⁵ mind mapping software. Using this tool, I drafted the initial set of definitions for each of the dimensions, facets, and sub-facets based on the definitions extracted from the original articles. The relationships between dimensions were captured with arrows.

A high-level copy of the initial version of the taxonomy is shown in Figure 4.8. The first version of the taxonomy had 13 top-level dimensions and over 40 sub-facets. This created an extremely broad but shallow faceted taxonomy, which would make it difficult for users to leverage portions of. For this reason, I decided to reduce the number of top-level dimensions and create a slightly narrower and deeper taxonomy structure. I tested a variety of arrangements for the top-level dimensions, facets, and sub-facets, as shown in Appendix D. I settled on three top-level dimensions (environment, intrinsic, and interaction)

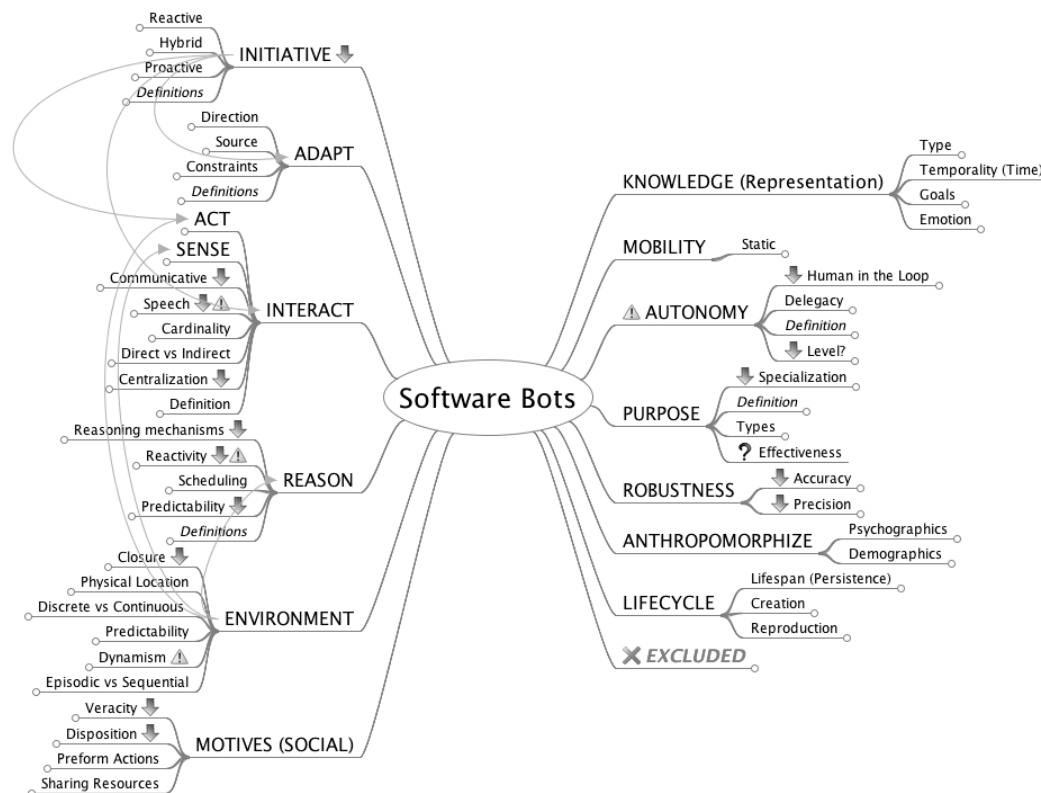


Figure 4.8: A high level overview of the preliminary version of the Software Bot Taxonomy using FreeMind⁹⁵.

and the remainder of the facets/sub-facets were placed under the appropriate dimension. This was an extremely iterative process, and multiple incremental versions of the taxonomy were created.

Facet Values: At this point, I also identified the possible classification values for each of the facets/sub-facets. When describing these values, I looked back at the articles that the terms were extracted from. When the values were not adequately defined in the literature, I used my knowledge of the subject matter to help identify related work that could expand the definitions.

Revisiting Merged & Mapped Terms: Once the taxonomy's structure was relatively stable, I re-read each of the articles that were used to generate the taxonomy to ensure that the terms had been correctly *merged* and *mapped*. I corrected any errors, added any missed references, and ensured that no term mappings had been lost in the process. If I was unsure if two terms were exact mappings (i.e., proper variant terms), I erred on the side of caution and did not map them together.

A summary of the term mappings that I identified when building the proposed taxonomy is provided in Appendix E.

Collaborator Consensus: My collaborator reviewed the changes that I made to the overall structure of the taxonomy, as well as the definitions I drafted for each dimension, facet, sub-facet, and facet value. This helped ensure that the changes I made were in line with the initial card stored version of the taxonomy.

In the next Chapter, I present the finalized version of the proposed taxonomy of software bots, and provide a guide to help readers interpret and use the taxonomy.

4.6 Summary

In this chapter, I described the iterative methodology used to generate my updated taxonomy of the observable properties and behaviours of modern software bots. I adapted Usman et al.'s software engineering taxonomy generation methodology to ensure rigor and allow to a multi-stage data collection, term extraction, taxonomy construction, and validation process. I collected articles that discussed the characteristics of software bots from three data sources (i.e., systematic literature search, literature search snowballing, and online search), and extracted any terms used to describe the observable properties and behaviours of software bots. The extracted terms were then reduced (through mapping and merging variant terms), and card sorted to allow the new taxonomy's dimensions, facets, and facet values to naturally emerge from the data.

In the next chapter, I present the resulting holistic taxonomy of software bots and provide a guide to help users interpret the taxonomy. I describe each of the three top-level dimensions (environment, intrinsic, and interaction), the facets/sub-facets that fall under each of the dimensions, and the range of values for each facet. I also discuss the taxonomy's validation process (cf. Chapter 6) and explore the implications of the proposed taxonomy (cf. Chapter 7).

⁸⁷<http://www.getty.edu/research/tools/vocabularies/aat/>

⁸⁸<http://dl.acm.org.ezproxy.library.uvic.ca/dl.cfm>

⁸⁹<http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/Xplore/home.jsp>

⁹⁰<http://www.sciencedirect.com.ezproxy.library.uvic.ca/science/search>

⁹¹<https://link-springer-com.ezproxy.library.uvic.ca/>

⁹²<http://onlinelibrary.wiley.com.ezproxy.library.uvic.ca/advanced/search>

⁹³<https://scholar.google.ca/>

⁹⁴<https://www.slideshare.net/HeatherHedden/mapping-merging-multilingualtaxonomies>

⁹⁵<http://freemind.sourceforge.net/>

Chapter 5

A Taxonomy of Software Bot

In this chapter, I introduce a taxonomy of the observable properties and behaviours of software bots, which I refer to as “*bots*” for brevity. At the top level, the taxonomy has three dimensions. These dimensions describe the first level of the taxonomy and provide an overview of the taxonomy at a broader level:

- (a) The bot’s **environment**. These facets describe the environment(s) that the bot operates in.
- (b) The **intrinsic** properties of bot itself. These are the facets that the bot designer has complete control over when building the bot.
- (c) The bot’s **interactions** within its environment. These facets describe the ways in which bots interact with their environment and those operating within it.

Figure 5.1 provides an overview of the taxonomy’s dimensions and the top-level facets. Each of the taxonomy’s dimensions, facets/sub-facets, and facet values will be discussed in greater detail later in this chapter. Before presenting the taxonomy, I first provide a general guide to help readers and users interpret the proposed taxonomy and understand its basic usage.

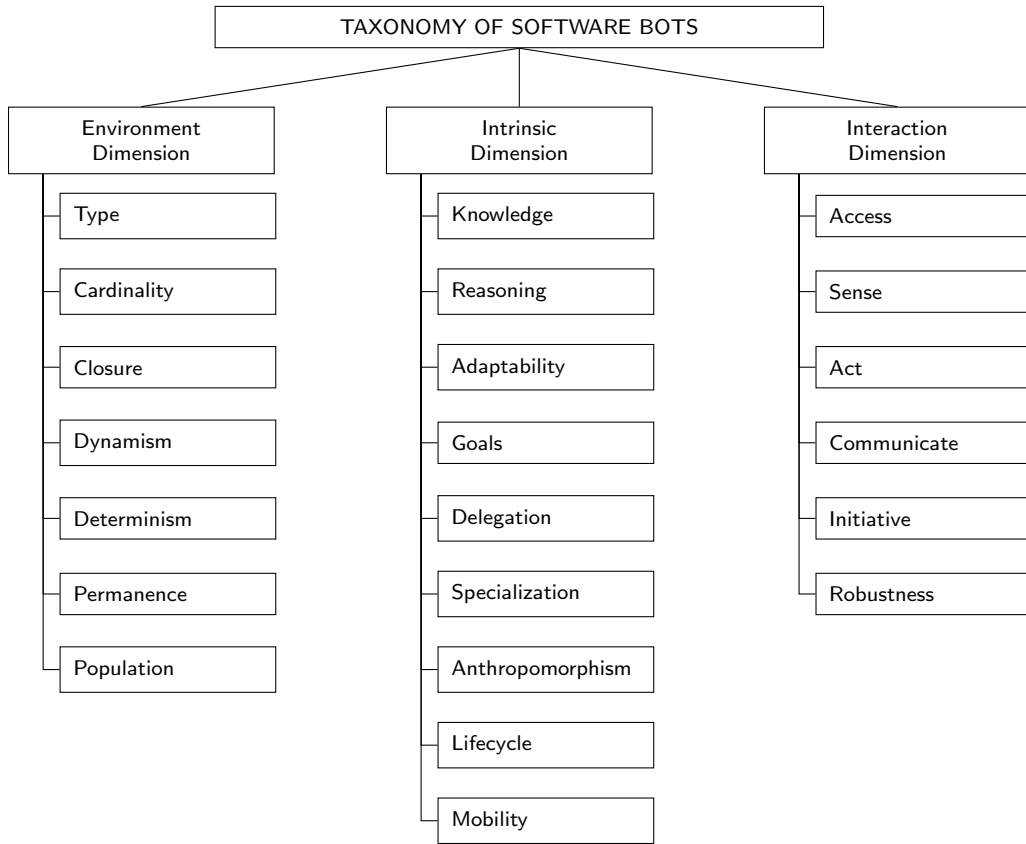


Figure 5.1: A high-level view of the Software Bot Taxonomy’s structure.

5.1 Reader’s Guide

This taxonomy presents a *faceted classification* of the emergent properties and behaviours of software bots to allow for flexible classification in a consistently changing field. Faceted taxonomies allow for the subject matter (in this case software bots) to be classified from multiple, independent perspectives called facets. Each of these facets are combined to create a full classification of a software bot.

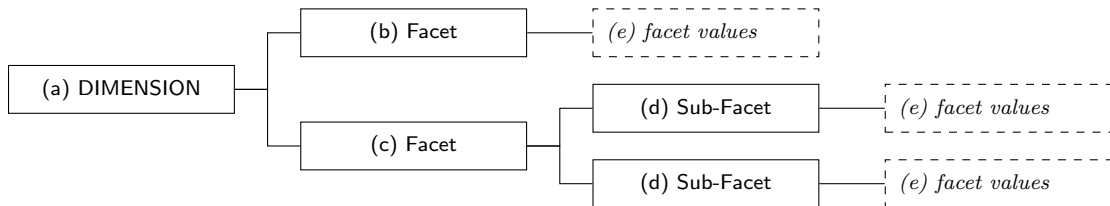


Figure 5.2: Example of the structure of the proposed taxonomy.

Dimensions: The facets describing the observable properties and/or behaviours of software bots are organized under three high-level dimensions: the properties describing the **environment** in which the bot is situated, the **intrinsic** properties of the bot, and the behaviours of the bot as it **interacts** within its environment. Each of these three high-level dimensions is decomposed into facets, sub-facets, and facet values, as shown in Figure 5.2

Facets and Sub-Facets: Facets (Figure 5.2b/c) describe the various properties that software bots can exude for the corresponding dimension (Figure 5.2a). Facets can either be decomposed further into sub-facets (Figure 5.2d) or they can have a set of possible values (Figure 5.2e).

Facet Values: Each lowest level facet or sub-facet (Figure 5.2b/d) has set of values for classifying the bot for the property or behaviour (Figure 5.2e). A bot should be able to be classified with the facet values. The facet values can be one of three types: independent **boolean sub-facets**, mutually **exclusive states**, or **ranges**.

There are 3 boolean “example” sub-facets:	There are 3 exclusive “example” states:	Possible “example” values range from “A” to “C”:
$\oplus$ Value A This value is a sub-facet, and is rated true or false independently. $\oplus$ Value B This value is a sub-facet, and is rated true or false independently. $\oplus$ Value C This value is a sub-facet, and is rated true or false independently.	Value A If this value is true, all other values are false. Value B If this value is true, all other values are false. Value C If this value is true, all other values are false.	Value A This is the min value in the range. Value B This is the middle value in the range. Value C This is the max value in the range.
(a)	(b)	(c)

Figure 5.3: Examples of the three possible facet value types: (a) boolean sub-facets, (b) exclusive states, and (c) ranges.

If the values are **boolean sub-facets**, then they are actually condensed sub-facets with two possible values, either true or false. The bot should be classified on each boolean sub-facet independently. Boolean values are represented with an $\oplus$ icon beside facet values, as shown in Figure 5.3(a).

If the values are mutually **exclusive states**, then the bot should fall under exactly one of the values, as shown in Figure 5.3(b). These values are either nominal or ordinal with discrete states.

If the values are a **range**, then the bot should be positioned somewhere along the range of values, as shown in Figure 5.3(c). These values are ordinal and the bot can lie anywhere along the range of values represented with an arrow icon ($\downarrow$).

Relationships: Since the taxonomy is faceted, relationships can exist between facets and/or sub-facets. Although many of the facets are independent and have no meaningful relationships between them, many of the facets are related to each other (i.e., the values

on some facets can influence or restrict the values on other facets). In the following taxonomy, we identify some key relationships between the many of the facets and/or sub-facets. However, it is likely that many other interrelationships exist.

5.1.1 General Usage Guidelines

This taxonomy, first and foremost, presents an attempt to update and organize the emergent properties of software bots. More specifically, this taxonomy presents a *controlled vocabulary* (i.e., preferred terms [63]) for discussing the *observable properties and behaviours of software bots*. Secondly, this taxonomy provides a *range of possible values* for each category of properties or behaviours. Lastly, the taxonomy also provides a set of *alternative terminology* (i.e., variant terms [63]) that map between the controlled vocabulary and terms used to describe the same content in previous research. Below, I discuss some general considerations when using the software bot taxonomy.

Non-Prescriptive: This taxonomy attempts to present an unbiased look at the variety of observable properties and behaviours of software bots. The goal of this taxonomy is to provide a deeper understanding of existing software bots as a whole. Therefore, this taxonomy does not provide guidelines nor recommendations for selecting between the range of possible values for a property or behaviour of bots.

Due to the taxonomy’s non-prescriptive approach, it should be noted that this taxonomy also **does not** focus on **privacy**, **security**, or **ethics** related considerations. It is the taxonomy’s user’s responsibility to carefully consider these factors when interpreting and using this taxonomy.

Flexible Usage: Since the taxonomy supports faceted analysis, users can utilize as many or as few facets as they require for their given task. Furthermore, the facets/sub-facets are organized under three high-level dimensions, so users can use the facets from one dimension or a combination of dimensions. Users also do not need to know the name of the categories for which the bot is being classified prior to using the taxonomy, since a bot should be able to be classified by all the dimensions.

Expanding the Taxonomy: Although this taxonomy presents an initial effort towards classifying the observable properties and behaviours of software bots, it should not be considered complete. The taxonomy should be continuously re-evaluated to accommodate shifting research goals, new developments, and emerging ideas in software bots. In the taxonomy itself, some opportunities for future expansions are highlighted.

Since the taxonomy is multi-faceted, it allows for graceful expansion as each facet is independent. This allows additional facets/sub-facets, facet values, and/or term mappings

to be added where required. Before a new facet is added, the user should carefully consider if it fits within the knowledge area (i.e., emergent systems) and is classifying the correct subject matter (observable properties/behaviours) of the software bot taxonomy, as shown in Section 4.2. They should also ensure that the new content does not repeat information presented elsewhere in the taxonomy.

Users should also ensure that they position the new content under the correct dimension, facet, and/or sub-facet. If adding a new facet/sub-facet, the new facet’s values as well as the value type (boolean sub-facets, mutually exclusive states, or a range) should be identified. Any relationships between the new facets and additional term mappings should also be defined. Additional values, term mappings, or relationships can also be added to an existing facet in a similar manner. Ideally, the validity of the expansions should also be tested (e.g., tagging examples of bots, expert opinion, benchmarking, etc. [1]).

5.2 Environment Dimensions

To better understand the bot, we have to first understand its environment. The environment dimension describes the surrounding in which the bot lives and operates. What we can observe about bots is how they behave with the environment around them, therefore the environment likely has an influence on the bot’s behaviours. As shown in Figure 5.4, there are seven top-level facets that fall under the environment dimension: **type**, **scope**, **closure**, **dynamism**, **predictability**, **permanence**, and **population**.

A bot might operate in many distinct environments. In this case, each of the bot’s environments should be classified independently to provide a more complete picture of the influences that each of the environments may have on the bot.

5.2.1 Environment Type

The bot environment’s type describes the setting (often times a system) which the bot inhabits, participates, or accesses. Bots may differ on the type of environment in which they execute (i.e., intrinsic dimensions) and where they interact (i.e., interaction dimensions).

There are two **exclusive** environment type states:

- Standalone** The bot is not tied or restricted to a specific platform [44]. The bot is hosted independently, but can access platforms much in the same way other non-bot users would.
- Platform** The bot is “*integrated*” [61] into the platform. The bot can either be hosted independently or through the platform, but it accesses the platform through non-user methods (e.g., through APIs). Platform bots are often seen as “*aug-*

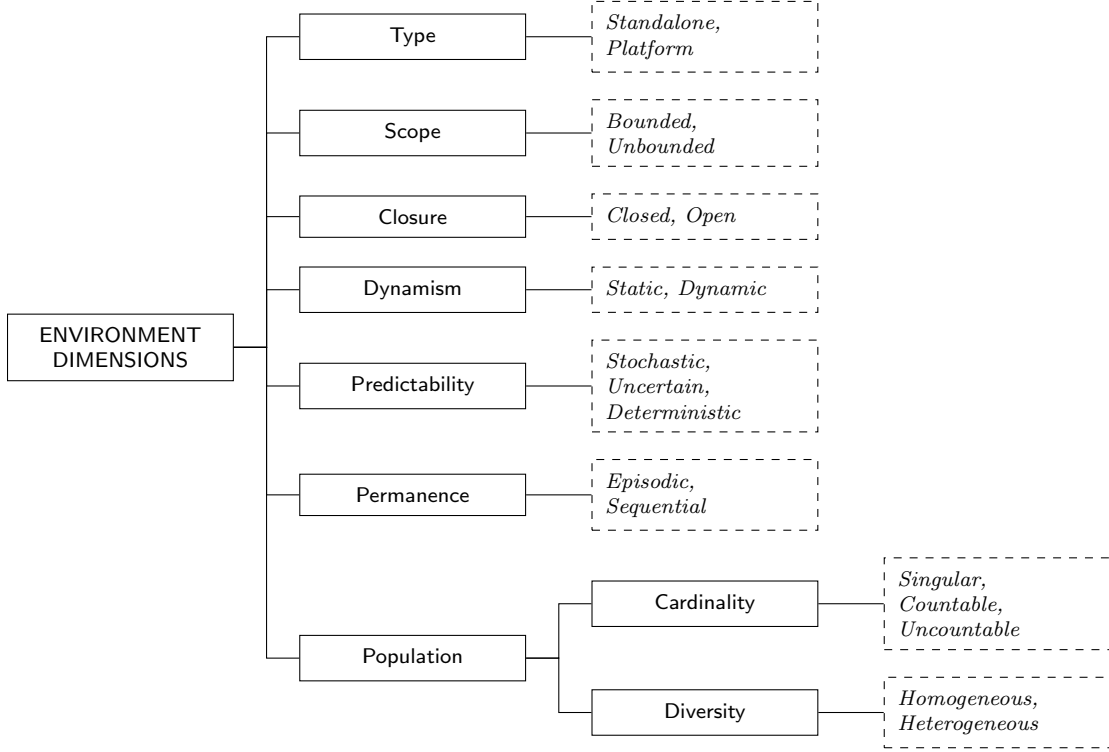


Figure 5.4: The Software Bot Taxonomy’s Environment Dimensions. The dimensions/facets/sub-facets and the range of possible values which the facet can take on are shown in the solid and dashed boxes, respectively.

menting” a system’s behaviour [61]. Some common examples of platform types include but are not limited to:

- (i) **Social platforms:** Bots operate in online human communities (e.g., Facebook Messenger, Skype, Slack, etc.).
- (iii) **Ambient platforms:** Bots operate in the real world and users call upon them with voice commands (e.g., Alexa, Siri, and other “consumer voice” [39] bots).
- (ii) **Computer systems:** Bots operate within computer systems (e.g., networks [69], operating systems [69], databases [69], etc.).

5.2.2 Scope

The bot’s scope describes the size of its environment [66]. More specifically, the scope describes whether the environment is bounded or not.

There are two **exclusive** scope states:

- Bounded** The environment is limited with respect to its size [43]. If the bot's environment is bounded, then the bot is limited in how far it can (hypothetically) travel.
- Unbounded** The environment is not limited with respect to its size [43]. If the bot's environment is continuous, then the bot is not limited in how far it can (hypothetically) travel.

5.2.3 Closure

The bot environment's closure describes who is able to access the environment [66]. The closure determines whether or not outsiders (i.e., those not already inhabiting the bot's environment) are able to access the environment.

There are two **exclusive** closure states:

- Closed** Access to the environment is limited by constraints [66, 70]. A closed environment imposes restrictions on who can access it (e.g., requiring user accounts). Facebook Messenger, for example, would be a closed environment since you need an account to access it.
- Open** If the environment is not closed, then it is open. An open environment allows others to freely access it without imposing restrictions [70, 66, 43]. The internet is an example of an open environment.

5.2.4 Dynamism

The bot environment's dynamism is the degree to which the bot's environment is capable of being changed by outside forces [66, 43].

There are two **exclusive** dynamism states:

- Static** All of the changes in the environment can be attributed to the bot's actions [66, 43]. This means that the bot's environment cannot change while the bot is deliberating [5].
- Dynamic** If the environment is not static, then it is dynamic. A dynamic environment changes as a result of forces outside of the bot's control. The changes in a dynamic environment can stem from those inhabiting the environment, environmental factors (e.g., time), or randomness [71, 66, 43].

5.2.5 Predictability

The bot environment's predictability is the degree to which, given the same conditions, the outcome of the bot's actions can be predicted [43, 66].

Possible predictability values **range** from *deterministic* to *stochastic*:

↓	Stochastic	If the environment is not deterministic or uncertain, then it is stochastic. In a stochastic environment, the results of the actions performed are random and cannot be predicted [43].
	Uncertain	The results of the bot's actions can be partially (but not fully) predicted [5].
	Deterministic	The results of the actions that the bot performs in the environment can be fully predicted [43, 66, 5]. With respect to classifying the environment's determinism, we ignore the uncertainty that stems from the actions of others in the environment (e.g., bots, humans, other systems). Even in a deterministic environment, bots are not required to predict the actions of others for the environment to be deterministic.

5.2.6 Permanence

The bot environment's permanence describes how long the effect of changes, stemming from actions taken in the bot's environment, persist. Action permanence is defined by how long the change of *environmental state* lasts. The permanence of actions can differ based on who performed them (e.g., the bot, other bots, or humans), the type of action performed (e.g., complexity, criticality), etc.

There are two **boolean** permanence sub-facets:

- ⊕ **Episodic** The actions performed in the environment only temporarily affect the environment's state [43, 5]. After the current interaction (episode) with the environment is over, the environment reverts back to its previous state and the previously performed actions do not impact the outcome of future actions. When Mention-Bot, for example, posts a message on a new GitHub pull request, it does not have any effect on the future state of the environment itself.
- ⊕ **Sequential** The environment is sequential if the actions performed in the environment persist [43, 5]. The actions permanently change the environment's state and may impact the future actions performed within the environment. When Alexa turns on a light, for example, the light remains on until another action is taken to turn it off.

5.2.7 Population

The population describes the active entities situated within the bot’s environment. There is a distinction, however, between “*active*” and “*passive*” objects in the bot’s environment [43]. *Passive objects* can only be manipulated [43] and cannot change their own attributes, drives, or behaviours. The environment’s population describes the overall make-up of the *active objects* inhabiting, participating, or accessing the bot’s environment. These active objects can be other bots, systems, humans, etc. The environment’s population is described by two sub-facets: **cardinality** and **diversity**.

5.2.7.1 Cardinality

The population cardinality describes the size of the population in the bot’s environment. In other words, the cardinality describes the number of *active objects* in the bot’s environment.

Possible cardinality values **range** from *singular* to *uncountable*:

↓ Singular	The bot is the only member of the population.
↓ Countable	The population can be reasonably counted.
↓ Uncountable	The population cannot be reasonably counted.

5.2.7.2 Diversity

The population diversity describes the composition of the population [66]. In other words, population diversity describes the differences of the *active* objects in the bot’s environment. If the environment’s cardinality is not singular, the environment can also be described by its diversity.

Possible diversity values **range** from *homogeneous* to *heterogeneous*:

↓ Homogeneous	All members of the population are the same type [43, 72, 66] or behave in a similar manner. If the bot is the only member of the population, the environment is homogeneous.
↓ Heterogeneity	If the population is not homogeneous, it is heterogeneous. A heterogeneous population has a diverse set of inhabitants [43, 72, 66]. These inhabitants can be other bots, systems, users, etc. that live or operate in the bot’s environment [72]. The heterogeneity of the population can range from almost homogeneous to extremely diverse.

5.3 Intrinsic Dimensions

The intrinsic dimension is composed of facets that describe internal properties or properties belonging to the bot itself. There are a total of 7 facets and 28 sub-facets that fall under the intrinsic dimension, as shown in Figure 5.5.

The bot's developer has complete control over its intrinsic dimensions. Although some of these intrinsic facets touch on the inner workings of the software bot itself, they are still relatively visible from a black box approach. However, for the most part, I try to focus on the externally observable, intrinsic properties of software bots.

5.3.1 Knowledge

A bot's knowledge is what the bot knows or understands. They have the ability to store and use information to achieve their goals. Since knowledge is a very high-level concept, it is broken down into the following sub-facets: **memory** and **source**.

5.3.1.1 Memory

A bot's memory describes its ability to both store and access its knowledge. Hypothetically, a bot should have the ability to store and access any knowledge type it understands.

There are three **boolean** memory sub-facets:

- ⊕ **Long-term** The bot is able to store and access past events, actions, etc. More specifically, the bot remembers what happened before [73].
- ⊕ **Short-term** The bot is able to temporarily store and access the current context, events, actions, etc. Bots that have short-term memory may have the ability to understand where it is, when it is, who it's talking to, etc. [73].
- ⊕ **Future** The bot is able to store and access predictions of future events, actions, etc.

5.3.1.2 Source

The knowledge source describes where the bot's knowledge originates from [61].

There are three **boolean** knowledge source sub-facets:

- ⊕ **Encoded** The bot's knowledge is directly encoded (by programmer or creator) [61]. Encoded knowledge can only be provided before runtime.
- ⊕ **Supplied** The bot's knowledge is provided by someone or something in its environment [61]. Supplied knowledge can only be provided to the bot at runtime.
- ⊕ **Learned** The bot's knowledge is inferred from its environment [61]. Learned knowledge can only be inferred at runtime.

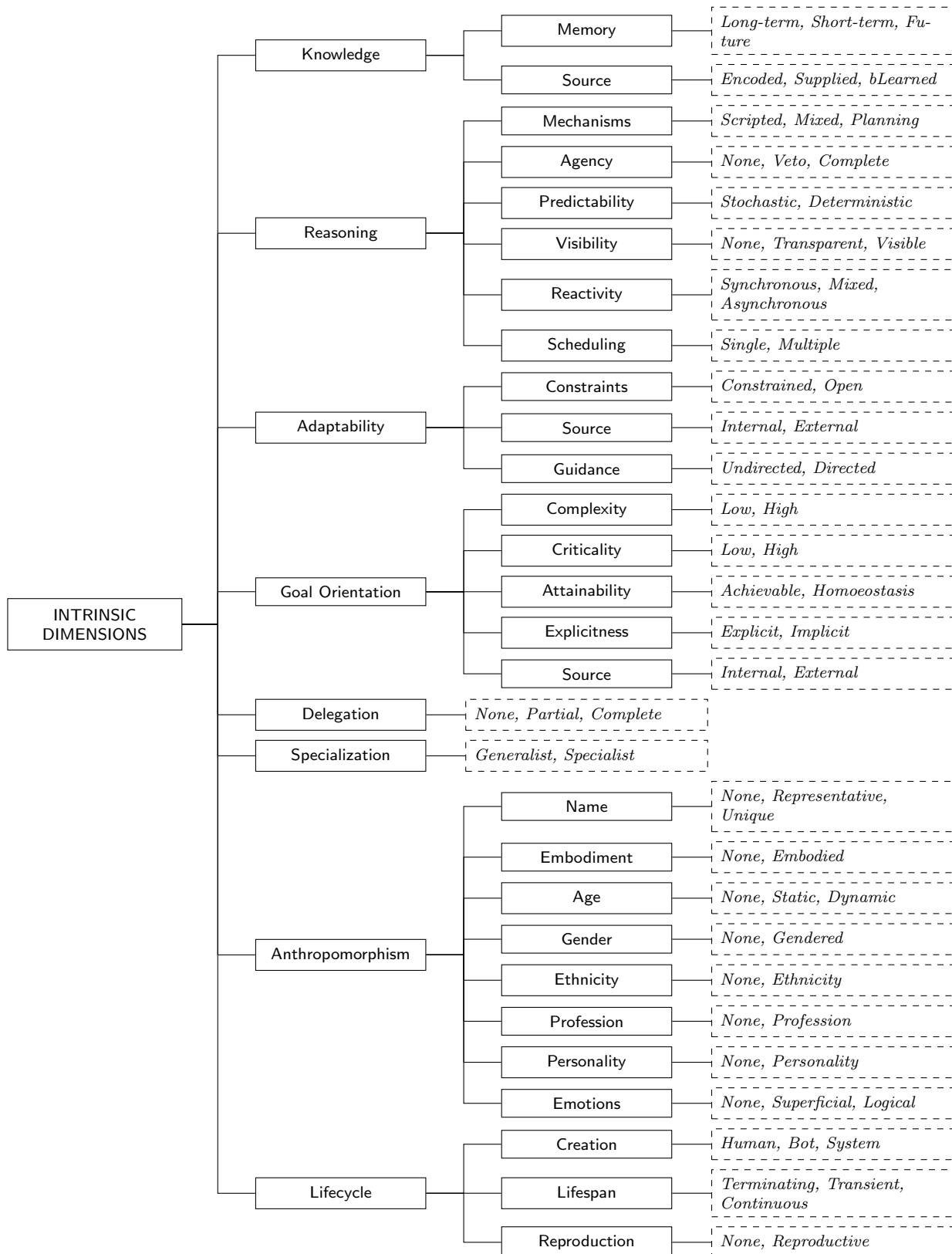


Figure 5.5: The software bot taxonomy's intrinsic dimensions. The dimensions/facets/sub-facets as well as the range of possible values which the facet can take on are shown in the solid and dashed boxes, respectively.

5.3.2 Reasoning

A bot's reasoning describes its capacity to apply logic to achieve its goals. Since reasoning is a very high-level concept, the reasoning dimension is broken down into six sub-facets: **mechanisms**, **agency**, **predictability**, **visibility**, **reactivity**, and **scheduling**. The bot's reasoning refers to its ability to reflect upon the meaning of its inputs (see *sensing* and *knowledge*) to generate outputs (see *actions*) as it attempts to realize its goals (see *goal orientation*).

5.3.2.1 Mechanisms

A bot's reasoning mechanism is the way that it processes inputs and/or generates outputs in order to realize its goals. Although technically a bot's reasoning mechanism falls within the *black box* of the system, we opted to include it in the taxonomy since the outputs of the reasoning mechanism can be observed. Therefore, the type of reasoning mechanism can usually be inferred based on the bot's stimulus (what the bot senses) and response (the actions or behaviours of the bot).

Possible reasoning mechanism values **range** from *scripted* to *planning*:

Scripted	The bot has scripted reasoning if it responds to a set of predefined stimuli with a corresponding set of preprogrammed responses. A scripted bot does not have to come up with its own responses to inputs since it already has a set of responses. However, this might limit the bot's ability to respond to unexpected inputs.
Mixed	The bot has mixed reasoning mechanisms if it uses a combination of planned and scripted reasoning mechanisms [65, 66]. In certain situations, the bot may react to inputs with scripted responses. At other times, the bot may plan its responses to the situation. These hybrid systems can take the form of script ' <i>overridden by plan</i> ', ' <i>modified by plan</i> ', etc. [65]
Planning	The bot has planned reasoning if it does not have a predefined script mapping inputs to outputs, but instead makes decisions based on the situation and its current <i>knowledge</i> .

5.3.2.2 Agency

A bot's agency describes its ability to perform the tasks it requires to achieve its goals without interference. The bot may be required to get clearance or permission from an *external party* (e.g., a human, bot, or another system) prior to performing any actions. The bot's degree of agency may change depending on the type of *goal*, the situation, or who it's interacting with.

Possible agency values **range** from *none* to *complete*:

None	The bot has no agency if it requires an external party to approve actions before the bot can perform them.
Veto	The bot has the ability to carry out the tasks required to realize its goals, however, an external party can veto the bot's actions. The external party can veto actions either before (preventing) or after (undoing) they have been performed.
Complete	The bot does not require permission to carry out the tasks required to realize its goals. When discussing bots, complete agency is often described as autonomy.

It should be noted that I adopt the philosophical definition of agency, “*the capacity of an entity (a person or other entity, human or any living being in general, or soul-consciousness in religion) to act in any given environment*” [74]. Although, I do acknowledge that agency is a loaded term and often has a different meaning in both the domains of software engineering and artificial intelligence.

5.3.2.3 Predictability

The bot's reasoning predictability is the degree to which the bot's outputs (e.g., actions or behaviours) can be predicted given the same conditions.

Possible predictability values **range** from *stochastic* to *deterministic*:

Stochastic	The bot is stochastic if the results of its reasoning mechanism appear as though they are random.
Mixed	The bot has a mixed predictability if it is stochastic for some input types and predictable for others, or is a combination for an input.
Deterministic	The bot is deterministic if the results of its reasoning mechanism are the same when provided with the same inputs and conditions.

5.3.2.4 Visibility

The bot's reasoning visibility is the degree to which it makes its decisions or actions visible to others.

A bot may vary its degree of visibility based on the type of *action* it's performing or its *current context*. The bot may also make actions more or less visible to different groups (e.g., types of users, other bots, systems, etc.)

Possible visibility values **range** from *None* to *Visible*:

None	The bot's visibility is none if all of its decisions or actions are hidden.
Transparent	The bot's visibility is transparent if its decisions or actions leave visible traces when the bot is not actively trying to make its processes visible.
Visible	The bot is visible if it actively works to make its decisions or actions visible. The bot creates additional artifacts for the sole purpose of providing visibility into its decisions or actions.

5.3.2.5 Reactivity

A bot's reasoning reactivity is the time the bot takes to respond to stimuli [66]. Reactivity is not purely the time to process its response, as some bots may implement an artificial response delay (e.g., to make it look like the bot is doing work or thinking). Reactivity describes both the time it takes the bot to select and perform on its response. However, the bot's response to the stimuli may be to do nothing.

A bot may vary the speed of its reactivity based on various factors such as the type of *action* it's performing, the *current context*, or what it's responding to (e.g., humans, other bots, something in the environment).

Possible reactivity values **range** from *synchronous* to *asynchronous*:

Synchronous	The bot responds at the same time (or very shortly after) as the stimuli is perceived (i.e., in a synchronous manner [75]).
Mixed	The bot uses a mixture of synchronous and asynchronous response times. The bot can range from mostly synchronous, to mostly asynchronous.
Asynchronous	The bot responds to the stimuli after some time has passed (i.e., in an asynchronous manner).

5.3.2.6 Scheduling

A bot's reasoning scheduling describes the bot's strategy for dealing with multiple inputs or outputs that need to be reasoned about. The bots scheduling also describes how many different reasoning processes the bot can handle simultaneously.

Scheduling restricts the bot's ability to process what it *senses* and produce meaningful responses in the form of *actions*. A bot may also display different scheduling behaviours based on the task (i.e., type of input/output).

There are two **exclusive** scheduling states:

Single Tasked The bot is single tasked if it can only handle one stimulus or task at a time. Single-tasked bots can be further broken down into:

- (i) **Non-Interrupting:** Since only one task can exist at a time, the new instance of a task is ignored [76].
- (ii) **Hybrid:** The bot is hybrid if it is interrupting and non-interrupting based on the situation or priority of the stimuli.
- (iii) **Interrupting:** Since only one task can exist at a time, the new instance of a task overrides the original one [76].

Multiple Tasked The bot is multiple tasked if it is capable of handling more than one stimuli or task at once [76]. Multi-task bots may process tasks based on the order in which they arrive (e.g., queues requests or adds them to a stack), some prioritization metric (e.g., highest criticality first), or randomly.

5.3.3 Adaptability

A bot's adaptability refers to the bot's ability to modify its own behaviour or functionality at runtime. In general, these adaptations are the bot's attempt to improve its effectiveness or optimize its performance. Hypothetically, a bot may be able to adapt any of its behaviours. More specifically, a bot can adapt any of its intrinsic or interaction facets or sub-facets.

There are two **exclusive** adaptability states:

Non-Adaptive The bot is not able to change its behaviour at runtime [65]. A non-adaptive bot's behaviours remain unchanged after it has started running.

Adaptive The bot is able to change at least some of its behaviours at runtime. If a bot is adaptive, then it can be further described by the following sub-facets: **constraints**, **source**, and **guidance**.

5.3.3.1 Constraints

If a bot is adaptive, it can also be described by its adaptation constraint. These constraints determine restrictions on the bot's adaptation capabilities. [77, 65].

A bot can have varying levels of adaptation constraints based on the behaviour (i.e., facet or sub-facet) it is trying to adapt, as well as the *source* or level of *guidance* provided during adaptation.

Possible constraint values **range** from *constrained* to *open*:

	Constrained	The bot is able to adapt its dimensions, but it is restricted by scope, extent, or activity [65]. This provides safeguards to ensure the bot still functions well on its core responsibilities.
↓	Open	The bot is freely able to adapt its behaviour.

It should be noted that adaptation constraints differ from reasoning agency as the constraints block the bot's ability to adapt even if it had the agency to do so.

5.3.3.2 Source

If a bot is adaptive, it can also be described by the source that triggered its adaptation. The source of adaptation describes where the bot's motivation to adapt stemmed from. The source of a bot's adaptation can vary based on the type of behaviour (i.e., facet or sub-facet) it is adapting.

There are two **boolean** source sub-facets:

- ⊕ **Internal** The bot's adaptation process is triggered from within the bot itself. Internal sources for adaptation are likely to be logically programmed into the bot by its designer (i.e., by programmer or creator of the bot [61]).
- ⊕ **External** The bot's adaptation process is triggered by something in the bot's environment. These sources include interactions, communication [71], social streams [50], other systems, environmental changes [77], etc.

It should be noted that the bot's *adaptation source* differs from the *knowledge source*, since knowledge can be accumulated without undergoing a behavioural adaptation.

5.3.3.3 Guidance

Guidance refers to whether or not the bot had some form of help, support, or supervision during its adaptation process. After adaptation has been triggered, the direction provided during or after adaptation can influence the outcome of the adaptation process.

Different types of adaptations may require different levels of guidance. Thus, the type of guidance may differ based on the type of behaviour (i.e., facet or sub-facet) that is being adapted.

There are two **boolean** guidance sub-facets:

- ⊕ **Undirected** The bot's adaptation is undirected if its adaptation outcome is not directly shaped or influenced by a source. Undirected adaptation is guided from within (e.g., from its programming) and the bot itself controls the adaptation process.
- ⊕ **Directed** The bot's adaptation is directed if its adaptation outcome is shaped by the source's actions. If a bot is directed, then it can be further described by the following sub-facets:

- (i) **Configuration:** A form of directed adaptation in which an external force directly changes or manipulates the bot's existing behaviours. Typ-

ically, these types of changes happen at a high level and users do not directly change the bot’s code. Some examples of this type of change include users updating bot settings or changing the bot’s configuration.

- (ii) **Subscription:** A form of directed adaptation in which an external force directly adds or removes behaviours [65]. Typically, these types of changes happen at a lower level and users directly change the bot’s code to add “*additional layers of competency*” [65]. A great example is the addition of new ‘*Alexa skills*’ to Amazon’s Alexa.
- (iii) **Reinforcement:** A form of directed adaptation in which the source shapes the adaptation by either rewarding or punishing the outcome of the bot’s adaptation [71]. Generally this happens after the adaptation has occurred, thus shaping the bot’s behaviour through the source’s response to the changes. The bot uses trial and error to determine the correct adaptations [71].

5.3.4 Goals

All bots have goals, which are one or more future states the bot is working towards or attempting to achieve. However, there is a huge variety in the types of goals bots can try to achieve. Since many types of goals exist, a bot’s goals can be further described by the following sub-facets: **complexity**, **criticality**, **attainability**, **explicitness**, and **source**.

A single bot may work towards a variety of different goals (and in most cases does). Each of a bot’s goals should be ranked independently according to the goal sub-facets.

5.3.4.1 Complexity

The bot’s goal complexity describes how complicated the bot’s goals are. The complexity of goals is a rather subjective measure, but provides some insight into the overall capabilities of the bot. Since bots can also have many different types of goals, each goal may have a different level of complexity.

Possible complexity values **range** from *low* to *high*:

	Low	The bot’s goals have a low complexity if it is a simple task.
↓	High	The bot’s goals are complex.

5.3.4.2 Criticality

The bot’s goal criticality describes the level of risk, importance, or urgency associated with the goal. While the specific type of criticality may differ from bot to bot, the criticality

sub-facet captures the overall perceived criticality. The criticality of the bot's goals is a subjective measure, but provides some insight into the overall risks associated with the bot.

Since bots can have many different types of goals, each goal may have a different level of criticality. The degree of risk may differ depending what the bot's goals are, the setting/environment (e.g., personal, cooperate, government), who it's interacting with (e.g., children or other risk groups, safety-critical systems), etc.

Possible criticality values **range** from *low* to *high*:

Low High ↓	The bot's goal criticality is low if the tasks are low risk, of low importance, or do not have any safety/security concerns. For example, a bot that sets a kitchen timer would have low overall criticality. The bot's goal criticality is high if the tasks are high risk, of high importance, or have any safety/security concerns. For example, a bot that sets a timer for some form of safety-critical system would have high criticality.
--	--

5.3.4.3 Attainability

The bot's goal attainability is the bot's ability to complete or achieve its goal. The attainability does not reflect how *likely* the bot is to achieve its goal, but rather if the goal itself can be completed (i.e., does the goal have an explicit end state). Since bots can have many different types of goals, each goal can have a different degree of attainability.

There are two **boolean** attainability sub-facets:

- ⊕ **Achievable** The bot's goal is achievable if the goal has an explicit end state that can be successfully reached [61]. Once the goal's end state has been reached, the goal is considered complete and the bot is no longer actively working toward this goal.
- ⊕ **Homoeostasis** The bot's goal attainability is homoeostasis if its goal is never considered complete [61]. For homoeostasis goals, either the bot's desired goal state can never fully be reached or the bot must try to maintain the desired state indefinitely.

5.3.4.4 Explicitness

The bot's goal explicitness is the degree to which the bot's goals are explicitly defined or described [66]. Since bots can have many different types of goals, each goal may have a different level of explicitness.

There are two **boolean** explicitness sub-facets:

- ⊕ **Explicit** The bot’s goal is explicit if it is clearly defined, described, and with no room for interpretation [66]. For example, asking Alexa to find the *next* show time of movie *x* at theatre *y*.
- ⊕ **Implicit** An implicit goal is not clearly defined, but instead ambiguous [66]. There may be many possible interpretations of an implicit goal. For example, asking Alexa to find a *good movie* playing *nearby*.

5.3.4.5 Source

The bot’s goal source describes where the bot’s goals originated from. More specifically, the goal source describes what triggered a new instance of a goal behaviour. Since bots can have many different types of goals, each of its goals may stem from a different source.

There are two **boolean** source sub-facets:

- ⊕ **Internal** Internal goals are derived from within the bot (at runtime), or provided to it in its source code (before runtime).
- ⊕ **External** The bot’s goal source is external if its goals are adopted from external stakeholders (e.g., user, another bot, system) or something in its environment. External goals can only originate at runtime.

It should be noted that *goal source* differs from *adaptation source*, as goal source triggers a new instance of a *preexisting* goal behaviour. That is, the bot was capable of working towards that goal prior to the source activating it. The adaptation source triggers an adaptation of a behaviour, which could be learning a new type of goal behaviour.

5.3.5 Delegation

A bot’s delegacy describes its permission or authority to act on behalf of or to represent others [61, 66]. The bot may act as a delegate for any of the inhabitants of its environment (e.g., users, groups of users, other bots, systems). The bot’s degree of delegacy may vary depending on the *goal* or type of *action* it’s required to perform, as well as who the bot is acting as a delegate for.

Possible delegacy values **range** from *none* to *complete*:

↓	None	The bot does not have the authority to act on behalf of others. However, bots in this category can appear to be acting on behalf of others, but do so without permission and likely with malicious intent.
	Partial	The bot has authority to do things on behalf of the user, but does not pretend to be the person it's representing. For example, Google Duplex will call a business and book services on a user's behalf.
	Complete	The bot has the authority to both act on behalf of and pretend to be the user themselves. For example, an online ticket purchasing bot may pretend to be the user in order to purchase the ticket on their behalf.

5.3.6 Specialization

The bot's specialization is the degree to which the bot focuses its efforts in a specific area. More specifically, a bot specialization describes how similar the bot's goals are with respect to the types of tasks, domain, etc.

Possible specialization values **range** from *generalist* to *specialist*:

↓	Generalist Bot A generalist bot supports a wide range of (usually simple) tasks, and directs users to the appropriate external resources when deeper knowledge is required [46]. Generalist bots are often the first step in a multi-step process that guides users to the next correct step [46]. A good example of a generalist bot is Apple's Siri, who can perform a variety of simple tasks but re-directs users when deeper knowledge is required to perform the task.
	Specialist Bot A specialist bot is designed to perform specific tasks in a limited domain [46]. These tasks are generally similar in purpose/focus. These bots typically have deeper knowledge in a specific domain than generalist bots, and apply this knowledge to help them complete their goals. An example of a specialist bot is DoNotPay, which guides users through the process of disputing parking tickets.

Since the purpose of this taxonomy is not to present another role-based classification of bots, specialization can be viewed as a high-level way of conceptualizing the bot's purpose without singling out specific roles.

5.3.7 Anthropomorphism

A bot's level of anthropomorphism is the degree to which the bot has been given human-like characteristics or traits. Most software bots exude at least some degree of anthropomorphism. Since anthropomorphism is a very high-level concept, this facet is broken down into

eight sub-facets: **name**, **age**, **gender**, **ethnicity**, **embodiment**, **profession**, **personality**, **emotions**.

Although this taxonomy identifies some common types of bot anthropomorphism, this list is by no means comprehensive but instead focuses on some key types of anthropomorphism we believe will aid in a better understanding of software bots. We believe that the sub-facets of anthropomorphism will likely grow as developers continue to build more human-like bots.

5.3.7.1 Name

The bot's name describes how others refer to the bot, address it, or identify it [39]. This facet describes the degree to which the bot has its own name.

There are three **exclusive** name states:

- None** The bot does not have a name. These bots are likely addressed by some generic file name or not at all.
- Representative** The bot takes the name of the company it represents or the service it provides. For example, `Mention-Bot`'s name represents the service it provides.
- Unique** The bot has been given its own identifiable name. Alexa, Siri, and Cortana, the voice assistants from Amazon, Apple, and Microsoft, respectively, were all given their own unique names.

5.3.7.2 Embodiment

The degree to which bots have been given a visible icon, figure, or form to represent themselves [39, 78]. Since software bots live in the virtual world, their embodiment must also be virtual.

There are two **exclusive** embodiment states:

- None** If the bot does not have a visible form, it has no embodiment.
- Embodied** If the bot has a visible form, then it is embodied. If the bot is embodied, then it can be further described by the following sub-facets:
 - $\oplus$ **Logo** The bot has a logo that does not take a life-like form. For example, Alexa has non life-like logo.
 - $\oplus$ **2D Avatar** The bot has a two-dimensional life-looking avatar, which represents the bot (e.g., 2D agents [78]). For example, `Poncho`'s avatar is a rain-gear wearing cat and `MentionBot`'s avatar is a smiling robot.

⊕ **3D Avatar** The bot has a three-dimensional life-looking avatar, which represents the bot (e.g., 3D agents [78]).

5.3.7.3 Age

The bot’s age describes whether the bot has a visible or identifiable age . The anthropomorphized age of the bot is a superficial age provided to the bot by the bot’s developer. Those interacting with the bot can either visibly see the bot’s age, or the bot may reveal its age in some way. A bot’s age may or may not be correlated with the bot’s *lifespan*.

There are three **exclusive** age states:

- None** The bot does not have a visible or identifiable age.
- Static** The bot has a visible or identifiable age, but its age does not increase over time. For example, Apple’s Siri claims to be the “*same age as you are*”.
- Dynamic** The bot has a visible or identifiable age and it continues to age over time. For example, Amazon’s Alexa calculates its age based on the day it was released.

5.3.7.4 Gender

The bot’s gender describes whether the bot has a visible or identifiable gender [39]. Those interacting with the bot can either visibly see the bot’s gender, or the bot may reveal its gender in some way.

There are two **exclusive** gender states:

- None** The bot does not have a visible or identifiable gender.
- Gendered** The bot has a visible or identifiable gender. For example, Alexa says it’s “*female in character*”, while Poncho claims to be male.

5.3.7.5 Ethnicity

The bot’s ethnicity facet describes whether or not the bot has a visible or identifiable species, race, or ethnicity [39]. More specifically, those interacting with the bot can either visibly see its ethnicity or the bot reveals its ethnicity in some way. The bot’s ethnicity may be visible as a result of its embodiment (e.g., logo or avatar).

There are two **exclusive** ethnicity states:

- None** The bot does not have a visible or identifiable species, race, or ethnicity.
- Ethnicity** The bot has a visible or identifiable species, race, or ethnicity. For example, Poncho will tell you it’s a cat (i.e., species). Alexa, Siri, and Cortana, all, by default, have North American accents.

5.3.7.6 Profession

The bot’s profession facet captures whether or not the bot has a visible or identifiable profession. Those interacting with the bot can either visibly see the bot’s profession (e.g., wearing a uniform), or the bot reveals it in some way.

There are two possible **profession states**:

- None** The bot does not have a visible or identifiable profession.
- Profession** The bot has a visible or identifiable profession. The bot’s profession might be similar to or dramatically different from the tasks that the bot performs. For example, *Eliza* is a therapist.

5.3.7.7 Personality

The bot’s personality is made up of the behaviours, quirks, or characteristics that give it a unique and distinctive character. The American Psychological Association describes personality as the “*individual differences in characteristic patterns of thinking, feeling, and behaving*”[79] Early research has shown that a bots personality changes the way in which users interact with it . But too much personality might not be a good thing, either. According to Slack, “*a little goes a long way*”.⁹⁶

There are two **exclusive** personality states:

- None** The bot does not have a deliberate, anthropomorphized personality.
- Personality** The bot has a personality. The amount of personality a bot has can vary dramatically, just like with humans. The *FoodNetwork* bot, for example, will remind you, “*does it matter? I’m a bot!*” when you ask it personal questions. Other bots have hours of content developed to answer even the silliest of user questions.

It’s worth noting that *personality* is a very subjective and complex dimension. In this taxonomy, we simply provide a binary *has a personality or does not have added personality* facet. However, personality is a complex category and we strongly encourage the expansion of this taxonomy by breaking this dimension down into further sub-facets.

5.3.7.8 Emotions

The bot’s emotions are the degree to which the bot appears to have human-like emotions or attitudes [71]. These emotions can range from basic emotions (e.g., fear, happiness, sadness), to more advanced emotions (e.g., guilt, jealousy, pride), although we do not distinguish between them in this taxonomy.

Emotions can trigger interactions, influence reasoning/adaptation, influence goals, bias behaviours. Emotions can be impacted by the environment or interactions.

There are three possible **emotion states**:

None	The bot does not have visible emotions.
Superficial	The bot has superficial emotions if it only displays its emotions when interacting with others. These emotions do not have an effect on the bot's behaviour. When you insult Siri, for example, it responds with “[t]here’s no need for that!” or “[w]hat did I do to deserve that?”, but its behaviour otherwise remains the same.
Logical	The bot has logical emotions if it both displays and has the ability to use emotions. The bot's current emotional state (i.e., the emotion the bot is currently feeling) has an influence on its other behavioural facets. For example, if you insult Poncho it will first respond with, “[u]h... rude”, but if you continue the abuse Poncho says, “[o]k, well then I think I’m going to take a short break,” and then ignores you for a few minutes.

Similar to *personality*, it's worth noting that *emotions* are a very subjective and complex dimension. In this taxonomy, we provide a simple division between emotions that have an effect on the bot's behaviour, and ones that do not. However, emotions are a complex category and we strongly encourage the expansion of this taxonomy by breaking this facet down into further sub-facets. If a bot has a visible *personality* it is usually more likely to also show *emotions*.

5.3.8 Life Cycle

The bot's life cycle describes the various phases that the bot goes through in its “*life*”. Since life cycle is a very high-level concept, it is broken down into three sub-facets: **lifespan**, **creation**, **reproduction**, and **adaptation**.

5.3.8.1 Creation

The bot's creation is the way in which the bot was brought to life [39]. More specifically, creation describes the process in which the bot's code was generated and executed.

There are three **exclusive** creation states:

Human	The bot was created by a human [39]. Most software bots are created and executed by humans.
Bot	The bot was created by another bot. For example, BotFather will create a new bot based on your requests.

System The bot was created by another system. For example, many bots in online games are spawned by a system.

5.3.8.2 Lifespan

The bot's lifespan refers the length of time that the bot continues to function. Lifespan specifically refers to the time the bot would have run if left completely alone with ample resources (i.e., if it did not receive any external intervention such as turning it off or a power outage).

There are three **exclusive** lifespan states:

Terminating The bot has a terminating lifespan if it eventually stops on its own accord. We further break this down in two categories of terminating programs:

- (i) **Determinate**: The time taken for the bot to terminate has exact or predictable limits.
- (ii) **Indeterminate**: The time taken for the bot to terminate is not known, defined, or predictable.

Transient The bot has a transient lifespan if it passes in and out of existence [80]. The bot will pass into existence for a period of time, often to complete a task, then disappears until it is needed again. The bot's passing in and out of execution can be internally or externally triggered. Sometimes bots that are tied to a specific *platform* operate in a transient manner.

Continuous The bot has a continuous lifespan if it never self-terminates. The only way in which a continuous bot stops its execution is if it's acted upon by an external force that causes it to stop.

5.3.8.3 Reproduction

The bot's reproduction describes the bot's ability to spawn other bots [64]. The bot can spawn a new bot by triggering a new instance of an existing bot or by creating a new bot itself.

There are two **exclusive** reproduction states:

None The bot is not able to trigger or create new bots.

Reproductive The bot is able to trigger or create new bots. They also have the ability to trigger the same type of program they are, a different type of program, or multiple types of programs.

5.4 Interaction Dimensions

The interaction dimension is composed of facets that describe the bot's interactions with the different elements in its *environment*. There are a total of seven facets and eight sub-facets that fall under the interaction dimension, as shown in Figure 5.6.

Although these interaction facets touch somewhat on the inner workings of the software bot itself, they still take a black box approach to examining bot behaviours. More specifically, they try to focus on the wide range of externally observable behaviours that the bot can exude when interacting with the various elements in its environment.

5.4.1 Access

A bot's access is the degree to which the bot has access to its environment [66, 43]. Since the environment can impose restrictions on both the bot's ability to *sense* or *act* within their environment, the bot may have different degrees of access for sensing and acting.

Possible access values **range** from *partial* to *complete*:

None	The bot is not allowed to access any of its environment [43].
Partial	The bot is allowed to access a subset of its environment [43, 66].
↓ Complete	The bot is allowed to access all of its environment [43, 66].

5.4.2 Sense

A bot's sensing describes the degree to which the bot is able to perceive stimuli in its environment. The type of stimuli the bot is able to sense is limited by its *access*, as well as which type of *knowledge* it understands.

There are two **exclusive** sense states:

Non-Sensing	The bot is non-sensing if it does not try to perceive any external stimuli in its environment.
Sensing	The bot is sensing if it tries to perceive a limited set of stimuli in its environment. Sensing is supported by sensors [55, 43, 5]. Different sensors help the bot perceive different properties in its environment.

5.4.3 Act

A bot's acting describes the bot's ability to act upon or make changes in its environment [69]. The type of actions the bot is able to perform is limited by its *access* as well as the type of actions the bot's *reasoning mechanisms* are capable of selecting and have the *agency* to perform.

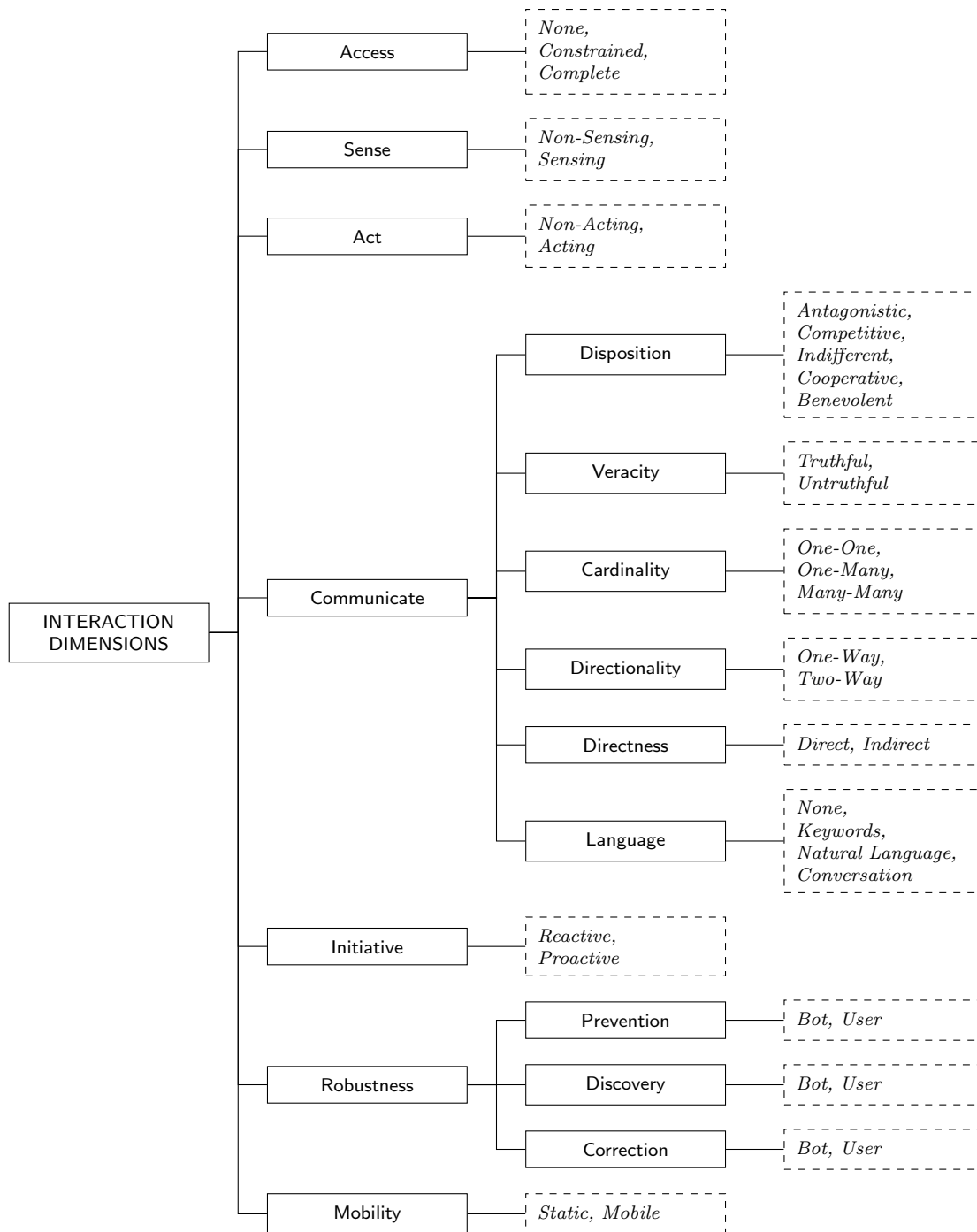


Figure 5.6: The software bot taxonomy's interaction dimensions. The dimensions/facets/sub-facets, as well as the range of possible values which the facet can take on, are shown in the solid and dashed boxes, respectively.

There are two **exclusive** act states:

- Non-Acting** The bot is non-acting if it does not try to act upon or make changes to its environment.
- Acting** The bot is acting if it tries to act upon or make changes to its environment. Acting is supported by effectors [55, 43] or actuators [5]. Different effectors help the bot perform different actions in its environment.

5.4.4 Communicate

A bot's communication is the degree to which the bot is able to meaningfully interact with others in its environment. To have the ability to communicate, a bot must be able to both *act* and *sense*, and also have *other inhabitants in its environment*. These requirements also extend to the sub-facets of communicate.

There are two **exclusive** communicate states:

- Non-communicative** The bot is non-communicative if it does not interact with others in its environments [69].
- Communicative** The bot is communicative if interacts with others in its environments [69, 81, 43, 82, 83, 84, 61, 71, 5]. If a bot is communicative, it can be further described by the following six sub-facets: **disposition**, **veracity**, **cardinality**, **directness**, **speech**, and **centralization**.

5.4.4.1 Disposition

If a bot is communicative, it can also be classified by its disposition. A bot's disposition describes its willingness to help, perform actions for, or share resources with others in its environment [65]. A bot may display different patterns of disposition based on who it's interacting with, the type of request, the bot's current goal(s), its current internal state, the current context, etc.

Possible disposition values **range** from *antagonistic* to *benevolent*:

Antagonistic	The bot is antagonistic if it makes purposefully antagonistic attempts to inconvenience or undermine others. Many antagonistic bots exist in online game settings.
Competitive	The bot is competitive if it acts in favour of its own self-interests [65, 66, 71, 43, 5]. A competitive bot will collaborate with others only when the outcome benefits them as well.
Indifferent	The bot is indifferent if it is unaware of the needs of others (inadvertently or by choice). An indifferent bot also does not concern itself with the actions of others. The bot does not assist others in its environment, nor does it purposefully attempt to inconvenience or undermine them.
Cooperative	The bot is cooperative if it is willing to help others in its environment, potentially sacrificing its own goals. The bot is <i>collaborative</i> if it coordinates its efforts with others [64, 85, 86].
Benevolent	The bot is benevolent if it always helps others in its environment, even if it is detrimental to its own goals or best interests [65]. Most of the common user-focused bots fall under this category. For example, Alexa at least attempts to perform any task that is asked of it.

5.4.4.2 Veracity

If a bot is communicative, it can also be classified by its veracity. A bot's veracity refers to the bot's deliberate adherence to or divergence from the truth during its communications [65]. A bot may display different patterns of veracity based on who it's interacting with, its current internal state, the current context, etc.

Possible veracity values **range** from *truthful* to *untruthful*:

Untruthful	The bot's veracity is untruthful if it intentionally attempts to deceive the individuals that it interacts with [65]. A good example of untruthful bots are impersonator bots.
Mixed	The bot's veracity is mixed if it exhibits both deceiving and truthful tendencies based on the situation.
Truthful	The bot's veracity is truthful if it does not attempt to deceive the individuals it interacts with [65].

5.4.4.3 Cardinality

If a bot is communicative, it can also be classified by its cardinality. A bot's cardinality describes the number of users the bot's environment is capable of communicating with at the same time.

There are three **boolean** cardinality sub-facets:

- ⊕ **One-One** The bot is one-to-one if it is only capable of interacting with one individual at a time.
- ⊕ **One-Many** The bot is one-to-many if it is capable of interacting with many users at the same time [39]. In this case, the bot might be required to distinguish between users.
- ⊕ **Many-Many** The bot is many-to-many if it is capable of interacting with many users while they are also interacting among themselves. In this case, the bot must understand when the users are trying to interact with the bot versus each other.

5.4.4.4 Directionality

If a bot is communicative, it can also be classified by its directionality [87]. A bot's directionality describes whether the bot's interactions with users is bidirectional.

There are two **boolean** directionality sub-facets:

- ⊕ **One-Way** The bot is one-way if it is only capable of receiving inputs from or sending outputs to a user.
- ⊕ **Two-Way** The bot is two-way if it is capable of both receiving inputs from and sending outputs to a user.

5.4.4.5 Directness

If a bot is communicative, it can also be classified by its directness. A bot's directness describes the way the bot communicates with others in its environment [81]. A bot may display different levels of directness based on who it's interacting with, its current internal state, the current context, etc.

Possible directness values **range** from *direct* to *indirect*:

- Indirect** The bot's communication is indirect if it communicates with others indirectly (e.g., through mediators, non-message interactions, or artifacts) [81, 84].
- Direct** The bot's communication is direct if it communicates with others directly (e.g., direct messages or requests) [81, 84, 80, 50].

5.4.4.6 Language Capability

If a bot is communicative, it can also be classified by its conversational ability. A bot's conversation facet reflects its ability to communicate through human understandable language.

A bot may display different levels of conversation based on its *goals*, current internal state, the current context, who it's interacting with, etc. For a bot to be able to use human language, the bot has to also be able to understand human language (see *knowledge*). A bot may be able to converse in multiple languages or in just a single language.

Possible dialog values **range** from *none* to *conversation*:

None	The bot's level of speech is none if it is not able to use human language.
Keywords	The bot's level of speech is keywords if it is only able to communicate with others using keywords or short, predetermined phrases [38, 75]. For example, <code>MentionBot</code> is able to post a scripted message on pull-requests.
Natural Language	The bot's level of speech is natural language if it is able to communicate with others using natural language [38, 70, 39]. Most chatbots fall into this category.
Conversation	The bot's level of speech is conversation if it is able to engage in a meaningful two-way dialogue with others in its environment [70, 38].

5.4.5 Initiative

The bot's initiative describes the way that the bot's interactions with its environment are initiated. A bot's initiative is related to its ability to *sense* stimuli in its *environment*, reason appropriately about the changes it detects, and affect change via its *actions*. Furthermore, a bot's initiative can vary depending on its *goals*, current internal state, the current context, etc.

Possible initiative values **range** from *reactive* to *proactive*:

Reactive	The bot is reactive if it initiates actions in response to a specific stimuli, " <i>motivational cue</i> " [77], or trigger in its environment [80, 65, 55, 88, 69, 70].
Proactive	The bot is proactive if it takes action to control the situation rather than responding after something has happened [78, 55, 52, 69, 85]. For example, a bot might remind users that they haven't been to the gym in a few days.

It should be noted that a bot's initiative might also change based on the lens you're looking through. Thus, for the scope of the thesis, we examine initiative through the lens of what the bot is interacting with. For example, if the bot is interacting with a user, we consider initiative from the user's perspective of the bot.

5.4.6 Robustness

The bot's robustness describes its ability to handle errors or ambiguity [89]. A bot's robustness encompasses the full life cycle of dealing with errors or ambiguity. Since robustness is a high-level concept, it can be broken down further into three sub-facets: **prevention**, **discovery**, and **correction** [90].

A bot may display different levels of robustness based on who it's interacting with, its current internal state (see knowledge representation), the current context, etc.

5.4.6.1 Error Prevention

The bot's error prevention describes the strategies that it uses to reduce or prevent errors when receiving inputs from users [90].

There are two **boolean** prevention sub-facets:

- ⊕ **Bot** The bot is responsible for preventing errors in the inputs it is receiving. Some strategies for bot-led error prevention include using less error-prone designs to influence/constrain users during bot-user interactions (e.g., guided dialogues, restricted inputs, structured responses), or using additional/contextual information to supplement the user's inputs (e.g., episodic memory, user data, user settings, user preferences) [90].
- ⊕ **User** The bot relies on users adapting their behaviour to select less error-prone input mechanisms. In this case, the bot must be designed to support the user's ability to spontaneously alter their input or interaction behaviours. Some strategies to help the user prevent errors include changing input styles (using text inputs over voice in loud environments), or using redundancy (showing the results of voice to text) [90].

5.4.6.2 Error Discovery

The bot's error discovery describes the strategies that the bot uses to detect errors in the inputs it has received [90].

There are two **boolean** discovery sub-facets:

- ⊕ **Bot** The bot is responsible for discovering errors in the inputs it receives. Some strategies to help the bot discover errors include comparing the request against its knowledge base to identify unusual inputs, using more than one modality to determine possible interpretations of inputs, or using statistical model thresholds to determine incorrectly identified inputs [90].

- $\oplus$ **User** The bot relies on users to detect errors. The bot provides the user with its interpretation of the input, and the user determines if the bot interpreted it correctly. Some strategies to help the user discover errors include:
- (i) **Implicit:** The bot repeats its interpretation of the input back to the user after the bot executes the task, usually along with the result [90].
 - (ii) **Explicit:** The bot repeats its interpretation of the input back to the user before the bot executes the task [90].
 - (iii) **Alternatives:** The bot provides users with a set of possible interpretations [90].
 - (iv) **Multi-Modal:** The bot uses a different mode to confirm its interpretation (e.g., the bot uses text to confirm a spoken input) [90].

5.4.6.3 Error Correction

The bot's error correction describes the strategies that the bot uses to recover from detected errors in the inputs the bot has received [90].

There are two **boolean** correction sub-facets:

- $\oplus$ **Bot** The bot is responsible for correcting errors in the inputs it receives. The bot can recover from errors in much the same manner that it detects the errors. Some strategies to help the bot recover from errors include rejecting inputs that fail to meet a threshold, attempting to correct errors using its existing knowledge, and correcting it based on simultaneous inputs (i.e., multi-modal) [90].
- $\oplus$ **User** The bot relies on the user to correct errors, and it provides the user with a way to correct misinterpreted information. Some strategies to help the user recover from errors include allowing the user to correct the mistake using the same modality (e.g., repeating, spelling it, rephrasing it) or from a different modality (e.g., switching from voice to writing) [90].

Since robustness of a bot was not well defined in the existing literature on software bots, I adapted the error handling taxonomy presented by Bourguet [90] to relate to software bot robustness.

5.4.7 Mobility

A bot's mobility describes its ability to move around within its environment. A bot's degree of mobility can vary for different dimensions. A bot could be mobile in where it *interacts* (e.g., acts, senses, communicates) and/or where it *reasons*. For the scope of this taxonomy,

however, we pay most attention to the mobility of the bot’s *interaction* facets, since a move of *reasoning* would not typically be visible.

Possible mobility values **range** from *static* to *mobile*:

Static	The bot is static if it is not able to move within its environment [65].
Mobile	If the bot is not static, then it is mobile. A mobile bot can move around its environment [66, 69, 65]. However, the degree of the bot’s mobility can vary. For example, some Slack bots can move between channels (low mobility). On the other hand, GoogleBot and BingBot, both web crawlers, are examples of highly mobile bots.

There is a difference between a mobile bot and a bot that exists on multiple platforms. A mobile bot is, at least in some capacity, able to control its ability to move around. A multi-platform bot, however, usually has a new instance of the bot for each platform. For example, multiple Alexa-compatible devices that were synced to the same account does not make Alexa a mobile bot. Similarly, the ability for a Facebook Messenger bot to be accessed through multiple devices does not make the bot mobile.

5.5 Summary

In this chapter, I presented an updated, holistic taxonomy of software bot characteristics. I described each of the three top-level dimensions (environment, intrinsic, and interaction), the facets/sub-facets that fall under each of the dimensions, the range of values for each facet, and included examples of modern software bots for illustration. I also provided a brief readers guide to help users interpret, use, and expand the proposed taxonomy.

For the remainder of this thesis, I discuss the taxonomy’s validation process (cf. Chapter 6), explore the implications of the proposed taxonomy (cf. Chapter 7), and provide suggestions for how the taxonomy can be operationalized in the future (cf. Chapter 7).

⁹⁶<https://api.slack.com/best-practices/voice-and-tone>

Chapter 6

Taxonomy Validation

To ensure that the taxonomy correctly captured the range of observable software bot properties and behaviours, I validated it by:

- (i) **benchmarking** the proposed taxonomy against the existing classifications of software bots (Section 6.1),
- (iii) demonstrating the utility of the taxonomy through the **tagging** of three publicly available software bots (Section 6.2), and
- (ii) testing the utility and usability of the taxonomy through a **domain expert tagging** session (Section 6.3).

I describe each of the three approaches in greater detail below. Any limitations to my approach are discussed in the following chapter (cf. Chapter 7.2).

6.1 Benchmarking

The software bot taxonomy was validated through benchmarking, the process of comparing a new taxonomy against existing classifications [1]. The proposed software bot taxonomy was benchmarked against the classifications used to generate it (cf. Section 4.3) in order to determine if the merging of the taxonomies was completed correctly. Since I tried my best to ignore the structure of the classifications during the extraction phase (cf. Section 4.4.1), benchmarking provided insights into how the new taxonomy compared to the ones that provided the data used to generate it.

Overall, the software bot taxonomy covered almost all of the observable properties and behaviours of software bots that were explicitly described in the literature search and snowballed articles. If the properties were not identified directly in the new taxonomy, some abstracted version of them usually was. This helped validate that the terms from the previous classifications were properly merged to create the new software bot taxonomy. It should be noted, however, that many of the articles also explicitly described some properties

that were outside of the new taxonomy’s scope and were not included, e.g., implementation details, architectural details, languages used, etc. Although most of the online articles did not provide a formal classification of bots, I also benchmarked them to show how the articles both influenced the creation of and compared to the new taxonomy of software bots.

The detailed results from the benchmarking of each existing classification against my proposed software bot taxonomy are provided in Appendix F. In the following chapter, I also discuss some trends that emerged during the benchmarking validation step.

6.2 Subject Matter Tagging

During this phase, the proposed taxonomy was used to characterize three existing software bots, which were publicly available online. Since I only assessed a relatively small sample of software bots, the bots were selected opportunistically. I selected the bots with a range of different purposes: one virtual assistant (Alexa⁹⁷), one weather bot (Poncho⁹⁸), and one software development bot (Mention-Bot⁹⁹).

The first bot, Amazon’s Alexa, is a voice service that can be accessed from a variety of Amazon and third-party devices. Alexa responds to voice requests for a set of basic built-in capabilities (e.g., making to-do lists, playing music, or providing traffic reports), and optional user-installed skills (e.g., workout plans, talk to your cat, or wine pairings) from third-party vendors. Alexa can also be used for home automation as it can control many smart devices. It used the online EchoSim¹⁰⁰ testing tool to access Alexa for the classification. EchoSim is an online simulator of Alexa’s functionality, used by developers to test their *Alexa Skills*. The second bot, Poncho, is a Facebook Messenger bot that provides personalized weather updates, ranging from daily updates to extreme weather warnings. Poncho is also capable of playing text-based games with users. Lastly, Mention-Bot is a GitHub-based bot that automatically mentions potential reviewers on pull requests. Mention-Bot was designed for large GitHub projects that are too big for people to efficiently monitor notifications. By automating the process of mentioning potential reviewers on new pull requests, Mention-Bot notifies the right people early on. It should be noted that for the purpose of classifying Mention-Bot, I assessed its behaviours as if it was being used on a public repository.

Each of the three bots was classified using the proposed software bot taxonomy in May 2018. Since two of the bots could respond to natural language queries, I determined their values for many of the intrinsic and interaction facets by directly asking them questions. For example, I asked them questions like “*what is your gender?*” or “*what is your age?*” to determine their *anthropomorphized gender* and *age*, respectively. For the facets that could not be asked of them directly, I assessed their capabilities by interacting with them further or (when required) examining their documentation.^{6.2,101} I determined the values of the bots’

environment facets by examining their platform’s documentation, specifically Messenger¹⁰², GitHub¹⁰³, and Amazon’s voice services¹⁰⁴. A summary of each of the bots’ classifications on the environment, intrinsic, and interaction dimensions is provided in Appendix G.

However, without deeper insights into the inner workings of the bots, it is hard to know if these classifications are completely accurate as I may have missed some of the bots’ less visible functionality. I also acknowledge that these bots are likely to change, which in turn would alter their classifications on the proposed software bot taxonomy.

During this validation step, I noticed that there was one facet that I had originally overlooked during the taxonomy creation; the bot’s *interaction directionality* (i.e., does the bot engage in a two-way or one-way interaction style). Both Alexa and Poncho had a two-way interaction style, while Mention-Bot had a one-way interaction style since it could only post messages for users and not receive inputs from them. To account for this, I added the additional facet to the proposed taxonomy.

6.3 Domain Expert Tagging

To further validate the utility of the proposed taxonomy, I conducted a domain expert tagging session. At the start of the session, the participant was asked a few quick questions about their background and prior experience with software bot technologies. They were then asked to try classifying their bot using the latest version of the software bot taxonomy. The participant was free to ask any questions they had during the session. After the participant was finished tagging their bot, they were asked a few follow up questions regarding their experience using the software bot taxonomy.

The participant was invited to participate because of their previous experience researching and developing software bots. Informed consent was collected at the time of the study and the research was conducted with the approval of the Human Research Ethics Board (HREB) of the University of Victoria (cf. Appendix I). A copy of the questions asked in this study is provided in Appendix H.

The session took approximately 90 minutes and was conducted in person in a laboratory setting. The participant (P1) had approximately 8 months of experience in researching, designing, and building bots. At the time of the study, they worked in a quality assurance, testing, and customer support role at a local technology start-up. The bot (B1) they selected to classify was a bot that they built to automate some aspects of their company’s customer support. Since they had found that different customers often asked similar questions, P1 decided to build a bot to answer commonly asked questions and provide basic support for their offerings.

With the help of the facilitator, P1 was able to successfully tag their bot on all but two facets: cardinality (under the environment dimension) and autonomy (under the intrinsic

dimension). With respect to cardinality, P1 found the existing description of “*cardinality describes the size of the bot’s environment*” was not clear enough for them to be able to classify the bot’s environment as either discrete (“*limited or bound with respect to size*”) or continuous (“*not limited or bound with respect to size*”). Based on this feedback, I re-examined the facet, updated its description, and changed the facet value names from *discrete/continuous* to *bounded/unbounded*.

With respect to autonomy, P1 pointed out that B1 fit with the description of both low (“*it can come up with possible tasks, but requires an external party to approve of the bot performing them*”) and high (“*it can select and perform tasks completely on its own*”) autonomy. Upon reflection, I realized that I had combined multiple aspects of autonomy into a single facet. Autonomy, with respect to bots, can be viewed as the freedom to decide on a plan, the responsibility or ability to carry it out, and the degree to which you make your progress visible. Based on this, the high-level concept of autonomy was broken down into three separate *reasoning* sub-facets: **mechanisms**, **clearance**, and **visibility**. It also became clear that autonomy can differ based on the bot’s goal type. To account for this, I added two additional *goal* sub-facets: **complexity** and **criticality**.

The testing session also had the added benefit of providing some insight into the usability and understandability of the taxonomy. Although the bot was ultimately classified on all of the facets except for the ones previously mentioned, many possible usability improvements were also identified. For many of the facets, P1 requested either some form of an example or clarification. Based on this feedback, I added additional examples and revised the descriptions of the facets that caused the most confusion. After these changes had been made to the taxonomy, I provided P1 with the new version of the taxonomy and the update classification of B1. A copy of the classification of B1 is provided in Appendix J.

6.4 Summary

In this chapter, I discussed my three-phase approach to validating my proposed taxonomy of software bot characteristics. First, I benchmarked the taxonomy by comparing it against the existing classifications used to generate the taxonomy. This phase helped ensure that the terms for the original articles were correctly merged to create the new taxonomy. The second phase was the subject matter tagging, where I tagged three publicly available software bots using the proposed taxonomy. This phase helped ensure that the taxonomy’s facet values were complete. Lastly, I conducted a domain expert tagging session to test both the utility and usability of the taxonomy. During this, I worked with a domain expert to tag a bot they had built using the proposed taxonomy and gather their feedback on it.

All three of these phases were performed iteratively, and I used the insights gathered during the validation phase to further improve the proposed taxonomy. However, my valida-

tion approach was not without its limitations. In the next chapter, I discuss the limitations of the validation approach and provide suggestions to expand the validation in the future. I also discuss the implications of my proposed taxonomy of software bot characteristics and provide suggestions for how it can be operationalized.

⁹⁷<https://developer.amazon.com/alexa>

⁹⁸<https://www.poncho.is/>

⁹⁹<https://www.github.com/facebookarchive/mention-bot>

¹⁰⁰<https://www.echosim.io/>

¹⁰¹<https://developer.amazon.com/documentation>

¹⁰²<https://developers.facebook.com/docs/messenger-platform>

¹⁰³<https://developer.github.com/v4/reference/object/bot/>

¹⁰⁴<https://developer.amazon.com/alexa-voice-service>

Chapter 7

Discussion, Limitations, and Future Work

In this section, I reflect on the work presented in this thesis, identify some limitations to my approach, and explore how this work can be used in the future.

7.1 Why Another Software Bot Taxonomy?

One of the main questions that can be asked when reading this thesis is, why do we need yet another taxonomy for software bots? The purpose of the taxonomy proposed in this thesis was not to create another distinct taxonomy, but instead to bring together and build upon existing classifications to create a holistic taxonomy of software bots. In the next sections, I explore some shortcomings of the previous software bot classifications and highlight the ways in which my proposed taxonomy attempts to overcome them.

Limited Subject Matter: One of the major issues with the existing classifications of software bots was the limited scope of their subject matter. In fact, none of the existing classifications presented a holistic view of the larger domain of software bot technologies. Instead, they presented a classification of a limited sub-type of software bots and/or focused on a specific set of properties or behaviours. All but one of the articles returned in the systematic literature search and snowballing phases focused on software agents or more restrictive agent-subtypes. Similarly, many of the online articles focused on currently trendy chatbots. Many of these classifications also focused only on specific properties of behaviours of software bots or software bot sub-types. Although many of these classifications provide insights into the technologies they focused on, they miss some of the bot properties and behaviours that emerge when you consider the larger domain of software bots. They also missed the relationships between many of the facets in the taxonomy. To help mitigate these

issues, the content I used to build the taxonomy included different subtypes of software bots and a wide range of properties/behaviours.

Much of the terminology used in the different sub-domains of bot research was also extremely inconsistent. By merging the existing classifications together, I identified a *controlled vocabulary* for discussing the observable behaviours/properties of software bots. I was also able to map the *alternate terms* (describing the same concept) back to the controlled vocabulary [63]. Thus, the proposed taxonomy helps provide a more holistic view of the variety of observable software bot properties and behaviours.

Too Low Level: Many of the properties or behaviours of software bots described existing classifications were extremely low-level. Finding the correct level of abstraction for the various facets, sub-facets, and facet values required an immense amount of work. Often, too much information about a specific facet was presented, and I had to abstract away some of the details to make the taxonomy manageable. For example, the previous classifications contained numerous ways a bot could act or sense its environment. As a result, I abstracted these details into two higher-level categories (acting and sensing) instead of trying to present a comprehensive list of all possible actions they could perform or things they could sense. Although the taxonomy lost some details in this process, I believe that it created a much more manageable taxonomy. If needed, users can find more details by exploring the works referenced or even extend the more generic taxonomy to suit their own unique needs.

Too High Level: Conversely, when refining the proposed taxonomy’s facets, I found that many of the properties or behaviours described were too high level. Many of the existing definitions combined multiple potential facets into a single concept. These definitions were often ambiguous or used inconsistently by different researchers. In these cases, I decomposed the high-level concept down into multiple related sub-facets.

A good example of this is the concept of “*autonomy*”, as existing definitions often combine aspects of multiple potential facets. Although I describe the process of decomposing autonomy in greater detail in Section 6.3, I determined that it was composed of at least three distinct sub-facets: *agency*, *reasoning mechanism*, and *visibility*. Examples of other high-level concepts that were decomposed into many sub-facets include: *intelligence*, *sapiency*, *adaptability*, etc.

Other times, only a very high-level concept—with no obvious sub-facets—was provided. In these cases, I looked to literature outside of the domain of software bots to provide the additional information required to develop the facet values. For example, to design the sub-facets of software bot robustness, I drew inspiration from an existing taxonomy of error handling in human-computer interfaces [90]. I also relied on my previous experiences using software bots to help fill in other minor gaps.

Out of Date: The existing software bot classifications were also often out of date as many were from the 1990s and early 2000s before recent advances in AI (e.g., speech recognition, natural language processing, computer vision) increased computing capabilities, adoption of voice/messaging platforms, etc. These new technologies are the building blocks for many modern software bots. In fact, I noticed that many of the articles collected during the informal search (i.e., the emerging research on software bots) explored a few areas that the other older articles did not. One of the areas that the informal articles covered—but was a major gap in the systematic searched and snowballed articles—were the sub-facets related to *anthropomorphism*. Other examples of facets that were more commonly found in the recent articles include: *robustness*, *language capability*, *interaction cardinality*, and specific *environment types / platforms*.

Lack of Validation: Lastly, all but four of the articles lacked validation for their proposed classifications of software bots; however, the three validation methods were quite limited. Two of the articles validated their classifications by suggesting use cases [55] or scenarios [76]. The other classifications were validated by tagging a small set of software agents and identifying gaps in the existing research [78, 61].

The taxonomy proposed in the thesis was validated through benchmarking, user tagging sessions, and expert tagging of subject matter, as described in Section 6. I also believe that future work is required to more rigorously validate my proposed taxonomy.

7.2 Limitations

In this section, I discuss the limitations and implications of the major design choices I made during the construction of the taxonomy.

Black Box Approach: The taxonomy focuses on classifying the observable properties and behaviours of software bots. I adopted a “*black box*” approach so that software bots could be classified using the taxonomy even if their inner structures were not known. As a result, the taxonomy abstracts many technical and implementation details. The field of software bots continues to grow in popularity, while at the same time new tools, techniques, and algorithms are being developed. Thus, if the proposed taxonomy included these types of details, it would likely be out of date quite quickly. Since I was not able to adequately explore the inner workings of software bots, this is a good avenue for future work.

The taxonomy also focused on the software bot as a whole (i.e., emergent properties [2]), as like other software systems, a bot is more than a sum of its parts. I attempted to define the relationships between facets where possible, however, future work is required to continue to explore the relationships between the various facets of the proposed taxonomy.

Article Selection: The articles used to generate the proposed taxonomy likely influenced both the content and structure of the resulting taxonomy. Although the systematic literature search (cf. Section 4.3.1) provided a solid foundation of academic literature, only articles that had the search terms were included. This may have ruled out many articles that did not provide classifications or focused on less popular subtypes of software bots. The strictness of this selection criteria may also have inadvertently disqualified some articles that could have contributed valuable software bot insights. All of these articles also focused mainly on software agents and agent subtypes. I tried to mitigate these issues by introducing a round of snowballing (cf. Section 4.3.1) and an informal online search (cf. Section 4.3.3).

Even after attempting to balance the articles selected, a majority of them still focused on software agents. As a result, the proposed taxonomy may have a *bias towards agent-like technologies* and may *lack generalizability* to the larger population of software bots. Since many of the articles used to generate the taxonomy were based in theoretical research, some aspects of the proposed taxonomy may also *lack generalizability* to real-life software bots. I attempted to mitigate these risks through tagging a set of publicly available software bots using the proposed taxonomy (cf. Section 6.2), as well as having a domain expert tag their own software bot with the taxonomy (cf. Section 6.3). As future work, comparing the taxonomy against a broader range of software bot technologies would help further validate its generalizability.

Although I provided an inclusion reason for each of the articles I selected during the online search step (cf. Appendix B.4), there were no formal selection criteria for these articles. As a result, I likely also introduced some degree of *selection bias* with respect to the articles used to generate the taxonomy.

Term Extraction: Since I was the only researcher to conduct the term extraction, I may have introduced some *selection bias* with respect to the terms used to generate the taxonomy. I also included only terms that the articles' authors explicitly stated to be a property/behaviour of software bots. Due to this criteria, I may have missed some of the authors' *implied* or more *subtle* dimensions of software bots. I also may have introduced interpretation bias when the authors did not provide a clear description of the extracted property/behaviour.

Term Inclusion: Although the initial card sorting during the taxonomy generation process was completed with a collaborator and the final taxonomy was both run by domain experts and user tested, I performed many of the later steps of the taxonomy generation alone (e.g., drafting descriptions, revising the taxonomy, validation). Ultimately, I selected the dimensions, facets/sub-facets, and values that made it into the taxonomy, which likely

introduced some degree of researcher bias. I tried to mitigate this by taking detailed notes for repeatability/transparency and consulting with both my collaborator and other domain experts throughout the taxonomy generation process. It should also be mentioned that the taxonomy is not a comprehensive list of all possible observable properties/behaviours, but rather a curation of the key facets that I believe are important when trying to understand software bot technologies.

Taxonomy Structure: It is also worth mentioning that many other taxonomy structures could have emerged during the taxonomy generation process. As discussed in Section 4.5.3, multiple variations of high-level dimensions were considered. In the end, I selected three high-level dimensions that provided the most natural feeling division of facets/sub-facets. However, I may have been unintentionally biased by the classification structures proposed in the articles that were used to generate the proposed taxonomy. It is also possible that using different data sources may have created a different taxonomy of software bot properties/behaviours. Similarly, very different taxonomies would have emerged if I had selected an alternate taxonomy structure (e.g., hierarchy or tree).

Validation: Although the taxonomy was validated through benchmarking, researcher tagging, and domain expert tagging, these validation strategies were limited. The benchmarking validation only compared the new taxonomy against the articles that were used to generate it (cf. Section 6.1). Although this was effective in ensuring that important properties/behaviours included in the articles were not overlooked, it would not identify potentially overlooked facets that were not included in the articles. Thus, the taxonomy should be benchmarked against a larger set of articles in future work.

Similarly, the taxonomy should be used to tag a larger population of software bots. The existing validation only tagged three software bots with the proposed taxonomy (cf. Section 6.2). These bots were selected opportunistically (i.e., “cherry picked”) since they all had a different purpose (i.e., a virtual assistant, a weather bot, and a software development bot) and displayed a range of behaviours. However, these bots were also similar in many ways: e.g., they all performed tasks for users, were non-malicious, publicly available, used human language, were platform bots, and were not mobile. Although I was able to successfully classify these three bots on all of the proposed taxonomy dimensions, based on the sample size and selection mechanism, these results cannot be generalized to the entire population of software bots.

Although a domain expert tagging session was held (cf. Section 6.3), it was extremely limited (one participant tagging one bot) and needs to be expanded to cover both a larger set of users and software bots. Due to this, the results of the testing sessions cannot be generalized to the population of potential users or the population of software bots.

Furthermore, the primary focus was on evaluating the utility of the taxonomy (i.e., can it be used to classify a bot) and not the usability of the taxonomy. During the session, the participant relied on the facilitator to help them classify their bot, which suggests that future work in testing and improving the taxonomy’s usability is also required.

I believe that the best way to validate the taxonomy’s generalizability to the larger population of software bots is to use the taxonomy to classify a wider variety of software bots. This may also have some added benefits for the operationalization of the taxonomy, as described below.

7.3 Operationalizing the Taxonomy

Another question readers may have is, how can I go about using the taxonomy presented in this thesis? What future work is required to further operationalize the proposed taxonomy?

In this thesis, I identified three potential user groups for the proposed taxonomy: **researchers, practitioners, end users**. These groups are also not mutually exclusive, as the same user may fall into different categories at different times. However, I distinguish between these three types of users since many of the usage scenarios for each group are distinct. Below, I discuss some ways in which these three groups can use the proposed taxonomy; however, future work is required to validate these assumptions and determine to what extent the existing taxonomy satisfies them.

7.3.1 Researchers

Researchers contribute to the software bot knowledge base by publishing their software bot research. Below, I highlight some key ways in which researchers can use the proposed software bot taxonomy.

Introducing Consistency: The terminology used in the field of software bot research is diverse and often inconsistent. Inconsistent or diverse terminology is also an obstacle to new research. Thus, having a set of consistent terminology in an emerging field makes it easier to build upon each others work and, in turn, speeds up research [1]. In the software bot taxonomy, I propose a set of controlled vocabulary for discussing the observable properties and behaviours of software bots, stemming from past research on software bot-like technologies. This set of preferred terms can be leveraged to help introduce more consistent terminology in future research on software bots.

Understanding Prior Work: By having a mapping between the alternative terminology used in previous research and my proposed taxonomy’s controlled vocabulary, researchers

can better understand past research into software bot technologies. The taxonomy’s mappings can be used as a sort of thesaurus of related bot terms. These mappings can also be used to help understand the ways that different researchers, across multiple research areas are discussing software bots and bot-subtypes.

Helping with Meta Research: The taxonomy can also help researchers better understand the state of research in the domain of software bots as it provides a map of many possible sub-areas of software bot research. Researchers could use the taxonomy to understand where their research fits within and how it connects to other research in the larger domain of software bot research. It can also provide a good starting point for the initial search process. Conducting literature reviews in a specific domain is hard when the terminology is not well defined or standardized—the taxonomy’s controlled vocabulary and mappings can help with this.

Tagging Bots: If a large population of software bots were to be classified using the proposed taxonomy, this would allow researchers to use the taxonomy in a variety of additional ways. First, the taxonomy could be used to identify software bot trends. How have software bot technologies evolved over the years? What are the properties or behaviours of successful bots? What are the properties or behaviours of unsuccessful bots? Can certain facets predict how a bot will be used?

Different role-based or high-level classifications of software bots could also be compared using the proposed taxonomy. For example, on which facets are all chatbots similar and on which facets do they differ? This could help better answer questions like, how to agents differ from other software bots? Or, what properties/behaviours do different types of malicious bots exude, helping us to better detect bad bots. The proposed taxonomy could also help clarify the differences between bot subtypes and other types of non-bot programs.

Identifying Areas for Future Research: Lastly, the proposed taxonomy can also help mature the software bot knowledge area by helping researchers identify existing gaps in the knowledge area and/or new avenues of research.

7.3.2 Practitioners

Practitioners design, develop, and maintain software bot technologies. They contribute to software bot knowledge base through the technical problems they solve and the bots they create. Below, I highlight some key ways in which practitioners can use the proposed software bot taxonomy.

Understanding Possibilities: Whether the practitioners use only a subset or the entire taxonomy, it can provide them with a way to better understand the range of possible properties and behaviours of software bots. Each of the taxonomy dimensions can be viewed as a set of bot design decisions. Practitioners can walk through each of the taxonomies facets/sub-facets in order to make sure they have considered all possible options for their bots properties. Thus, the taxonomy can serve as a sort of check-list of bot design considerations. This helps to get them thinking about the range of software bot properties, and which ones might best address their use cases. The taxonomy also provides practitioners with the vocabulary (both the controlled vocabulary and mapped terms) required to discuss software bots in a research setting.

Analyzing Bots: The proposed taxonomy can also help practitioners glean insights into the bots they have built, by classifying them on the taxonomy. In doing so, the taxonomy can help them reflect on the choices they made, understand the other options available, and determine if they made the right decisions for their bot’s use-case(s). Practitioners can also use the taxonomy to learn more about their bot’s properties or behaviours.

Building Better Bots: I believe that this taxonomy could be used to help practitioners build better bots. Although this taxonomy is currently non-prescriptive, the taxonomy could be extended to include a set of guidelines or best practices for building bot. Since the taxonomy describes the set of possible values for each of the facets or sub-facets, then recommendations could be derived from exploring the difference in the distribution facet values for successful and non-successful bots.

7.3.3 End Users

The last category of potential software bot taxonomy users are the software bot end-users. They look to the software bot knowledge base to increase their understanding, as well as guide their usage and adopting software bots.

Through the taxonomy does not provide any advice on the types of bots to adopt directly, it can help end-users learn more about the technologies they are interacting with and get them thinking about the risks/benefits associated with software bots. Understanding the possible software bots properties and behaviours can help end-user determine what they value in a bot, helping them make more informed decisions regarding the types of bots they welcome into their communication channels, workplaces, and homes.

Chapter 8

Conclusions

This thesis explored, organized, and updated the software bot knowledge area. In this chapter, I conclude the thesis by reflecting on the questions that I set out to answer, pointing out the research contributions that I made along the way, and leave you with some final thoughts on software bots.

8.1 Summary of Research

The work presented in this thesis resulted in three main research contributions. Below, I discuss each of the contributions as well as the research questions they helped answer.

RQ1: What is a software bot? Despite the growing popularity of software bots, there is no clear consensus on what exactly constitutes a bot or the boundaries of bot-hood. In Chapter 3, I addressed this research question by proposing an updated definition of a software bot. I arrived at this definition through exploring and building on many of the existing definitions of software bots from previous literature. I also compared software bots to a variety of other bot-like technologies.

RQ2: What properties of software bots can be used to classify them? The terminology currently used to describe software bots is fragmented and disjointed as it borrows from a variety of research domains. To answer this question, I first explored the range of existing classifications of software bots and discussed any shortcomings of these approaches, see Chapter 3. Since these existing classifications fail to adequately capture the intricacies of modern software bots, I proposed a new taxonomy of the observable properties and behaviours of software bots, as described in Chapter 5. The taxonomy brought together and built upon existing classifications of software bots to create a more holistic view of software bots, see Chapter 4. Lastly, in Chapter 5.1, I discussed how the proposed taxonomy could be used to better understand and classify software bot technologies.

RQ3: What existing terminology used in previous software bot research can be mapped onto the new classification scheme? To answer this question, first I reflected on the variety of terms that previous researchers have used to describe the observable properties and behaviours of software bots, see Chapters 4. Next, I mapped the various terms used to describe equivalent concepts to the controlled vocabulary in the proposed taxonomy. A copy of the term mappings is provided in Appendix E. Lastly, in Chapter 5.1, I discussed how these mappings may be used to better understand software bot technologies.

8.2 Final Remarks

Software bots are complex—and growing more so each day. For many years people have struggled to define exactly what makes a bot, a “*bot*”, which has often led them to define bots according to their own specific needs. Software bots are also a digital “*Frankenstein*” of research from a variety of different fields, both in academia and industry. Many of these different fields bring with them their own set of terminology for describing bots. I believe that the research presented in thesis serves to further highlight how complex software bots are and how much we still have to learn about them.

In this thesis, I explored the bot knowledge area, both historic and theoretic concepts and definitions, and generated a comprehensive and modern taxonomy of bots. Although this work provides an initial step towards a deeper understanding of modern software bot technologies, much work remains. It is my hope, however, that this research can assist people researching, building, and using software bots.

Appendices

Appendix A

Collection Queries

The following queries were ran the first week of June 2017, and included anything published prior to that date.

A.1 ACM Digital Library

- *acmdlTitle:(+taxonom* +agent*) or acmdlTitle:(+classif* +agent*) or acmdlTitle:(+character* +agent*)*
- *acmdlTitle:(+taxonom* +bot) or acmdlTitle:(+classif* +bot) or acmdlTitle:(+character* +bot)*
- *acmdlTitle:(+taxonom* +bots) or acmdlTitle:(+classif* +bots) or acmdlTitle:(+character* +bots)*
- *acmdlTitle:(+taxonom* +chatbot*) or acmdlTitle:(+classif* +chatbot*) or acmdlTitle:(+character* +chatbot*)*
- *acmdlTitle:(+taxonom* +chatterbot*) or acmdlTitle:(+classif* +chatterbot*) or acmdlTitle:(+character* +chatterbot*)*

A.2 IEEE Xplore

- *("Document Title":taxonom* OR "Document Title":classif* OR "Document Title":character*) AND ("Document Title":bot OR "Document Title":bots OR "Document Title":chatbot* OR "Document Title":chatterbot* OR "Document Title":agent*)*

A.3 ScienceDirect (Computer Science section)

- *TITLE-ABSTR-KEY(taxonom* or classif* or character*) and TITLE(agent* or bot or bots or chatbot* or chatterbot*)[Journals(Computer Science)]*

A.4 SpringerLink

- *Searching for taxonom* OR classif* OR charact* in Article Titles AND agent* OR bot OR bots OR chatbot* OR chatterbot* in Article Title*

A.5 Wiley Online (Computer Science section)

- *taxonom* OR classif* OR charact* in Article Titles AND agent* OR bot OR bots OR chatbot* OR chatterbot* in Article Title*

Appendix B

Data Extracted from Article Selection

B.1 Excluded Systematic Search Articles

Ref	Date	Authors	Title	Venue	Reason
[91]	2008	Schmeck and Müller-Schloer	A Characterization Of Key Properties Of Environment-Mediated Multi-agent Systems	workshop paper	E4: focuses on specific subtype of subject: Multi-agent systems since it focuses on the interactions of multiple agents and not single agents
[92]	1995	Aitken, Schmalhofer, and Shadbolt	A Knowledge Level Characterization Of Multi-Agent Systems	workshop paper	I3: doesnt try to provide a schema for subject classification
[93]	2008	Ko, Han, Kim, and Youn	A New Agent Characterization Model And Grouping Method For Multi-Agent System	conference paper	E2: this paper focuses on the algorithmic classification of agents, E4: this paper focuses on the bots in multi-agent systems
[94]	2007	Dagon, Gu, Lee, and Lee	A Taxonomy Of Botnet Structures	conference paper	E4: focuses on botnet structures (network focus)
[95]	2000	Wong and Sycara	A Taxonomy Of Middle-Agents For The Internet	conference paper	E4: middle agents in multi-agent systems (specifically: matchmakers and facilitators)
[96]	2013	van Oijen and Dignum	Agent Communication For Believable Human-Like Interactions Between Virtual Characters	workshop paper	I3: doesnt examine a classification schema, I1: doesnt contain search keywords
[97]	2009	Afonso and Prada	Agents That Relate: Improving The Social Believability Of Non-Player Characters In Role-Playing Games	conference paper	I3: doesnt examine a classification schema, I1: doesnt contain search keywords
[98]	2004	Oluyomi, Karunasekera, and Sterling	An Agent Design Pattern Classification Scheme: Capturing The Notions Of Agency In Agent Design Patterns	conference paper	I4: focuses on the implementation design patterns for bots, not functional / non-functional properties

[99]	2005	Riedl and Young	An Objective Character Believability Evaluation Procedure For Multi-Agent Story Generation Systems	conference paper	I3: doesnt examine a classification schema, I1: doesnt contain search keywords
[64]	1999	Yap and Keong	Are Life-Like Characteristics Useful For Autonomous Agents?	conference paper	I3: doesnt examine a classification schema, I1: doesnt contain search keywords
[100]	2002	Doyle	Believability Through Context Using Knowledge In The World To Create Intelligent Characters	conference paper	I3: doesnt examine a classification schema, I1: doesnt contain search keywords, I4: focuses on the implementation of bots, not functional / non-functional properties
[101]	2012	Eslahi, Salleh, and Anuar	Bots And Botnets: An Overview Of Characteristics, Detection And Challenges	conference paper	E4: focuses on botnet structures (network focus)
[102]	2006	Cascalho, Antunes, Corrêa, and Coelho	Characterising Agents' Behaviours: Selecting Goal Strategies Based On Attributes	workshop paper	I3: focuses on bots from an engineering perspective and proposes a framework to alter bots behaviours
[103]	2013	Sumi and Nagata	Characteristics Of Robots And Virtual Agents As A Persuasive Talker	conference paper	I3: focuses on evaluating the effect of agents using verbal / facial expression
[104]	2005	Davidsson, Johansson, and Svahnberg	Characterization And Evaluation Of Multi-Agent System Architectural Styles	workshop paper	I3: focuses on evaluating different multi agent systems architectures, not on the properties themselves, E3: focuses on properties of MAS as a whole
[105]	2007	Stinson and Mitchell	Characterizing Bot's Remote Control Behavior	conference paper	E1: focuses on evaluating remote-control behaviours of bots, E3: focuses on botnets and malicious bots
[106]	2005	Hu and Liu	Characterizing Complex Behavior In (Self-Organizing) Multi-Agent Systems	conference paper	I3: focus on evaluating types of MAS, not of characterizing properties of bots, E3: focus on MAS as a whole
[107]	2012	den Bosch, Brandenburgh, Muller, and Heuvelink	Characters With Personality!	conference paper	I3: evaluates the effects of personality on virtual agents, does not provide a classification, E3: focused on virtual agents in games
[108]	2009	Hayashi and Miwa	Cognitive And Emotional Characteristics Of Communication In Human-Human/Human-Agent Interaction	conference paper	I3: examines the characteristics of human-agents interactions, not the characteristics of agents themselves violates, I4, E1
[109]	2009	Prasad, Kumar, and Raju	Comparison Shopping Agents: The Essential Characteristics And Challenges To Be Met	conference paper	I3: evaluates characteristics, doesnt provide a classification, E4: focuses only on comparison shopping bots
[110]	1995	Darley	Constructive And Destructive Obedience: A Taxonomy Of Principal-Agent Relationships	conference paper	I3: not focused on software agents, E6: not found online
[111]	2011	Smajgl, Brown, Valbuena, and Huigen	Empirical Characterisation Of Agent Behaviours In Socio-Ecological Systems	journal paper	I2: focusing on modeling human behaviours with agents, not software bots themselves, I3: doesnt try to characterize functionality, I4: doesn't focus on the properties of an emerging system, E3: focuses on the application of bots in other domains

[112]	2007	Zoric, Smid, and Pandzic	Facial Gestures: Taxonomy And Application Of Non-verbal, Non-emotional Facial Displays For Embodied Conversational Agents	academic book	E4: focuses only on classifying the facial expressions / gestures of embodied conversational agents
[113]	2016	Merrick	Game-Playing Agents And Non-Player Characters	academic book	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bots properties
[114]	2009	DiPaola	Intelligent Expression-Based Character Agent Systems	workshop paper	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bots properties
[115]	2002	Cavazza, Charles, and Mead	Interacting With Virtual Characters In Interactive Storytelling	conference paper	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bots properties
[116]	2005	Ito	Introducing Multimodal Character Agents Into Existing Web Applications	conference paper	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bot properties
[117]	1996	Tosa and Nakatsu	Life-Like Communication Agent-Emotion Sensing Character And Feeling Session Character	journal paper	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bot properties, instead focuses on a proposed bot properties
[118]	2003	Mukhopadhyay, Peng, Raje, Palakal, and Mostafa	Multi-Agent Information Classification Using Dynamic Acquaintance Lists	journal paper	I3: does not provide a classification of bot properties, instead focuses on a proposed bots properties, E2: focuses on the algorithmic classification of agents
[119]	2002	Kitamura, Tsujimoto, Yamada, and Yamamoto	Multiple Character-Agents Interface: An Information Integration Platform Where Multiple Agents And Human User Collaborate	conference paper	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bot properties, instead focuses on a proposed bots properties
[120]	2001	Jones and Firozabadi	On The Characterisation Of A Trusting Agent – Aspects Of A Formal Approach	academic book	I3: does not provide classification of bot properties
[121]	2003	Arafa and Mamdani	Scripting Embodied Agents Behaviour With Cml: Character Markup Language	conference paper	I3: does not provide classification of bot properties, instead the properties of languages for building bots
[122]	1999	Takeuchi and Katagiri	Social Character Design For Animated Agents	workshop paper	I3: does not provide classification of bot properties, instead compares HCI aspects of bot properties
[123]	2009	Neto and da Silva	Synthetic Characters With Personality And Emotion	workshop paper	I3: does not provide classification of bot properties, instead proposes an architecture for building bots
[124]	2011	Cabri, Puviani, and Zambonelli	Towards A Taxonomy Of Adaptive Agent-Based Collaboration Patterns For Autonomic Service Ensembles	conference paper	I3: does not provide classification of bot properties, instead proposes a taxonomy for bot building/communication patterns
[125]	2009	Damiano and Lombardo	Using Values To Turn Agents Into Characters	conference paper	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bot properties

[126]	2008	Ke and Mostafa	Visualizing Multi-Agent Collaboration For Classification Of Information	journal paper	I3: does not provide classification of bot properties, instead visualizes communication patterns between agents
[127]	2004	Kitamura	Web Information Integration Using Multiple Character Agents	academic book	I1: does not contain desired definition of keywords in title, I3: does not provide classification of bot properties, instead proposed a multi-agent platform
[128]	2014	Baraniuk	You've Got Personality... For A Bot	academic magazine	I1: does not contain keywords in title, I3: does not provide classification of bot properties
[129]	2004	Hare and Deadman	Further towards a taxonomy of agent-based simulation models in environmental management	journal paper	E4: focuses on agent simulations
[130]	2010	Read, Talevich, Walsh, Chopra, and Iyer	A Comprehensive Taxonomy Of Human Motives: A Principled Basis For The Motives Of Intelligent Agents	conference paper	I2: does not provide classification software bots, but instead intelligent agents of a non-software variety
[131]	2003	Höppner	An Agents Definition Framework And A Methodology For Deriving Agents Taxonomies	conference paper	E5: the full article is not available online
[132]	2006	Filatov	About One Approach To The Classification Of Program Agents	conference paper	E2: focuses on the algorithmic classification of agents
[133]	1997	Jennings and Campos	Towards A Social Level Characterisation Of Socially Responsible Agents	academic magazine	I3: doesnt examine a classification schema, instead presents a framework for agent architecture
[131]	2003	Höppner	An Agents Definition Framework And A Methodology For Deriving Agents Taxonomies	Conference	I3: doesnt examine a classification schema, instead develops an algorithm to classify software agents based on other definitions

B.2 Included Systematic Search Articles

Ref	Date	Authors	Title	Schema	Subject	Scope	Methodology	Validation
[72]	1993	Bird	Toward A Taxonomy Of Multi-Agent Systems	taxonomy	agents (multi-agent system)	proposing new (based on previous work)	expert opinion	none
[64]	1999	Yap and Keong	Are Life-Like Characteristics Useful For Autonomous Agents?	characterization	agents (autonomous agents)	Proposing new	expert opinion	none
[86]	2003	Parunak, Brueckner, Fleischer, and Odell	A Design Taxonomy Of Multi-Agent Interactions	taxonomy	agents (multi-agent)	extending existing model	expert opinion	none
[65]	2005	Hector and Narasimhan	A New Classification Scheme For Software Agents	classification scheme / ontology	software agents	extending existing model	expert opinion	none
[66]	2007	Moya and Tolk	Towards A Taxonomy Of Agents And Multi-Agent Systems	taxonomy	Agents and Multi-Agent Systems	proposing new (based on others)	expert opinion	none
[71]	2008	van Otterlo, Dastani, Wiering, and Meyer	A Characterization Of Sapient Agents	characterization	sapient & cognitive Agents	proposing new	expert opinion	none
[88]	2009	Sakarkar and Shelke	A New Classification Acheme For Autonomous Software Agent	classification	agents (autonomous agents)	proposing new (based on others)	expert opinion	none
[43]	2009	Tolk and Uhrmacher	Agents: Agenthood, Agent Architectures, And Agent Taxonomies	taxonomy / classification	agents	extending existing model	expert opinion	none
[76]	2012	Aguero, Rebollo, Carrascosa, and Julian	Agent Capability Taxonomy For Dynamic Environments	taxonomy	agents	extending existing model	expert opinion	(utility) limited scenarios
[61]	1997	Sánchez	A Taxonomy Of Agents	taxonomy	agents	extending existing model	expert opinion	none
[86]	2003	Parunak, Brueckner, Fleischer, and Odell	A Preliminary Taxonomy Of Multi-Agent Interactions	taxonomy	agents (multi-agent system)	proposing new	expert opinion	none
[78]	2000	Huang, Eliens, van Ballegooij, and de Bra	A Taxonomy Of Web Agents	taxonomy	agents (web agents)	proposing new	expert opinion	(utility) limited agent tagging, identifying research gaps
[77]	2003	Munroe and Luck	Agent Autonomy Through The 3M Motivational Taxonomy	taxonomy	agents	proposing new	expert opinion	none

[55]	2004	Tosic and Agha	Towards A Hierarchical Taxonomy Of Autonomous Agents	taxonomy	agents (autonomous agents)	proposing new (based on previous work)	expert opinion	(utility) suggested use case scenarios
[69]	1996	Franklin and Graesser	Is It An Agent, Or Just A Program?: A Taxonomy For Autonomous Agents	Taxonomy	agents (autonomous agents)	proposing new (based on previous work)	expert opinion	none

B.3 Included Snowballed Articles

Ref	Date	Authors	Title	Schema	Subject	Scope	Methodology	Validation
[57]	1996	Nwana	Software Agents: An Overview	topology	software agents	proposing new	expert opinion	identified examples of bots for each category
[89]	1994	Kautz, Selman, Coen, Ketchpel, and Ramming	An Experiment in the Design of Software Agents	list: distinguishing properties	software agents	proposing new	list: “distinguishing features” & interaction framework	no evaluation for list
[80]	1998	Huhns and Singh	Agents and Multiagent Systems: Themes, Approaches and Challenges.	taxonomy	agents and multi-agent systems	proposing new	expert opinion	none
[85]	1994	Etzioni and Weld	A Softbot-Based Interface to the Internet	list: design principles	Agents	proposing new	inspired by an existing agent	none
[53]	1993	Foner	What’s An Agent, Anyway?	list: “crucial notions”	Agents	proposing new	inspired by an existing agent	none
[52]	1994	Wooldridge and Jennings	Agent Theories, Architectures, and Languages: a Survey	list: attributes of agency	Agents	based on existing definitions	expert opinion	none
[5]	1995	Russell and Norvig	Artificial Intelligence: A Modern Approach (Ch 2)	list: describes properties	intelligent agents	n/a	expert opinion	none
[134]	1995	Goodwin	Formalizing Properties of Agents	framework	agents (virtual/-physical)	proposing new	expert opinion	none

B.4 Included Online Articles

Ref	Date	Authors	Title	Venue	Subject	Inclusion Reason
[39]	2017	Shevat	Designing Bots: Creating Conversational Experiences	book	chatbots	provides a comprehensive look at the development of modern bots and their various properties
[50]	2016	Roy	What Bots May Come: A Learning Architecture for the Next Paradigm	Chatbot Magazine (online)	Chatbots	provides an updated collection of chatbot attributes (with a focus on learning)
[38]	2016	Dale	Industry Watch The return of the chatbots	Journal	chatbots	provides an overview of the existing bot landscape, reasons for bot's re-emergence, and where bot technologies are heading in the future
[135]	2016	Fisher	The Seven Habits of Highly Effective Chatbots: Lessons learned to help super charge your bot.	Chatbot Magazine	chatbots	provides a set of best practices for building chatbots and discusses many themes of modern software bots
[136]	2016	Grover	Bots won't replace apps. Better apps will replace apps.	blog	chatbots	provides a more pessimistic view on software bots, provides some background on the evolution of bots in conversational platforms, and discusses many themes of modern software bots
[137]	2015	Libov	Futures of text: A survey of all the current innovation in text as a medium	blog	agents/bots	explores some existing properties of modern bots and discusses what they want from bots going forward
[75]	2016	Block	How to build a better bot	Chatbot magazine	bots	provides a set of guidelines for building better bots, discusses bot conversational abilities, and highlights some properties of bots
[138]	2016	MacPherson	How to interact with bots? Dealing with the complexity of a new design paradigm	Chatbot Magazine	bots	discusses how different behaviours of modern software bots can be used to create a feeling of authority and expertise in humans interactions
[139]	2015	Clément and Guitton	Interacting with bots online: Users reactions to actions of automated programs in Wikipedia	journal	bots	discusses the current roles that bots can play on Wikipedia, and the perceptions humans have of them
[140]	2000	Jurafsky and Martin	Dialog Systems and Chatbots	book chapter	chatbots	provides a detailed description of how chatbots can interact via dialogue

[46]	2017	Bot Nerds	Types of Bots: An Overview	website	bots	provides a great overview of the various types/properties of modern software bots
[45]	2016	Storey and Zagalsky	Disrupting Developer Productivity One Bot at a Time	Conference	bots	provides a modern look at the properties of software bots, how software developers are using bots, and introduces a preliminary productivity framework for evaluating/studying software bots
[141]	2017	Storey	To Bot or Not: How Bots can Support Collaboration in Software Engineering	Keynote	bots	provides a modern look at the properties of software bots, how software developers are using bots, and introduces a preliminary productivity framework for evaluating/studying software bots
[83]	2017	Long, Vines, Sutton, Brooker, Feltwell, Kirman, Barnett, and Lawson	“Could You Define That in Bot Terms”?: Requesting, Creating and Using Bots on Reddit	conference	bots	presents a study of how bot are being discussed and created on Reddit, their roles, and the actions they can perform
[73]	2014	Schenk	Bot Design Patterns: different ways to make different bots	blog	bots	discusses challenges when building chatbots

Appendix C

Excluded Cards

Ref	Card	Exclusion Reason
[84]	communication: mutual adjustment, informal communication among workers “conversation” (direct)	too specific to agent systems, does not make sense for single agents
[84]	communication: standardization of work processes (e.g., setting up a work station on an assembly line) (indirect)	too specific to agent systems, does not make sense for single agents
[84]	communication: standardization of outputs / input (indirect)	too specific to agent systems, does not make sense for single agents
[84]	communication: standardization of skills and knowledge (direct)	too specific to agent systems, does not make sense for single agents
[66]	competency (capability to effectively manipulate the problem domain environment to accomplish the prerequisite tasks)	too high level of a concept (effectively + complete tasks)
[78]	environment: 2D environment vs 3D environment	too much detail about the environment
[136]	snap commands	extraction error, should have been “slash” commands
[69]	control via language (in which written)	implementation related details
[61]	location of behaviour or architecture: monolithic (middleman between user and database), embedded, or independent. Where the data is located in reference to the user.	multi-agent system and implementation specific detail

Appendix D

Restructuring the Taxonomy

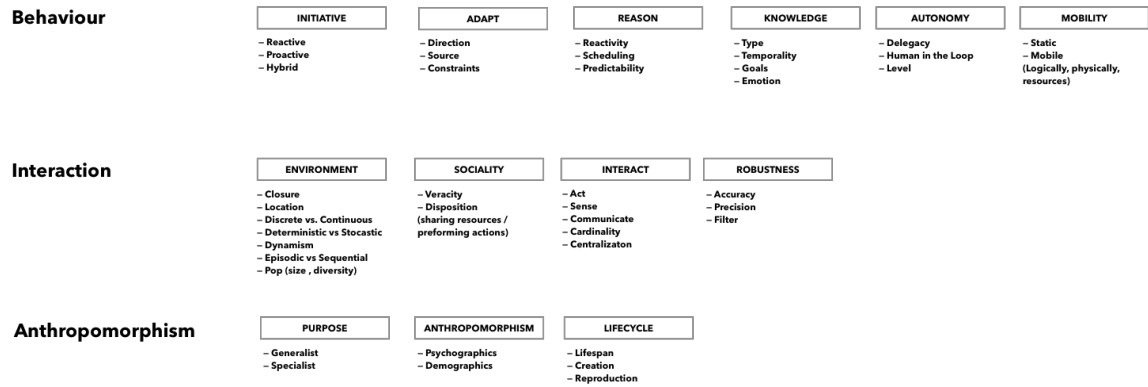


Figure D.1: Version #1 of the re-ordering of dimensions, facets, and sub-facets.

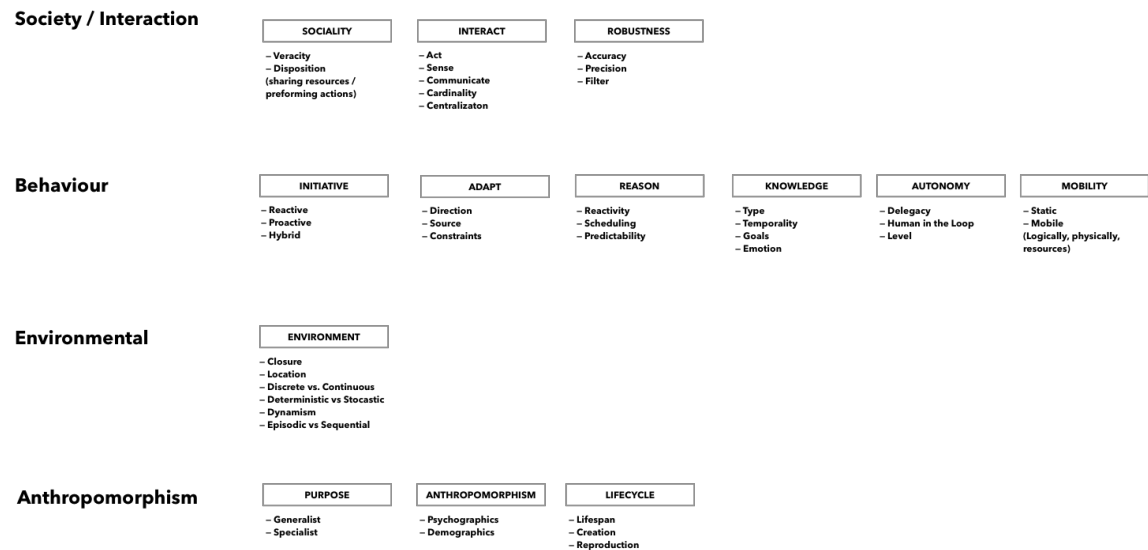


Figure D.2: Version #2 of the re-ordering of dimensions, facets, and sub-facets.

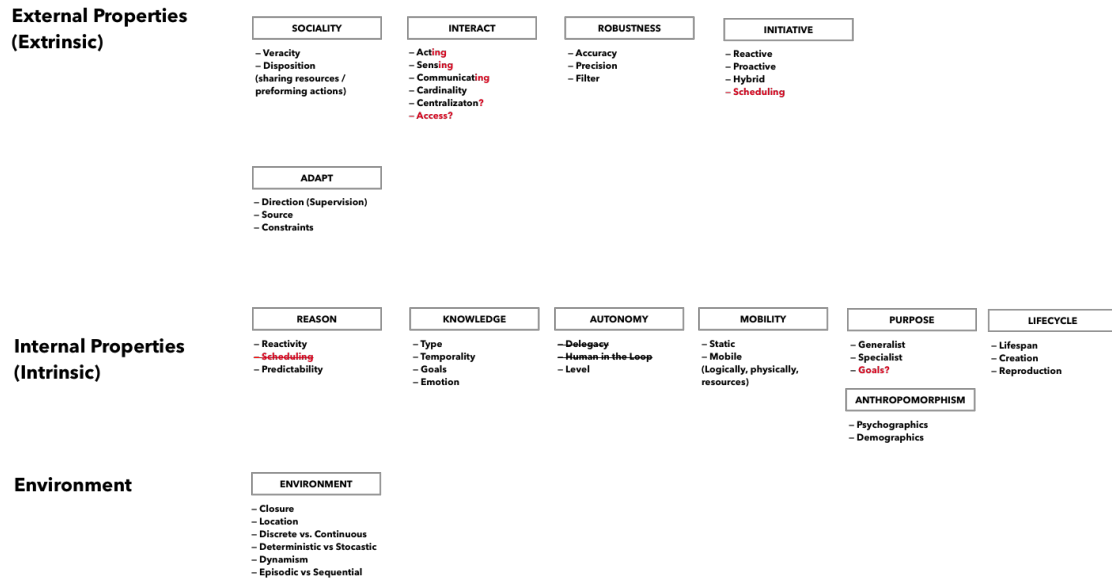


Figure D.3: Version #3 of the re-ordering of dimensions, facets, and sub-facets.

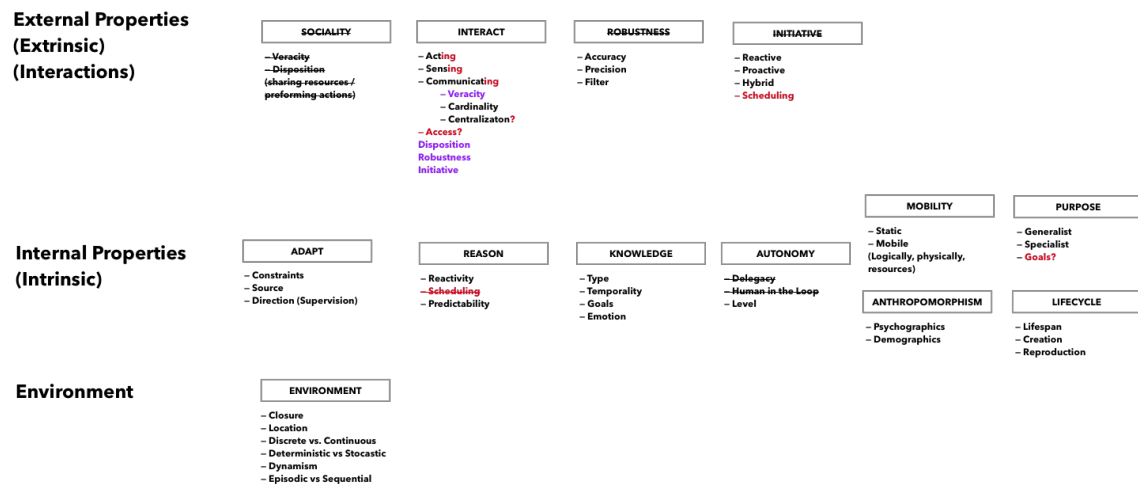


Figure D.4: Version #4 of the re-ordering of dimensions, facets, and sub-facets.

Appendix E

Mapped Terminology

Tables E.1, E.2, & E.3 present a high-level summary of the outcome of the terminology mapping effort for the **environment**, **intrinsic**, and **interaction** dimensions, respectively. A list of all previous literature used to generate the mappings can be found in Appendices B.2, B.3, & B.4.

E.1 Environment Dimension

Summary of the mappings between the terminology used in the proposed taxonomy and previous software bots research for the environment facets.

Environment Dimension		
Section	Preferred Term	Mapped Terms
5.2.1	Type	
5.2.1(i)	Standalone	multi-platform [89], augmenting architectures [61]
5.2.1(ii)	Platform	platforms [141], integrating architectures [61] voice platforms [39], consumer platforms [39], business platforms [39], legacy platforms [39]
5.2.2	Scope	
5.2.2(i)	Bounded	discrete [5, 43]
5.2.2(ii)	Unbounded	continuous [5, 43]
5.2.3	Closure	closure [66]
5.2.3(i)	Closed	closed [66, 43]
5.2.3(ii)	Open	open [81, 66, 43]
5.2.4	Dynamism	
5.2.4(i)	Static	static [5, 66, 43]
5.2.4(ii)	Dynamic	dynamic [5, 66, 43], real-time [80]
5.2.5	Predictability	predictable [80], omniscience [5], deterministic [5]
5.2.5(i)	Stochastic	stochastic [69, 43], non-deterministic [5]
5.2.5(ii)	Uncertain	
5.2.5(iii)	Deterministic	deterministic [5, 69, 43]
5.2.6	Permanence	

5.2.6(i)	Episodic	episodic [5, 43]
5.2.6(ii)	Sequential	sequential [43], historical [80], non-episodic [5]
5.2.7	Population	population [66, 43]
5.2.7.1	Cardinality	cardinality [66], size [66]
5.2.7.1(i)	Singular	
5.2.7.1(ii)	Countable	countable [66], few [43]
5.2.7.1(iii)	Uncountable	uncountable [66], large scale [43]
5.2.7.2	Diversity	diversity [66], uniqueness [80], composition [72]
5.2.7.2(i)	Homogeneous	homogeneous [72, 66, 43]
5.2.7.2(ii)	Heterogeneous	heterogeneous [72, 81, 66, 43]

E.2 Intrinsic Dimension

Summary of the mappings between the terminology used in the proposed taxonomy and previous software bots research for the intrinsic facets.

Intrinsic Dimension		
Section	Preferred Term	Mapped Terms
5.3.1	Knowledge	knowledge [52], belief [52, 55, 71], modeling [80], internal state [5, 55], mental states [55]
5.3.1.1	Memory	memory [53, 45, 39],
5.3.1.1(i)	Long-term	historical events [66], past actions (self/others) [66], action effects (self/others) [66]
5.3.1.1(ii)	Short-termepisod	episodic [50], contextual [50, 39]
5.3.1.1(iii)	Future	
5.3.4.5	Source	
5.3.4.5(i)	Encoded	encoded [61]
5.3.4.5(ii)	Supplied	supplied [61]
5.3.4.5(iii)	Learned	learned [61]
5.3.2	Reasoning	reasoning [66], proactiveness [65], intelligence [46]
5.3.2.1	Mechanisms	controller [134], action selection mechanism [69], control process [77], reasoning architecture [66], decision making [43], deliberation [71]
5.3.2.1(i)	Scripted	script bots [46], reactive [80], pure reaction [65], reflex [5], non-planning [69], tropistic [66], simple rules [141], rule-based [140]
5.3.2.1(ii)	Mixed	hybrid [65, 66], smart bots [46], corpus-based [140]
5.3.2.1(iii)	Planning	planning [69], pure planning [65], deliberative [80, 134], flexible [69], declarative [66], intelligent [141, 46]
5.3.2.2	Agency	execution autonomy [80]
5.3.2.2(i)	None	controlled [80]
5.3.2.2(ii)	Veto	
5.3.2.2(iii)	Complete	independent [80]
5.3.2.3	Predictability	

5.3.2.3(i)	Stochastic	
5.3.2.3(ii)	Deterministic	
5.3.2.4	Visibility	
5.3.2.4(i)	None	
5.3.2.4(ii)	Transparent	
5.3.2.4(iii)	Visible	
5.3.2.5	Reactivity	responsiveness [64, 39], response time [138]
5.3.2.5(i)	Synchronous	synchronous [75], reflexive [134], reactive [69, 55], responsiveness [55]
5.3.2.5(ii)	Mixed	
5.3.2.5(iii)	Asynchronous	asynchronous [137]
5.3.2.6	Scheduling	
5.3.2.6(i)	Single	exclusive [76], non-interrupting [76], interrupting [76]
5.3.2.6(ii)	Multiple	multiple [76]
5.3.3	Adaptability	adaptive [69, 55, 43, 71], adaptiveness [65], learning [57, 53, 69, 71], non-learning [69], fixed [80], growth [64], non-adaptive [65]
5.3.3.1	Constraints	
5.3.3.1(i)	Constrained	constraint based [65]
5.3.3.1(ii)	Open	
5.3.4.5	Source	
5.3.4.5(i)	Internal	
5.3.4.5(ii)	External	social learning [50]
5.3.3.3	Guidance	
5.3.3.3(i)	Undirected	autodidactic [80], learning [65]
5.3.3.3(ii)	Directed	teachable [80], personalizable [53], personalization [45] subscription [65], reinforcement [71]
5.3.4	Goals	goals [64, 71], goal oriented [89, 69, 55], goal directedness [77, 55] task [134, 76], drive [69], pro-active [69], purposeful [69], desire [55], capability[76]
5.3.4.1	Complexity	risk [77]
5.3.4.1(i)	Low	
5.3.4.1(ii)	High	
5.3.4.2	Criticality	domain [53]
5.3.4.2(i)	Low	
5.3.4.2(ii)	High	
5.3.4.3	Attainability	
5.3.4.3(i)	Achievable	achievable [134], binary-task [134]
5.3.4.3(ii)	Homoeostasis	maintenance-task [134]
5.3.4.4	Explicitness	
5.3.4.4(i)	Explicit	explicit [66], ground goals [89], procedural [80]
5.3.4.4(ii)	Implicit	implicit [66], charitable [89], incomplete specification [89], declarative [80]
5.3.4.5	Source	

5.3.4.5(i)	Internal	internal trigger [39]
5.3.4.5(ii)	External	external trigger [39]
5.3.5	Delegacy	delegacy [61, 66], delegate [57, 53]
5.3.5(i)	None	
5.3.5(ii)	Partial	
5.3.5(iii)	Complete	
5.3.6	Specialization	
5.3.6(i)	Generalist	generalist [46], versatile [57], super bot [39]
5.3.6(ii)	Specialist	specialist [46], domain specific bot [39]
5.3.7	Anthropomorphism	anthropomorphism [53], lifelike character [61]
5.3.7.1	Name	
5.3.7.1(i)	None	
5.3.7.1(ii)	Representative	brand name [39], functional name [39]
5.3.7.1(iii)	Unique	human name [39]
5.3.7.2	Embodiment	
5.3.7.2(i)	None	
5.3.7.2(ii)	Embodied	2D [78], 3D [78], logos [39], icons, [39]
5.3.7.3	Age	age [39]
5.3.7.3(i)	None	
5.3.7.3(ii)	Static	
5.3.7.3(i)	Dynamic	
5.3.7.4	Gender	gender [141, 39]
5.3.7.4(i)	None	
5.3.7.4(ii)	Gendered	
5.3.7.5	Ethnicity	
5.3.7.5(i)	None	
5.3.7.5(ii)	Ethnicity	
5.3.7.6	Profession	
5.3.7.6(i)	None	
5.3.7.6(ii)	Profession	
5.3.7.7	Personality	personality [141, 50, 138, 39]
5.3.7.7(i)	None	lifeless [135]
5.3.7.7(ii)	Personality	character [69], persona [137]
5.3.7.8	Emotions	emotional [52, 50], sentiment [39]
5.3.7.8(i)	None	
5.3.7.8(ii)	Superficial	
5.3.7.8(iii)	Logical	emotional attitude [57, 71]
5.3.8	Lifecycle	
5.3.8.1	Creation	
5.3.8.1(i)	Human	
5.3.8.1(ii)	Bot	
5.3.8.1(iii)	System	
5.3.8.2	Lifespan	lifespan [80], age [64]
5.3.8.2(i)	Terminating	death [64]

5.3.8.2(ii)	Transient	transient [80]
5.3.8.2(iii)	Continuous	temporally continuous [57, 89, 69], long-lived [80], persistent [55]
5.3.8.3	Reproduction	reproduction [64]
5.3.8.3(i)	None	
5.3.8.3(ii)	Reproductive	

E.3 Interaction Dimension

Summary of the mappings between the terminology used in the proposed taxonomy and previous software bots research for the interaction facets.

Interaction Dimension		
Section	Preferred Term	Mapped Terms
5.4.1	Access	accessible [5]
5.4.1(i)	None	inaccessible [5], nonaccessible [43]
5.4.1(ii)	Constrained	partial [66]
5.4.1(iii)	Complete	complete [66], accessible [5, 43]
5.4.2	Sense	sense [69], knowable environment [80], percept [134], perception [43], perceiving [5], sensors [55], events [76], inputs [75]
5.4.2(i)	Non-Sensing	
5.4.2(ii)	Sensing	
5.4.3	Act	acting [5, 69], action [43, 71], controllable environment [80], affect [134], effectors [55], behaviour [76], capability [76], output [75]
5.4.3(i)	Non-Acting	
5.4.3(ii)	Acting	
5.4.4	Communicate	communication [89, 43], communicative [69, 84, 81, 65], social ability [57, 52, 69, 77, 66], non-communicative [69, 65], singular vs multiplicity [78]
5.4.4.1	Disposition	socialness [64], sociality [77, 55], cooperativity [66]
5.4.4.1(i)	Antagonistic	antagonistic [57, 80, 81], Contention [84, 81], malevolent [65]
5.4.4.1(ii)	Competitive	competitive [80, 84, 81, 66, 43, 71], selfish [77], self-interested [65]
5.4.4.1(iii)	Indifferent	non-helpful [57], independent [66], autistic [80]
5.4.4.1(iv)	Cooperative	cooperative [66, 43, 71], cooperate [57, 80, 77], cooperation [53, 86, 84], collaborate [64, 65, 88], coordinate [84, 81]
5.4.4.1(v)	Benevolent	benevolent [57, 80, 65], altruistic [57]
5.4.4.2	Veracity	veracity [52, 57]
5.4.4.2(i)	Truthful	truthful [57, 65]
5.4.4.2(ii)	Untruthful	untruthful [65], lies [57]

5.4.4.3	Cardinality	
5.4.4.3(i)	One-One	1:1 [137], personal agent [61], personal bot [39], private channel [39]
5.4.4.3(ii)	One-Many	1:N [137], public channel [39]
5.4.4.3(iii)	Many-Many	M:N [137], group agents [61], team bot [39]
5.4.4.4	Directionality	
5.4.4.4(i)	One-Way	
5.4.4.4(ii)	Two-Way	two-way [89], discourse [53], conversation [84, 81], turns [140], respond [73]
5.4.4.5	Directness	
5.4.4.5(i)	Direct	direct [84, 81]
5.4.4.5(ii)	Indirect	indirect [84, 81]
5.4.4.6	Language	conversation management [39]
5.4.4.6(i)	None	
5.4.4.6(ii)	Keywords	specific vocabulary [80], universal commands [140], restrictive grammar [140], template generation [140], prompts [140], slash commands [39]
5.4.4.6(i)	Natural Language	language [80, 39]
5.4.4.6(i)	Conversation	conversation [140, 73, 39], dialog [140]
5.4.5	Initiative	initiative [140], invocation [39]
5.4.5(i)	Reactive	reactivity [52, 77, 88], perceptive [134], pull [45], react [73], user lead invocation [39]
5.4.5(ii)	Proactive	proactive [39], proactiveness [57, 52, 78, 77, 55], system-initiative [140], push [45], notify [73, 39], prediction [39]
5.4.6	Robustness	robustness [89], degrade gracefully [57, 53], accuracy [66], error handling [39]
5.4.6.1	Error Prevention	
5.4.6.1(i)	Bot	
5.4.6.1(ii)	User	user help [39]
5.4.6.2	Error Discovery	
5.4.6.2(i)	Bot	implicit confirmation [39], explicit confirmation [39]
5.4.6.2(ii)	User	
5.4.6.3	Error Correction	
5.4.6.3(i)	Bot	
5.4.6.3(ii)	User	human intervention [39]
5.4.7	Mobility	mobility [55, 65]
5.4.7(i)	Static	static [57, 65], stationary [80], non-mobile [69]
5.4.7(ii)	Mobile	mobile [57, 52, 69, 88], itinerant [80], move [64], physically mobile [65], logically mobile [65]

Appendix F

Benchmarking Validation

Below, I discuss how the many articles used to build my proposed taxonomy compare to it. The articles are organized by their data collection method and presented in chronological order.

F.1 Literature Search Articles:

(1993) Bird [72]: proposes a taxonomy of multi-agent systems which describes the properties of agents in these systems by extrapolating a set of general characteristics from three primary agent dimensions: distribution (of knowledge, control, operation), heterogeneity, and autonomy. Although some of the article’s dimensions were out of the scope of the new taxonomy’s subject matter (e.g., specific to multi-agent systems or describing implementation details), the following properties were mapped/merged to the new taxonomy: autonomy, composition of agents, and scope.

(1996) Franklin and Graesser [69]: explores existing definitions of agent-hood, provides a classification of the properties of autonomous agents, and rule-based classification of autonomous agents. In their classification of properties, they include four high-level properties that all autonomous agents must have (reactive, autonomous, goal-oriented, and temporally continuous) and five useful additions (communicative, learning, mobile, flexible, character). My proposed taxonomy covers these properties.

(1997) Sánchez [61]: describes three sub-types of agents: programmer, network, and user agents. User agents, which closely resemble the definition of software bots provided in Chapter 3, are characterized by their autonomy, scope, architecture, and knowledge. My proposed taxonomy covers all of their properties of user agents.

(1999) Yap and Keong [64]: proposes and defines a set of life-like characteristics for autonomous agents (e.g., growth, death, reproduction, etc). My proposed taxonomy covers these properties.

(2000) Huang et al. [78]: proposes a taxonomy of web agents based on three properties: their embodiment, implementation location, and their ability to interface. They use these three properties in combination to talk about a variety of high-level bot types/roles. My proposed taxonomy covers all of the non-implementation aspects of Huang et al. [78] taxonomy.

(2003) Munroe and Luck [77]: proposes a taxonomy for autonomous agents which is comprised of three types of motivations used to satisfy goals: domain motivations, constraint motivations, and social motivations. Each of these three motivations has their own set of sub-concepts, many of which I was able to merge/map to the new taxonomy. My proposed taxonomy covers many of the sub-concepts.

(2004) Tasic and Agha [55]: proposes a taxonomy of autonomous agents that takes a systems approach to classifying how agents function and behave. They specify two categories of autonomous agents, weak autonomous agents (control of their own state, are reactive, and persist) and strong autonomous agents which have all the properties of weak autonomous agents plus goal-directness and pro-activeness. My proposed taxonomy covers all of their requirements for both weak and strong agency. Their taxonomy, however, unlike my software bot taxonomy, does not include any anthropomorphizing of agents since the authors do not believe that agents should be described by their anthropomorphic features as they are artificial systems.

(2003) Parunak et al. [84]: proposes a preliminary taxonomy of multi-agent interactions (“co-*x*”) properties. The authors focus on the interactions between agents, but I believe that many of the same interaction characteristics can be observed in agent to non-agent communication. Although the concepts presented in this taxonomy were slightly reorganized in the new software bot taxonomy, my taxonomy covers all except the communication centralization/decentralization and alignment mechanisms since they were too implementation/multi-agent system specific (out of scope).

(2003) Parunak and Fleischer [81]: presents an expanded design taxonomy of multi-agent interactions (“co-*x*” properties). It should be noted that the authors build off of this taxonomy in their earlier recent work, described above [81]. Although the concepts presented in this taxonomy were slightly reorganized in the new software bot taxonomy, my taxonomy covers all except the communication centralization/decentralization and alignment mechanisms, since they were too implementation/multi-agent system specific (out of scope).

(2005) Hector and Narasimhan [65]: proposes a new ontology of software agents based on existing taxonomies. The proposed classification scheme has 6 main features along with a set of possible values for each: pro-activeness, adaptiveness, mobility, collaboration, veracity, and disposition. All of these characteristics of agents are covered by my taxonomy, however, they had to be re-organized since some of the category values were not mutually exclusive.

(2007) Moya and Tolk [66]: proposes a taxonomy of agents/multi-agent systems that has a similar structure to my taxonomy of software bots. Their taxonomy had three high-level categories of properties: the situated environment, the population, and the agent characteristics. The situated environment dimension was composed of four properties, each with their own set of possible values: closure, dynamism, determinism, and cardinality. The population dimension was composed of four properties, each with their own set of possible values: size, diversity, homogeneity, goal structure, cooperatively. The agent characteristics dimension was composed of four properties, each with their own set of possible values: reasoning (architecture, beliefs, goals, reactivity), perception (access, accuracy, memory), and action (communication, protocol, negotiation). Since some of the properties described in population and situated environment were specific to multi-agent systems, a small subset of the properties (goal structure, protocol, negotiation) were not able to be merged/mapped into the new taxonomy; however, all of the remaining properties were covered in my proposed taxonomy.

The structure of their taxonomy was also similar to the one that emerged from the taxonomy development process: situated environment + population = environment facets, and agent characteristics = intrinsic + interaction facets.

(2008) van Otterlo et al. [71]: presents a characterization of sapient agents, based on the agent's cognitive concepts and abilities. Although they focused primarily on sapient agents, many of the characteristics they describe can be applied to other bot types: beliefs, goals, actions, emotions, learning, and properties of the social environment. My proposed taxonomy covers all of the characteristics listed above. However, the article also discusses a great deal of implementation specific details that were outside of the scope of my taxonomy and therefore were not included.

(2009) Sakarkar and Shelke [88]: presents a classification scheme for autonomous software agents based on their application, technologies, behaviours, and design structure. However, only a subset of the agent behaviour categories (mobility and reactivity) were covered in my taxonomy, since the remainder of the classification focused on applications, technologies, and implementation details.

(2009) Tolk and Uhrmacher [43]: develop a three-dimensional agent space focusing on the (1) the characteristics of agents, (2) their environments, and (3) the community of agents. The chapter also discusses various existing agent architectures, presents strategies for realizing key agent behaviours, identifies some applications of agents. They reflect on existing hierarchical taxonomies of agents and propose three-dimensional taxonomy is based upon the earlier framework by [66]. My taxonomy covers all of the facets they propose in their taxonomy.

(2012) Agüero et al. [76]: propose an agent capability taxonomy for dynamic environments. Their taxonomy distinguishes between agent's tasks, capabilities (many tasks), and behaviours (many capabilities); however, since my proposed taxonomy takes a black-box approach my taxonomy does not distinguish between tasks/capabilities/behaviours and instead merges them under-scheduling, with the values of interrupting/non-interrupting.

F.2 Snowballing Articles:

(1995) Goodwin [134]: proposes a framework that discusses agents as a whole (both physical and virtual agents), and therefore some of the properties they described do not map well to my taxonomy (e.g., configuration = body + position). Although some of the properties in their framework are covered in my taxonomy (e.g., binary/maintenance tasks), perceptive, predictive, affect, controller), many of their other properties were too high level (e.g., capable, successful, interpretive, rational, sound, predictive) or implementation specific to be included.

(1993) Foner [53]: starts their paper by introducing Julia, a Lobner prize winning software agent, and demonstrates her capabilities. They end the paper by highlighting “critical notions” for agents: autonomy, personalization, discourse, risk/trust, domain, graceful degradation, and cooperation, anthropomorphism. My taxonomy covers all of their “critical notions” of agency.

(1994) Kautz et al. [89]: presents a framework for agent interaction, through an example of a software agent they built as part of their research. Although this paper focuses mostly on the design of software agents, it ends by presenting a list of distinguishing agent properties: communication, temporal continuity, responsibility, robustness, multi-platform, and autonomy. I was able to map/merge all of the properties except responsibility as it related to the role of the software agents (i.e., has the ability to handle private information).

(1994) Etzioni and Weld [85]: proposes the use of a softbot interface for exploring the Internet. They also outline many behavioural properties (e.g., goal oriented, charitable, balanced, integrated) of software bots in a set of design principles. My taxonomy covers all of these properties except for “balance”, as it is inside the black-box.

(1994) Wooldridge and Jennings [52]: starts their paper by listing a few requirements for weak agency (autonomy, social ability, reactivity, pro-activeness), strong agency (mentalistic notions, emotional state), and other agent attributes (mobility, veracity, benevolence, rationality). My proposed taxonomy covers these requirements. The rest of the article dives into other researchers’ theories of agency, agent architectures, and agent languages, all of which are outside the scope of my taxonomy.

(1995) Russell and Norvig [5]: provides a description of basic agent properties (perceiving, affecting), a variety of agent subtypes, and their key properties (e.g., goals, reflex, internal state). My proposed taxonomy covers all of the properties the authors explicitly defined as belonging to agents. They also provide a list of environment properties (accessible, deterministic, episodic, static, discrete) that my taxonomy also covers.

(1996) Nwana [57]: proposes a topology of software agents based on a combination of three attributes that agents should exhibit: cooperation, learning, and autonomy. They also provide a list of other possible agent properties (e.g., static/mobile, deliberative/reactive, versatility, benevolent/non-helpful, delegate, temporally continuous, emotional attitudes, etc.)—my proposed taxonomy covers all of these properties. The remainder of the paper

identifies and discusses high-level subtypes of agents (e.g., collaborative agents, interface agents, information agents, etc.), which is outside of the scope of my taxonomy.

(1998) Huhns and Singh [80]: provides one of the most thorough taxonomies of software agents. The taxonomy classifies agents based on key characteristics (both intrinsic and extrinsic), the multi-agent systems they are a part of, the frameworks they are developed with, the roles they play, and their environment. My taxonomy covers most of their intrinsic (lifespan, level of cognition, construction, mobility, adaptability, modeling) and extrinsic (locality, social autonomy, sociability, friendliness, and interactions) characteristics were able to be merged/ mapped with the new taxonomy. My taxonomy also covers a few, non-implementation specific system characteristics (uniqueness, interface autonomy, execution autonomy) and all of the environment properties (knowable, predictable, controllable, historical, teleological, and real-time). The rough structure of their taxonomy was similar to my taxonomy's as well: (intrinsic characteristics = intrinsic facets), (extrinsic + system characteristics = interaction facets), (environment-agent characteristics = environment + interaction facets). Although the taxonomy presents a wide range of agent characteristics, the article does not formally definition many of the properties. Thus, it is likely that many more may be covered in my taxonomy, but I was not able to tell.

Although this taxonomy was quite extensive, it was missing anything regarding the anthropomorphism of software bots, as well as dimensions related to recent technological advancements. The paper also discusses architectural/infrastructures for building software agents and an analysis of existing subtypes/applications of agents, both outside of the scope of my taxonomy.

F.3 Online Articles:

Although most of the online articles do not provide formal classifications of bots, I discuss how they influenced and compare to my proposed taxonomy of software bots.

(2014) Schenk [73]: discusses some different bot design patterns and the properties of each of the types of bots: notifiers (notify), reactors (react), responders (respond), and conversationalists (conversation). Although this article presents a lot of implementation details, abstractions of the properties described above are covered in my taxonomy.

(2015) Libov [137]: does not present a classification of software bots; however, they describe the current (and make predictions for the future) state of text-based interactions with software bots. They describe some key ways in which we interact with bots on messaging platforms (1:1, 1:N, M:N), their asynchrony, their natural language capabilities, and their personas – all of which are covered in my propose taxonomy. They also discuss a variety of interaction and conversation styles, that helped shape the communicate/act facets of my taxonomy.

(2016) Roy [50]: does not present a classification of software bots; however, they highlight some key properties of modern chatbot systems in the form of a “pyramid of conversation” requirements: facts/transactional (base), episodic/contextual, social learning,

emotion/sentiment, episodic memory, personality (top). My proposed taxonomy covers these properties.

(2016) Storey and Zagalsky [45]: examines the various roles that bots are playing in the software development life-cycle. Although out of the scope of my taxonomy, they provide a classification of software bot subtypes that are used in software development (code bots, test bots, dev-ops bots, support bots, documentation bots) as well as a preliminary productivity framework for evaluating bot efficiency and effectiveness. Their paper does mention a few key bot properties (personalization, memory, pull/push), which my proposed taxonomy covers.

(2017) Storey [141]: this keynote was based off the paper above [45], and provides a deeper look at the properties of software bots. All of the properties mentioned in the presentation (pull/push, script/intelligent, personality, gender, dwelling, etc) are covered in my proposed taxonomy covers.

(2016) Dale [38]: does not present a classification of software bots; however, they provide an analysis of the current trends in chatbot technologies, what caused their sudden return, and where they are headed in the future. Although Dale does not call out any properties of software bots directly, the themes presented in this paper provided a great background and helped to shape many of the facets in my taxonomy.

(2016) Fisher [135]: does not present a classification of software bots; however, they discuss some best practices for building modern chatbots. Although Fisher does not call out any properties of software bots directly, the themes presented in this paper provided the background required to help to shape many of the categories of taxonomy (e.g., personality, conversation, specialization, robustness).

(2016) Grover [136]: does not present a classification of software bots; however, they provide a background of the evolution of bots in instant messaging platforms, conversational UI. Although Grover does not call out any properties of software bots directly, the themes presented in this paper provided a great background and helped to shape many of the facets in my taxonomy (e.g., conversation, actions, specialization).

(2016) Block [75]: does not present a classification of software bots; however, they provide a set of guidelines for building better bots, discuss bots' conversational abilities (which helped shape my taxonomy's language facet), and highlights some properties of bots (input/outputs, autonomy, synchrony – which are covered in my taxonomy).

(2016) MacPherson [138]: does not present a classification of software bots; however, they discuss how the behaviours of modern software bots can be used to create a feeling of authority and expertise during bot-human interactions (vocabulary/jargon, asking questions, varying response time, personality/tone, etc). Although MacPherson does not call out any properties of software bots directly, the themes presented in this paper provided a great background and helped to shape many of the facets in my taxonomy (e.g., anthropomorphism, conversation, synchronously).

(2017) Long et al. [83]: does not present a classification of software bots; however, they present a study of how bots are being discussed/created on Reddit. They provide a detailed list of the functionality that bots can provide on twitter (abstracted under the act facet in my taxonomy), and the roles they play (abstracted under specialization).

(2017) Nerds [46]: provides a great overview of the different type of modern software bots and their properties. They provide a list of ways to characterize bots: generalist/specialist, script bot/smart bots/intelligent agents – all of which are covered in my taxonomy. The article also discusses a variety of interaction and conversation styles which helped shape the communicate/act facets of my taxonomy.

(2000) Jurafsky and Martin [140]: does not present a classification of software bots; however, they discuss conversations w.r.t. chatbots: what is a conversation, their architectures (rule-based, corpus-based), dialogue management (initiative, universal commands, template generation, prompts, etc), and how to evaluation dialogue systems. Although the paper mostly provides implementation details, many of the dialogue properties they mentioned are covered, directly or in an abstracted form, in my proposed taxonomy.

(2017) Shevat [39]: does not present a classification of software bots; however, they provide a comprehensive look at developing modern chatbots. The book presents the most thorough description of modern software bots’ anthropomorphism facets and interaction mechanisms. Since the book also provides guidelines on how to build chatbots, it goes into much greater detail than my taxonomy for many of the facets. However, my taxonomy covers most of the concepts from this book at an abstracted level.

Appendix G

Validation: Subject Matter Tagging

Tables J, J, and J present the results of assessing three software bots according to the proposed software bot taxonomy on the **environment**, **intrinsic**, and **interaction** dimensions, respectively.

Dimension	Facets / Sub-Facets		Values	Poncho	Alexa	Mention-Bot (public repo)
ENVIRONMENT	Type		Standalone			
			Platform	✓ (social - intergrated into facebook)	✓ (ambient)	✓ (social - intergrated into Github)
	Scope		Bounded	✓	✓	✓
			Unbounded			
	Closure		Open		✓	✓
			Closed	✓ (require a facebook account)		
	Dynamism		Static			
			Dynamic	✓ (changes to the API where is gets its data)	✓	✓ (changes can be made to the code or the PR while it's thinking)
	Predictability		Stochastic		✓	
			Uncertain		✓ (uncertain about changes to outside devices)	
			Deterministic	✓ (actions result in message being posted)		✓ (actions result in message being posted)
	Permanence		Episodic	✓ (although the messages themselves persist, they only temporarily effect the environment)		✓ (although the messages themselves persist, they only temporarily effect the environment)
			Hybrid			
			Sequential		✓ (the changes made persist, eg. turning on a light)	
	Population	Cardinality	Singular		✓	
			Countable	✓ (poncho + the user)	✓ (alexa + anyone in the environment = countable)	
			Uncountable			✓ (anyone with access to the web)
		Diversity	Homogeneous			
			Heterogeneous	✓ (poncho + the user)	✓ (alexa + anyone in the environment)	✓ (mention-bot + anyone with access to the web)

Dimension	Facets / Sub-Facets		Values	Poncho	Alexa	Mention-Bot (public repo)
INTRINSIC DIMENSION	Knowledge	Memory	Long-Term	✓	✓	
			Short-Term	✓	✓	
			Future			
		Source	Encoded	✓	✓	✓
			Supplied	✓ (e.g., users tell it their location)	✓	(e.g., the Github webhook tell the bot there was a new PR)
			Learned		✓	(e.g., the bot grabs all of the data needed to preform its tasks)
	Reasoning	Mechanisms	Scripted	✓		✓
			Mixed		✓	
			Planning			
		Agency	None		✓	
			Veto		✓	
			Complete	✓		✓
		Predictability	Deterministic	✓	✓	✓
			Stochastic			
		Visibility	None			
			Transparent	✓ (traces -- visible in the conversation)		(traces -- visible in the conversation)
			Visible		✓ (repeats what it's done, or "chimes" if in brief mode)	
		Reactivity	Synchronous	✓	✓	✓
			Asynchronous			
		Scheduling	Single	✓ (can process one task at a time)	✓ (can process one task at a time)	✓ (only ever one task at a time)
	Multiple					
	Adaptability	Constraints	Open			 (non-adaptive)
			Constrained	✓	✓	
		Source	Internal		✓	
			External	✓	✓	
		Guidance	Undirected			
			Directed	✓	✓	
	Goals	Complexity	Low-High	(Low -- simple tasks to complete)	(Medium -- tasks require interfacing w/ multiple services)	(Low -- simple task to complete)
			Criticality	Low-High	(Low -- not completing high risk tasks)	(Medium -- sensitive information & financial tasks)
		Attainability	Achievable	✓ (e.g. the weather now)	✓ (e.g. turn on the lights)	✓
			Homoeostasis	✓ (e.g. provide weather updates daily)	✓ (e.g. remind me to x every y)	✓ (posts every time a new PR is opened)
		Explicitness	Explicit	✓	✓	✓
			Implicit		✓	
		Source	Internal			✓ (programmed into bot)
			External	✓ (goals stem from user requests)	✓ (goals stem from users)	
	Delegacy		None	✓		✓
			Partial			
			Complete		✓ (e.g. acts on behalf of users to make purchases)	
	Specialization		Generalist		✓ (in the middle)	
			Specialist	✓	✓	✓
	Anthropomorphism	Name	None			
			Representative			✓ (<i>"mention-bot"</i> depicts role)
			Unique	✓ (<i>"Poncho"</i>)	✓ (<i>"Alexa"</i>)	
		Age	None	✓		✓
			Static			
		Dynamic		✓		
		Gender	None	✓	✓	✓
			Gendered	✓ (<i>"I'm a male cat"</i>)	✓ (<i>"I'm female in character"</i>)	
		Ethnicity	None	✓	✓	✓
			None	✓ (<i>"I'm a male cat"</i>)		✓ (smiling robot in avatar)
		Embodiment	None			
			Embodied	✓ (2D avatar)	✓ (Logo)	✓ (2D avatar)
		Profession	None	?	✓ (<i>"My job is to help you"</i>)	✓
			Profession	?		
		Personality	None			✓
			Personality	✓	✓	
		Emotions	None			✓
	Superficial			✓		
	Logical		✓ (If insulted, "Ok, well then I think I'm going to take a short break", and it becomes unresponsive)			
	Lifecycle	Lifespan	Continuous	✓	✓	✓
			Transient			
			Terminating			
		Creation	Human	✓	✓	✓
			Bot			
		System				
		Reproduction	Non-Reproductive	✓		✓
	Reproductive			✓ (Can trigger "routines")		

Dimension	Facets / Sub-Facets		Values	Poncho	Alexa	Mention-Bot (public repo)
INTERACTION DIMENSIONS	Access		None			
			Constrained		✓ (can't completely access ambient environment)	✓ (bot user types can have less permissions than full users)
			Complete	✓ (complete access to fb messenger conversation)		
	Sense		Non-Sensing			
			Sensing	✓	✓	✓
	Act		Non-Acting			
			Acting	✓	✓	✓
	Communicate	Disposition	Antagonistic			
			Competitive			
			Indifferent			✓ (doesn't care what others are doing in the environment)
			Cooperative			
			Benevolent	✓ (always tries to do what you ask)	✓ (always tries to do what you ask)	
			Veracity	Truthful	✓	✓
		Cardinality	Untruthful			
			One-One	✓	✓	
			One-Many		✓	
			Many-Many		✓	✓
		Directionality	One-Way			✓ (Bot talks to user)
			Two-Way	✓	✓	
		Directness	Direct	✓	✓	✓
			Indifferent			
		Speech	None			
			Keywords			✓ (it follows a pre-defined script)
			Natural Language	✓	✓	
			Conversation	(it tries, but hasn't reached here)		
	Initiative		Reactive	✓	✓	✓ (responds when PRs are open)
			Proactive	✓		
	Robustness	Error Prevention	Bot	✓ (leveraged guided dialogs + input shortcuts like buttons)	✓ (uses guided dialogs to help guide users to provide information required to complete tasks)	X (no robustness)
			User			
		Error Discovery	Bot	✓ (repeating inupts back to the user before (explicit), providing user with list of alternatives)	✓ (repeating inupts back to the user before (explicit) or after (implicit))	
			User			
		Error Correction	Bot			
			User	✓ (allows user to correct mistake using same modality)	✓ (allows user to correct mistake using same modality)	
	Mobility		Static	✓	✓	
			Mobile			✓ (can go between PRs)

Appendix H

Study Questions

If the participant selected the online computerized task option, the questions were administered in an online questionnaire format. If the participant selected the in-laboratory computerized task option, the questions were administered as a semi-structured interview.

H.1 Participant Background & Experience

These questions were asked prior to classifying their software bot. It should be noted that not all questions were asked to all participants.

1. What is your occupation?
e.g., academia (professor, researcher), student, industry, other
2. What is your domain?
e.g., computer science, life science, healthcare, mathematics, other
3. How much experience do you have using software bots? (if applicable)
None (1) (2) (3) (4) (5) (6) (7) Expert
4. How long have you been using software bots? (if applicable)
 - ☐ Less than one year
 - ☐ Between one and three years
 - ☐ More than three years
5. How much experience do you have designing/building software bots? (if applicable)
None (1) (2) (3) (4) (5) (6) (7) Expert
6. How long have you been designing / developing software bots? (if applicable)
 - ☐ Less than one year
 - ☐ Between one and three years
 - ☐ More than three years
7. How much experience do you have researching software bots? (if applicable)
None (1) (2) (3) (4) (5) (6) (7) Expert
8. How long have you been researching software bots? (if applicable)
 - ☐ Less than one year
 - ☐ Between one and three years

☐ More than three years

9. Please describe the bot you built (if applicable)

H.2 Software Bot Taxonomy Feedback

These questions were asked as a follow-up when required. It should be noted that not all questions were asked to all participants.

1. How difficult was it to classify the bot using the Software Bot Taxonomy?
Easy (1) (2) (3) (4) (5) (6) (7) Difficult
2. What aspects of the Software Bot Taxonomy did you like?
3. What aspects of the Software Bot Taxonomy did you not like?
4. Is there anything you would change about the Software Bot Taxonomy?
5. Is there anything you think is missing from Software Bot Taxonomy?
6. Is there anything else you would like to add?
7. Do you see yourself using the Software Bot Taxonomy in the future? (if applicable)
8. Who do you think would benefit from using the Software Bot Taxonomy? (if applicable)
9. Do you have any other feedback?
10. Would you be interested in participating in a follow-up interview via video conference or in person, allowing us to gain more insights on your usage of Software Bot Taxonomy?

Appendix I

H.R.E.B. Ethics Approval



Office of Research Services | Human Research Ethics Board
Administrative Services Building Rm B202 PO Box 1700 STN CSC Victoria BC V8W 2Y2 Canada
T 250-472-4545 | F 250-721-8960 | uvic.ca/research | ethics@uvic.ca

Certificate of Approval

PRINCIPAL INVESTIGATOR: Carlene Lebeuf	ETHICS PROTOCOL NUMBER: 17-350 Minimal Risk Review - Delegated
UVic STATUS: Master's Student	ORIGINAL APPROVAL DATE: 12-Oct-17
UVic DEPARTMENT: COSI	APPROVED ON: 12-Oct-17
SUPERVISOR: Dr. Margaret-Anne Storey	APPROVAL EXPIRY DATE: 11-Oct-18
PROJECT TITLE: The Design, Development and Usage of Software Bots	
RESEARCH TEAM MEMBER Co-Investigators (UVic): Dr. Matthieu Foucault, Alexey Zagalsky; Researchers (UVic): Courtney Williams, Tania Ferman	
DEFERRED PROJECT FUNDING: None	
CONDITIONS OF APPROVAL	
This Certificate of Approval is valid for the above term provided there is no change in the protocol. Modifications To make any changes to the approved research procedures in your study, please submit a "Request for Modification" form. You must receive ethics approval before proceeding with your modified protocol. Renewals Your ethics approval must be current for the period during which you are recruiting participants or collecting data. To renew your protocol, please submit a "Request for Renewal" form before the expiry date on your certificate. You will be sent an emailed reminder prompting you to renew your protocol about six weeks before your expiry date. Project Closures When you have completed all data collection activities and will have no further contact with participants, please notify the Human Research Ethics Board by submitting a "Notice of Project Completion" form.	
Certification	
This certifies that the UVic Human Research Ethics Board has examined this research protocol and concluded that, in all respects, the proposed research meets the appropriate standards of ethics as outlined by the University of Victoria Research Regulations Involving Human Participants.  Dr. Rachael Scarth Associate Vice-President Research Operations	

Certificate Issued On: 12-Oct-17



Appendix J

Expert Tagging Session

Tables J, J, and J present the results of the expert tagging software bots according to the proposed software bot taxonomy on the **environment**, **intrinsic**, and **interaction** dimensions, respectively.

Dimension	Facets / Sub-Facets		Values	B1
ENVIRONMENT	Type		Standalone	
			Platform	✓ (social - intergrated into facebook)
	Scope		Bounded	✓
			Unbounded	
	Closure		Open	
			Closed	✓ (require a facebook account)
	Dynamism		Static	
			Dynamic	✓
	Predictability		Stochastic	
			Uncertain	
			Deterministic	✓ (actions result in message being posted)
	Permanence		Episodic	✓ (although the messages themselves persist, they only temporarily effect the environment)
			Hybrid	
			Sequential	
	Population	Cardinality	Singular	
			Countable	✓ (B1 + the user)
			Uncountable	
		Diversity	Homogeneous	
			Heterogeneous	✓ (B1 + the user)

Dimension	Facets / Sub-Facets		Values	B1
INTRINSIC DIMENSION	Knowledge	Memory	Long-Term	
			Short-Term	✓
			Future	
		Source	Encoded	✓
			Supplied	✓
			Learned	
	Reasoning	Mechanisms	Scripted	✓
			Mixed	
			Planning	
		Agency	None	
			Veto	
			Complete	✓
		Predictability	Deterministic	✓
			Stochastic	
			None	
		Visibility	Transparent	✓ (traces -- visible in the conversation)
			Visible	
			Synchronous	✓
		Reactivity	Asynchronous	
			Single	
			Multiple	✓
	Adaptability	Constraints	Open	(non-adaptive)
			Constrained	
		Source	Internal	
			External	
		Guidance	Undirected	
			Directed	
	Goals	Complexity	Low-High	(Low -- simple task to complete)
		Criticality	Low-High	(Low -- not completing high risk tasks)
		Attainability	Achievable	✓
			Homoeostasis	
		Explicitness	Explicit	✓
			Implicit	
		Source	Internal	
			External	✓ (users ask for things)
	Delegacy		None	✓
			Partial	
			Complete	
	Specialization		Generalist	
			Specialist	✓
	Anthropomorphism	Name	None	
			Representative	✓ (depicts product)
			Unique	
		Age	None	✓
			Static	
			Dynamic	
		Gender	None	
			Gendered	✓ (visible in logo)
		Ethnicity	None	
			Ethnicity	✓ (smiling robot in avatar)
		Embodiment	None	
			Embodied	✓ (2D avatar)
		Profession	None	
			Profession	✓
		Personality	None	
			Personality	✓
		Emotions	None	
			Superficial	✓
			Logical	
		Lifecycle	Lifespan	Continuous
	Transient			✓ (is there when people access it on FB)
	Terminating			
	Creation		Human	✓
			Bot	
System				
Reproduction	Non-Reproductive		✓	
	Reproductive			

Dimension	Facets / Sub-Facets		Values	B1
INTERACTION DIMENSIONS	Access		None	
			Constrained	
			Complete	✓ (complete access to fb messenger conversation)
	Sense		Non-Sensing	
			Sensing	✓
	Act		Non-Acting	
			Acting	✓
	Communicate	Disposition	Antagonistic	
			Competitive	
			Indifferent	
			Cooperative	
			Benevolent	✓ (always tries to do what you ask)
		Veracity	Truthful	✓
			Untruthful	
		Cardinality	One-One	✓
			One-Many	
			Many-Many	
		Directionality	One-Way	
			Two-Way	✓
		Directness	Direct	✓
			Indifferent	
		Speech	None	
			Keywords	
			Natural Language	✓
			Conversation	
	Initiative		Reactive	✓
			Proactive	
	Robustness	Error Prevention	Bot	✓
			User	
		Error Discovery	Bot	
			User	✓
		Error Correction	Bot	
		User	✓	
	Mobility		Static	✓
			Mobile	

Bibliography

- [1] M. Usman, R. Britto, J. Börstler, and E. Mendes, “Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method,” *Information and Software Technology*, vol. 85, pp. 43 – 59, 2017.
- [2] I. C. Society, P. Bourque, and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R))*. IEEE Computer Society Press, 2014.
- [3] A. Leonard, *Bots: The Origin of New Species*. Penguin Books Limited, 1998.
- [4] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, “A proposal for the dartmouth summer research project on artificial intelligence,” *AI Magazine*, vol. 27, no. 4, p. 12, 2006.
- [5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 1995, vol. 25, ch. 2.
- [6] W. Thomson, “The sorting demon of maxwell,” in *Proceedings of the Royal Society*, vol. 9, 1879, pp. 113–114.
- [7] J. Bub, “Maxwell’s demon and the thermodynamics of computation,” *Studies in History and Philosophy of Science: Studies in History and Philosophy of Modern Physics*, vol. 32, no. 4, pp. 569–579, 2001.
- [8] R. P. Martin, *The Language of Heroes: Speech and Performance in the Iliad. Myth and Poetics*. Cornell University Press, 1990.
- [9] P. McCorduck, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. AK Peters/CRC Press, 2009.
- [10] W. Clark, J. Golinski, and S. Schaffer, *The Sciences in Enlightened Europe*. University of Chicago Press, 1999.
- [11] K. Čapek, *R.U.R. (Rossum’s Universal Robots)*. Oxford University Press, 1951, english translation.

- [12] I. Kato, “Development of wabot-1,” *Biomechanism*, vol. 2, pp. 173–214, 1973.
- [13] P. Mowforth and I. Bratko, “Ai and robotics; flexibility and integration,” *Robotica*, vol. 5, no. 2, pp. 93–98, 1987.
- [14] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch, “Industry 4.0: The future of productivity and growth in manufacturing industries,” *Boston Consulting Group*, vol. 9, 2015.
- [15] A. M. Turing, “Computing machinery and intelligence,” in *Parsing the Turing Test*. Springer, 2009, pp. 23–65.
- [16] V. Cerf, “Parry encounters the doctor,” Tech. Rep., 1973.
- [17] R. Wallace, “Artificial linguistic internet computer entity (alice),” 1995.
- [18] R. Hoffer, T. Kay, P. Levitan, and S. Klein, “Smarterchild,” ActiveBuddy, 2001.
- [19] M. L. Mauldin, “Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition,” in *AAAI*, vol. 94, 1994, pp. 16–21.
- [20] “Verbot sylvie,” Virtual Personalities Inc, 1997.
- [21] J. Hendler, “Avoiding another ai winter,” *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 2–4, 2008.
- [22] O. Barnett, K. Famiglietti, R. Kim, E. PHoffer, and M. Feldman, “Dxplain on the internet,” in *Proceedings of the American Medical Informatics Association Symposium*. American Medical Informatics Association, 1998, p. 607.
- [23] H. S. Nwana, “Software agents: An overview,” *The Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–244, 1996.
- [24] C. Hewitt, “Viewing control structures as patterns of passing messages,” *Artificial Intelligence*, vol. 8, no. 3, pp. 323–364, 1977.
- [25] A. H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 2014.
- [26] L. Gasser and M. N. Huhns, *Distributed Artificial Intelligence*. Morgan Kaufmann, 1989, vol. 2.
- [27] B. Chaib-Draa, B. Moulin, R. Mandiau, and P. Millot, “Trends in distributed artificial intelligence,” *Artificial Intelligence Review*, vol. 6, no. 1, pp. 35–66, 1992.

- [28] M. Wooldridge and N. R. Jennings, “Intelligent agents: Theory and practice,” *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.
- [29] J. Müller, M. Wooldridge, and N. Jennings, *Intelligent Agents III. Agent Theories, Architectures, and Languages*, 1996.
- [30] J. Sculley, “The knowledge navigator,” 1987, educom Keynote.
- [31] R. Kozierok and P. Maes, “A learning interface agent for scheduling meetings,” in *Proceedings of the 1st International Conference on Intelligent User Interfaces*, 1993, pp. 81–88.
- [32] H. Lieberman, “Letizia: An agent that assists web browsing,” 1995, pp. 924–929.
- [33] A. Chavez and P. Maes, “Kasbah: An agent marketplace for buying and selling goods,” in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent technology*, 1996, p. 40.
- [34] P. Maes, “Artificial life meets entertainment: lifelike autonomous agents,” *Communications of the ACM*, vol. 38, no. 11, pp. 108–114, 1995.
- [35] J. L. McCarthy, “Artificial intelligence: A general survey,” *Artificial Intelligence*, vol. 5, no. 3, pp. 317–322, 1974.
- [36] D. Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence*. Basic Books, 1993.
- [37] J. W. Ng and D. H. Lau, “Going beyond web browsing to web tasking: Transforming web users from web operators to web supervisors,” in *Proceedings of the World Congress on Services (SERVICES)*. IEEE, 2013, pp. 126–130.
- [38] R. Dale, “The return of the chatbots,” *Natural Language Engineering*, vol. 22, no. 5, pp. 811–817, 2016.
- [39] A. Shevat, *Designing Bots*. O’Reilly Media, 2017.
- [40] The smart audio report. National Public Radio (NPR) & Edison Research. [Online]. Available: <http://nationalpublicmedia.com/wp-content/uploads/2018/01/The-Smart-Audio-Report-from-NPR-and-Edison-Research-Fall-Winter-2017.pdf>
- [41] R. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, University of California Irvine, 2000.
- [42] C. Lebeuf, M. A. Storey, and A. Zagalsky, “Software bots,” *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2018.

- [43] A. Tolk and A. M. Uhrmacher, “Agents: Agenthood, agent architectures, and agent taxonomies,” in *Agent-Directed Simulation and Systems Engineering*. Wiley, 2009, pp. 75–109.
- [44] C. Vouillon. (2015) Software bots: From do-it-yourself companion bots to ai powered software. [Online]. Available: <https://medium.com/point-nine-news/software-bots-c56aeedcfec3>
- [45] M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in *Proceedings of the 24th ACM Sigsoft International Symposium on Foundations of Software Engineering*, 2016, pp. 928–931.
- [46] B. Nerds. (2017) Types of bots: An overview. [Online]. Available: <http://botnerds.com/types-of-bots/>
- [47] D. Geer, “Malicious bots threaten network security,” *Computer*, vol. 38, no. 1, pp. 18–20, 2005.
- [48] L. McLaughlin, “Bot software spreads, causes new worries,” *IEEE Distributed Systems Online*, vol. 5, no. 6, p. 1, 2004.
- [49] G. Maus, “A typology of socialbots (abbrev.),” in *Proceedings of the ACM Conference on Web Science*, 2017, pp. 399–400.
- [50] S. D. Roy. (2016) What bots may come: A learning architecture for the next paradigm. [Online]. Available: <https://chatbotsmagazine.com/what-bots-may-come-a35b2bb9bd58>
- [51] Squareweave, “The user is drunk,” 2013. [Online]. Available: <https://www.youtube.com/watch?v=r2CbbBLVaPk>
- [52] M. Wooldridge and N. R. Jennings, “Agent theories, architectures, and languages: A survey,” in *International Workshop on Agent Theories, Architectures, and Languages*, 1994, pp. 1–39.
- [53] L. Foner, “What’s an agent, anyway? a sociological case study,” Agents Memo 93, Tech. Rep., 1993.
- [54] B. Hayes-Roth, “An architecture for adaptive intelligent systems,” *Artificial Intelligence*, vol. 72, no. 1-2, pp. 329–365, 1995.
- [55] P. T. Tosic and G. A. Agha, “Towards a hierarchical taxonomy of autonomous agents,” in *IEEE International Conference on Systems, Man and Cybernetics*, 2004, pp. 3421–3426.

- [56] J. Weizenbaum, “Eliza: A computer program for the study of natural language communication between man and machine,” *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.
- [57] H. S. Nwana, “Software agents: An overview,” *The Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–244, 1996.
- [58] C. Lebeuf, M.-A. Storey, and A. Zagalsky, “How software developers mitigate collaboration friction with chatbots,” 2017.
- [59] M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in *Proceedings of the 24th ACM Sigsoft International Symposium on Foundations of Software Engineering*, 2016, pp. 928–931.
- [60] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering—a systematic literature review,” *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009.
- [61] J. A. Sánchez, “A taxonomy of agents,” Interactive and Cooperative Technologies Lab, Report ICT-97-1, 1997.
- [62] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, p. 38.
- [63] L. Rosenfeld and P. Morville, *Information Architecture for the World Wide Web*. O’Reilly Media, Inc, 2002.
- [64] M. Yap and N. W. Keong, “Are life-like characteristics useful for autonomous agents?” in *Proceedings of the Third Annual Conference on Autonomous Agents*, 1999, pp. 336–337.
- [65] A. Hector and V. L. Narasimhan, “A new classification scheme for software agents,” in *Third International Conference on Information Technology and Applications*, 2005, pp. 191–196.
- [66] L. J. Moya and A. Tolk, “Towards a taxonomy of agents and multi-agent systems,” in *Proceedings of the Spring Simulation Multiconference*, 2007, pp. 11–18.
- [67] S. Hannah, *Sorting out card sorting: Comparing methods for information architects, usability specialists, and other practitioners*, 2008.
- [68] D. Spencer and T. Warfel, “Card sorting: A definitive guide,” *Boxes and Arrows*, p. 2, 2004.

- [69] S. Franklin and A. Graesser, “Is it an agent, or just a program?: A taxonomy for autonomous agents,” in *International Workshop on Agent Theories, Architectures, and Languages*, 1996, pp. 21–35.
- [70] D. Britz, “Deep learning for chatbots,” 2016.
- [71] M. van Otterlo, M. Dastani, M. Wiering, and J.-J. Meyer, *A Characterization of Sapient Agents*. Springer London, 2008, pp. 129–141.
- [72] S. D. Bird, “Toward a taxonomy of multi-agent systems,” *International Journal of Man-machine Studies*, vol. 39, no. 4, pp. 689–704, 1993.
- [73] W. Schenk. (2014) Bot design patterns: different ways to make different bots. [Online]. Available: <http://willschenk.com/bot-design-patterns/>
- [74] E. Zalta, “Stanford encyclopedia of philosophy,” The Metaphysics Research Lab, 2003.
- [75] R. Block. (2016) How to build a better bot. [Online]. Available: <https://workshop.begin.com/how-to-make-a-better-bot-c038626fd401>
- [76] J. Agüero, M. Rebollo, C. Carrascosa, and V. Julian, *Agent Capability Taxonomy for Dynamic Environments*. Springer Berlin Heidelberg, 2012, pp. 37–48.
- [77] S. Munroe and M. Luck, “Agent autonomy through the 3m motivational taxonomy,” in *Proceedings of the International Conference on Agents and Computational Autonomy*, 2003, pp. 55–67.
- [78] Z. Huang, A. Eliens, A. van Ballegooij, and P. de Bra, “A taxonomy of web agents,” in *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, 2000, pp. 765–769.
- [79] (2018) Personality. American Psychological Association. [Online]. Available: <http://www.apa.org/topics/personality/>
- [80] M. Huhns and M. P. Singh, “Agents and multiagent systems: Themes, approaches and challenges,” in *Readings in Agents*, 1998, ch. 1.
- [81] H. V. D. Parunak and M. Fleischer, “A design taxonomy of multi-agent interactions,” in *International Workshop on Agent-Oriented Software Engineering*, 2003, pp. 123–137.
- [82] M. M. de Graaf, S. B. Allouch, and J. van Dijk, “What makes robots social?: A user’s perspective on characteristics for social human-robot interaction,” in *International Conference on Social Robotics*, 2015, pp. 184–193.

- [83] K. Long, J. Vines, S. Sutton, P. Brooker, T. Feltwell, B. Kirman, J. Barnett, and S. Lawson, “Could you define that in bot terms?: Requesting, creating and using bots on reddit,” in *Proceedings of the Conference on Human Factors in Computing Systems*, 2017, pp. 3488–3500.
- [84] H. V. D. Parunak, S. Brueckner, M. Fleischer, and J. Odell, “A preliminary taxonomy of multi-agent interactions,” in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003, pp. 1090–1091.
- [85] O. Etzioni and D. Weld, “A softbot-based interface to the internet,” *Communications of the ACM*, vol. 37, no. 7, pp. 72–76, 1994.
- [86] H. V. D. Parunak, S. Brueckner, M. Fleischer, and J. Odell, “A preliminary taxonomy of multi-agent interactions,” in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003, pp. 1090–1091.
- [87] E. Paikari and A. van der Hoek, “A framework for understanding chatbots and their future,” *The 11th International Workshop On Cooperative and Human Aspects of Software Engineering an ICSE workshop*, 2018.
- [88] G. Sakarkar and N. M. Shelke, “A new classification scheme for autonomous software agent,” in *International Conference on Intelligent Agent Multi-agent Systems*, 2009, pp. 1–2.
- [89] H. Kautz, B. Selman, M. Coen, S. Ketchpel, and C. Ramming, “An experiment in the design of software agents,” in *Association for the Advancement of Artificial Intelligence*, 1994, pp. 438–443.
- [90] M.-L. Bourguet, “Towards a taxonomy of error-handling strategies in recognition-based multi-modal human–computer interfaces,” *Signal Processing*, vol. 86, no. 12, pp. 3625–3643, 2006.
- [91] H. Schmeck and C. Müller-Schloer, *A Characterization of Key Properties of Environment-Mediated Multiagent Systems*. Springer Berlin Heidelberg, 2008, pp. 17–38.
- [92] J. S. Aitken, F. Schmalhofer, and N. Shadbolt, *A knowledge level characterisation of multi-agent systems*. Springer Berlin Heidelberg, 1995, pp. 179–190.
- [93] H. Ko, S. Han, U. Kim, and H. Y. Youn, “A new agent characterization model and grouping method for multi-agent system,” in *IEEE International Conference on Information Reuse and Integration*, 2008, pp. 86–91.

- [94] D. Dagon, G. Gu, C. P. Lee, and W. Lee, "A taxonomy of botnet structures," in *Twenty-third Annual Computer Security Applications Conference*, 2007, pp. 325–339.
- [95] H. C. Wong and K. Sycara, "A taxonomy of middle-agents for the internet," in *Proceedings of the Fourth International Conference on Multiagent Systems*, 2000, pp. 465–466.
- [96] J. van Oijen and F. Dignum, *Agent Communication for Believable Human-Like Interactions between Virtual Characters*. Springer Berlin Heidelberg, 2013, pp. 37–54.
- [97] N. Afonso and R. Prada, *Agents That Relate: Improving the Social Believability of Non-Player Characters in Role-Playing Games*. Springer Berlin Heidelberg, 2009, pp. 34–45.
- [98] A. Oluyomi, S. Karunasekera, and L. Sterling, "An agent design pattern classification scheme: capturing the notions of agency in agent design patterns," in *11th Asia-Pacific Software Engineering Conference*, 2004, pp. 456–463.
- [99] M. O. Riedl and R. M. Young, *An Objective Character Believability Evaluation Procedure for Multi-agent Story Generation Systems*. Springer Berlin Heidelberg, 2005, pp. 278–291.
- [100] P. Doyle, "Believability through context using "knowledge in the world" to create intelligent characters," in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 2002, pp. 342–349.
- [101] M. Eslahi, R. Salleh, and N. B. Anuar, "Bots and botnets: An overview of characteristics, detection and challenges," in *IEEE International Conference on Control System, Computing and Engineering*, 2012, pp. 349–354.
- [102] J. Cascalho, L. Antunes, M. Corrêa, and H. Coelho, "Characterising agents behaviours: selecting goal strategies based on attributes," in *International Workshop on Cooperative Information Agents*, 2006, pp. 402–415.
- [103] K. Sumi and M. Nagata, "Characteristics of robots and virtual agents as a persuasive talker," in *International Conference on Universal Access in Human-Computer Interaction*, 2013, pp. 414–423.
- [104] P. Davidsson, S. Johansson, and M. Svahnberg, "Characterization and evaluation of multi-agent system architectural styles," in *International Workshop on Software Engineering for Large-scale Multi-agent Systems*, 2005, pp. 179–188.

- [105] E. Stinson and J. C. Mitchell, “Characterizing bots remote control behavior,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2007, pp. 89–108.
- [106] B. Hu and J. Liu, “Characterizing complex behavior in (self-organizing) multi-agent systems,” in *International Conference on Computational Science and its Applications*, 2005, pp. 1274–1283.
- [107] K. V. den Bosch, A. Brandenburgh, T. J. Muller, and A. Heuvelink, “Characters with personality!” in *International Conference on Intelligent Virtual Agents*, 2012, pp. 426–439.
- [108] Y. Hayashi and K. Miwa, “Cognitive and emotional characteristics of communication in human-human/human-agent interaction,” in *International Conference on Human-Computer Interaction*, 2009, pp. 267–274.
- [109] R. Prasad, V. V. Kumari, and K. Raju, “Comparison shopping agents: the essential characteristics and challenges to be met,” in *International Conference on Intelligent Agent & Multi-agent Systems*, 2009, pp. 1–2.
- [110] J. M. Darley, “Constructive and destructive obedience: A taxonomy of principal-agent relationships,” *Journal of Social Issues*, vol. 51, no. 3, pp. 125–154, 1995.
- [111] A. Smajgl, D. G. Brown, D. Valbuena, and M. G. Huigen, “Empirical characterisation of agent behaviours in socio-ecological systems,” *Environmental Modelling & Software*, vol. 26, no. 7, pp. 837–844, 2011.
- [112] G. Zoric, K. Smid, and I. S. Pandzic, “Facial gestures: taxonomy and application of non-verbal, non-emotional facial displays for embodied conversational agents,” *Conversational Informatics: An Engineering Approach*, pp. 161–182, 2007.
- [113] K. E. Merrick, “Game-playing agents and non-player characters,” in *Computational Models of Motivation for Game-Playing Agents*. Springer, 2016, pp. 45–65.
- [114] S. DiPaola, “Intelligent expression-based character agent systems,” in *International Workshop on Intelligent Virtual Agents*, 2009, pp. 3–4.
- [115] M. Cavazza, F. Charles, and S. J. Mead, “Interacting with virtual characters in interactive storytelling,” in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 2002, pp. 318–325.
- [116] K. Ito, “Introducing multimodal character agents into existing web applications,” in *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, 2005, pp. 966–967.

- [117] N. Tosa and R. Nakatsu, “Life-like communication agent-emotion sensing character” mic” and feeling session character” muse”, in *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, 1996, pp. 12–19.
- [118] S. Mukhopadhyay, S. Peng, R. Raje, M. Palakal, and J. Mostafa, “Multi-agent information classification using dynamic acquaintance lists,” *Journal of the Association for Information Science and Technology*, vol. 54, no. 10, pp. 966–975, 2003.
- [119] Y. Kitamura, H. Tsujimoto, T. Yamada, and T. Yamamoto, “Multiple character-agents interface: An information integration platform where multiple agents and human user collaborate,” in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 2002, pp. 790–791.
- [120] A. J. Jones and B. S. Firozabadi, “On the characterisation of a trusting agents aspects of a formal approach,” in *Trust and Deception in Virtual Societies*, 2001, pp. 157–168.
- [121] Y. Arafa and A. Mamdani, “Scripting embodied agents behaviour with cml: character markup language,” in *Proceedings of the 8th International Conference on Intelligent User Interfaces*, 2003, pp. 313–316.
- [122] Y. Takeuchi and Y. Katagiri, “Social character design for animated agents,” in *IEEE International Workshop on Robot and Human Interaction*, 1999, pp. 53–58.
- [123] A. F. B. Neto and F. S. C. da Silva, “Synthetic characters with personality and emotion,” in *International Workshop on Intelligent Virtual Agents*, 2009, pp. 533–534.
- [124] G. Cabri, M. Puviani, and F. Zambonelli, “Towards a taxonomy of adaptive agent-based collaboration patterns for autonomic service ensembles,” in *International Conference on Collaboration Technologies and Systems*, 2011, pp. 508–515.
- [125] R. Damiano and V. Lombardo, “Using values to turn agents into characters,” in *International Conference on Agents and Artificial Intelligence*, 2009, pp. 283–296.
- [126] W. Ke and J. Mostafa, “Visualizing multi-agent collaboration for classification of information,” *Proceedings of the Association for Information Science and Technology*, vol. 45, no. 1, pp. 1–4, 2008.
- [127] Y. Kitamura, “Web information integration using multiple character agents,” in *Life-like Characters*, 2004, pp. 295–315.
- [128] C. Baraniuk, “You’ve got personality for a bot,” 2014.

- [129] M. Hare and P. Deadman, “Further towards a taxonomy of agent-based simulation models in environmental management,” *Mathematics and Computers in Simulation*, vol. 64, no. 1, pp. 25–40, 2004.
- [130] S. J. Read, J. Talevich, D. A. Walsh, G. Chopra, and R. Iyer, *A Comprehensive Taxonomy of Human Motives: A Principled Basis for the Motives of Intelligent Agents*. Springer Berlin Heidelberg, 2010, pp. 35–41.
- [131] S. Höppner, “An agents’ definition framework and a methodology for deriving agents’ taxonomies,” in *Ki: Advances in Artificial Intelligence*, 2003, pp. 618–632.
- [132] V. Filatov, “About one approach to the classification of program agents,” in *International Conference on Modern Problems of Radio Engineering, Telecommunications, and Computer Science*, 2006, pp. 410–411.
- [133] N. R. Jennings and J. R. Campos, “Towards a social level characterisation of socially responsible agents,” *IEEE Proceedings: Software*, vol. 144, no. 1, pp. 11–25, 1997.
- [134] R. Goodwin, “Formalizing properties of agents,” *Journal of Logic and Computation*, vol. 5, no. 6, pp. 763–781, 1995.
- [135] R. Fisher. (2016) The seven habits of highly effective chatbots. [Online]. Available: <https://chatbotsmagazine.com/the-seven-habits-of-highly-effective-chatbots-79a0e3c962db>
- [136] D. Grover. (2016) Bots won’t replace apps. better apps will replace apps. [Online]. Available: <http://dangrover.com/blog/2016/04/20/bots-wont-replace-apps.html>
- [137] J. Libov. (2015) Futures of text: A survey of all the current innovation in text as a medium. [Online]. Available: <http://whoo.ps/2015/02/23/futures-of-text>
- [138] S. MacPherson. (2016) How to interact with bots? dealing with the complexity of a new design paradigm. [Online]. Available: <https://chatbotsmagazine.com/how-to-interact-with-bots-dealing-with-the-complexity-of-a-new-design-paradigm-e89fd7131921>
- [139] M. Clément and M. J. Guitton, “Interacting with bots online: Users’ reactions to actions of automated programs in wikipedia,” *Computers in Human Behavior*, vol. 50, pp. 66–75, 2015.
- [140] D. Jurafsky and J. H. Martin, “Dialog systems and chatbots,” in *Speech and Language Processing: An Introduction to Natural Language Processing*. Prentice Hall, 2000, ch. 28.

- [141] M.-A. Storey, “To bot or not: How bots can support collaboration in software engine,” 2017, keynote at 11th IEEE International Conference on Global Software Engineering. [Online]. Available: <https://www.slideshare.net/mastorey/icgse-2016-storey>