

Submitted by  
**Tomás Murillo-Morales**

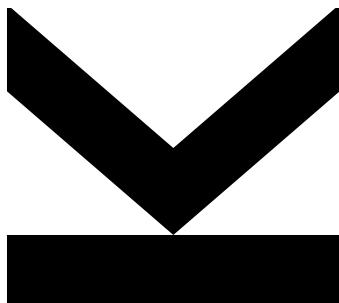
Submitted at  
**Institute of Integrated  
Studies**

Supervisor and  
First Evaluator  
**Dr. Klaus Miesenberger**

Second Evaluator  
**Doc. RNDr. Ivan Kopeček**

November 2021

# Accessible Diagrams through Natural Language



Doctoral Thesis

to obtain the academic degree of

Doktor der technischen Wissenschaften

in the Doctoral Program

Technische Wissenschaften



# Statutory Declaration

Tomás Murillo-Morales, M.Sc.

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.

This printed thesis is identical with the electronic version submitted.



Linz, 18. November 2021

---

Tomás Murillo-Morales



# Acknowledgements

I would like to thank my supervisor, Klaus Miesenberger, for letting me discover and pursue several exciting projects in the field of accessibility, including, but not limited to, this work.

This thesis would not have been possible without the groundwork put forward by Ivan Kopeček, Radek Ošlejšek, and Jaromír Plhák in their *Communicative Images*. For that I am also profoundly thankful to them.

Thanks to Bernard Stöger for his invaluable feedback as a blind accessibility expert. I would also like to thank the remaining members of the Institute Integriert Studieren for always being willing to share their expertise and engage in interesting discussions.

I would like to thank the organizers and participants of the International Camp on Communication and Computers (ICC), for creating an incredible inclusive environment for the spread of knowledge and ideas. A special gratitude goes to the persons who took time to participate in my workshops on accessible graphics; this thesis would not exist without you.

Last, but not least, my deepest gratitude goes to my family and friends, who made this whole thing much more bearable, especially in times of pandemic. Thank you!



# Kurzfassung

Diagramme bedienen sich der menschlichen Fähigkeiten der Wahrnehmung, der Kognition und des Gedächtnisses, um große Datenmengen effektiv mit sehenden Menschen zu teilen. Neben anderen Vorteilen können sie Inhalte, die per se implizit sind, explizit machen und als "externer Speicher" dienen, der dabei hilft, die kognitiven Anforderungen bei der Analyse des zu Grunde liegenden Datenbestandes zu bewältigen. Infolgedessen wurden Diagramme zum Standard in vielen Bereichen des täglichen Lebens. Blinde Menschen sind allerdings im allgemeinen von dieser Art grafischer Darstellung ausgeschlossen, genauso wie von anderen visuellen Inhalten wie Photographien. Dieser fehlende Zugang zu Diagrammen trägt unmittelbar zum sozialen Ausschluss aus der Informationsgesellschaft bei, von der Ausbildung und den Berufsaussichten bis hin zur Unterhaltung und vielen anderen täglichen Aktivitäten.

Die vorliegende Dissertation hat damit begonnen, zu untersuchen, ob aktuell verfügbare nicht-visuelle Alternativen zu Diagrammen den besonderen Bedürfnissen der blinden Bevölkerung in Hinsicht auf Zugänglichkeit Rechnung tragen. Das Resultat dieser Problemanalyse ist, dass die aktuellen Konzepte zum nicht-visuellen Zugang zu Diagrammen die Erfordernisse der Benutzbarkeit - Usability - für die blinde Bevölkerung klar verfehlt. Als Antwort hierauf präsentiert die vorliegende Arbeit ein in sich geschlossenes Framework für den holistischen Zugang blinder Menschen zu Diagrammen basierend auf einem adaptierten "Natural Language Interface" - NLI -, einer Schnittstelle zur natürlichen Sprache, welche von vielversprechenden, schon vorhandenen Forschungsarbeiten über den auf natürlicher Sprache beruhenden Zugang zu Photographien stammt. NLI's zu Diagrammen bieten viele interessante Eigenschaften für ihre Anwendung im Bereich der Accessibility, etwa die Fähigkeit, Informationen auf allen Ebenen der Abstraktion zu kommunizieren, ohne spezielle Hard- und Software zu benötigen, und sie sind eine intuitive Interaktionsmethode, die kein spezielles Training für die Benutzer erfordert.

Das vorgeschlagene Framework setzt sich aus vier Hauptelementen zusammen: (1) Eine Hierarchie von Ontologien, die Visualisierungs-Semantik unterstützen; (2) semantisch angereicherte Vektorgraphiken, denen mittels unterstützender Ontologien die Semantik dessen hinzugefügt wird, das sie darstellen sollen; (3) Semantic Annotator for Inkscape (SAI), ein Software-Werkzeug zum effizienten Erstellen von semantisch angereicherten Diagrammen; und (4) Accessible Universal Diagrams through Language (AUDiaL) - durch Sprache zugängliche, universelle Diagramme -, ein zugänglicher, web-basierter NLI Prototyp für semantisch angereicherte Diagramme, der für die besonderen Bedürfnisse blinder Anwender erstellt ist.

Eine Validierung von AUDiaL mit blinden Teilnehmern zeigt, dass ein geeignet adaptiertes NLI ein geeignetes Mittel zum effektiven, effizienten und benutzerfreundlichen nicht-visuellen Zugang zu Diagrammen sein kann. Die Teilnehmer, die an den vorgelegten Daten interessiert waren, zogen die Verwendung von AUDiaL klar der Arbeit mit den traditionellen Gegenstücken in der Diagrammverarbeitung vor. Darüber hinaus konnte das vorgelegte Framework als Grundlage für zukünftige Forschungen in heterogenen Bereichen dienen.

# Abstract

Diagrams leverage the human perceptual, cognitive, and memorial capacities to communicate large amounts of data effectively to sighted persons. Among other benefits, they can make otherwise implicit information explicit and serve as an external memory that lighten the cognitive demands of analyzing the underlying dataset. Consequently, diagrams have become commonplace in many aspects of daily life. However, blind persons are generally excluded from accessing this kind of graphic, even more so than other visual information such as photographs. Lack of access to diagrams contributes directly to social exclusion in the information society, from education and job prospects to entertainment and many everyday activities.

This thesis began by studying whether current non-visual accessible alternatives to diagrams met the specific access needs of the blind population. The conclusion from this problem investigation is that state-of-the-art approaches to non-visual access to diagrams invariably fail to meet the usability requirements of the blind population. As a result, this thesis proposes a self-contained framework for holistic non-visual access to diagrams based on an adapted Natural Language Interface (NLI), stemming from promising existing research in natural-language based access to photographs. NLIs to diagrams present many compelling properties for their application in accessibility, such as being able to communicate information at all levels of abstraction, not requiring specific hardware and software, and being an instinctive means of interaction without the need for user training.

The suggested framework is composed of four main elements. (1) A hierarchy of ontologies underpinning visualization semantics, (2) semantically-enhanced vector graphics which are augmented with the semantics of what they depict via supporting ontologies, (3) Semantic Annotator for Inkscape (SAI), a software tool for effective authoring of semantically-enhanced diagrams, and (4) Accessible Universal Diagrams through Language (AUDiaL), an accessible Web NLI prototype to semantically-enhanced diagrams designed to meet the specific needs of blind users.

Validation of AUDiaL with blind participants shows that a properly adapted NLI can be a favorable means to effective, efficient, and user-friendly non-visual access to diagrams. Participants interested in the data at hand displayed a strong preference for using AUDiaL as opposed to traditional diagram counterparts. In addition, the proposed framework could serve as the basis for further research in heterogeneous domains.



# Contents

<b>Kurzfassung</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Contents</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	2
1.2 Problem Statement . . . . .	14
1.3 Purpose . . . . .	17
1.4 Research Questions . . . . .	19
1.5 Method . . . . .	21
<b>2 State of the Art</b>	<b>27</b>
2.1 Linear Approaches . . . . .	28
2.2 Tactile and Haptic Approaches . . . . .	41
2.3 Sonification Approaches . . . . .	49
2.4 Hybrid Approaches . . . . .	58
2.5 Speech-supported Approaches . . . . .	62
2.6 Communicative Images . . . . .	71
2.7 Web Content Accessibility . . . . .	82
2.8 Summary . . . . .	87
<b>3 Formalizing Visualization Semantics</b>	<b>91</b>
3.1 Motivation and Problem Investigation . . . . .	91
3.2 Ontologies and “Linked Data” . . . . .	97
3.3 Linked Data Applications in Computer Vision . . . . .	112
3.4 Formalizing Visualization Semantics . . . . .	116
3.5 A Hierarchy of Ontologies for Visualization . . . . .	121
3.6 Upper Visualization Ontology . . . . .	130
3.7 Visualization Domain Ontology . . . . .	143
3.8 Data Domain Ontology . . . . .	158
3.9 Visualization Tasks Ontology . . . . .	163
3.10 System and User Ontologies . . . . .	170

3.11	Summary . . . . .	172
<b>4</b>	<b>Semantic Annotation of Diagrams</b>	<b>175</b>
4.1	Digital Image Formats . . . . .	175
4.2	Scalable Vector Graphics . . . . .	176
4.3	Semantically-Enhanced Graphics . . . . .	179
4.4	Authoring Semantically-Enhanced Graphics . . . . .	188
4.5	Semantic Annotator for Inkscape . . . . .	202
4.6	Summary . . . . .	210
<b>5</b>	<b>Natural Language Interface to Diagrams</b>	<b>213</b>
5.1	Natural Language Interfaces . . . . .	214
5.2	Natural Language Interfaces to Conceptual Models . . . . .	218
5.3	AUDiaL: Accessible Universal Diagrams through Language . . . . .	223
5.4	Requirements Specification . . . . .	224
5.5	NLI Prototype Design . . . . .	229
5.6	Question Analysis Phase . . . . .	236
5.7	Phrase Mapping Phase . . . . .	246
5.8	Disambiguation Phase . . . . .	252
5.9	Task Execution Phase . . . . .	266
5.10	Answer Generation Phase . . . . .	276
5.11	Summary . . . . .	279
<b>6</b>	<b>Accessibility and Validation of NLIs to Diagrams</b>	<b>281</b>
6.1	Motivation . . . . .	281
6.2	Resemblance preservation . . . . .	286
6.3	Spatial Indexing and Navigation . . . . .	287
6.4	Object Annotation . . . . .	292
6.5	Home Nodes . . . . .	294
6.6	High-level Summaries . . . . .	296
6.7	Review of HCI Techniques for Accessibility . . . . .	305
6.8	Web Interface Accessibility Evaluation . . . . .	305
6.9	Prototype Validation . . . . .	314
6.10	Research Setup . . . . .	316
6.11	Research Execution . . . . .	327
6.12	Data Analysis . . . . .	329
6.13	Conclusions from the Empirical Cycle . . . . .	344
6.14	Summary . . . . .	346
<b>7</b>	<b>Discussion</b>	<b>349</b>
7.1	Overview . . . . .	349
7.2	Results . . . . .	350
7.3	Outstanding Issues . . . . .	355
7.4	Future Work . . . . .	356

<b>List of Figures</b>	<b>359</b>
<b>List of Tables</b>	<b>363</b>
<b>List of Algorithms</b>	<b>365</b>
<b>Glossary</b>	<b>367</b>
<b>Acronyms</b>	<b>371</b>
<b>Appendix A: Instructions to Participants</b>	<b>375</b>
<b>Appendix B: Validation Diagrams - Images</b>	<b>377</b>
<b>Appendix C: Validation Diagrams - Long descriptions</b>	<b>381</b>
<b>Appendix D: Validation Tasks</b>	<b>383</b>
<b>Appendix E: Publications</b>	<b>385</b>
<b>Bibliography</b>	<b>387</b>



# Introduction

This thesis builds on the body of accessibility research and, more specifically, on the body of accessibility to knowledge representation graphics. In developed countries around the world access to digital information is vital in most aspects of a person's life, from entertainment and socialization to education and employment. Graphical information can be a very important component of the stream of information that permeates people's lives, such as personal photography in social networks or diagrammatically represented data supporting informational pieces in journalism or science. The human visual system automatically infers perceptual elements that, among other cognitive advantages, let sighted persons solve geometry problems effortlessly and infer otherwise implicit information in the form of spatial relationships, thus making graphical information highly didactic and a commonplace occurrence in everyday life.

Worldwide, there are 285 million visually impaired people, of which 39 million are blind. If they are to play a full role in society it is thus paramount that they have full access to information, including that in graphical form. Assistive technologies combined with best practices in the authoring process let linear information, such as text, become accessible to visually impaired readers, the most well-known example thereof perhaps being Braille text. However, in the case on two-dimensional information, such as graphics, the difficulty of creating appropriate alternative accessible counterparts increases significantly.

Many specific approaches to accessibility of graphics may be employed depending on the domain of the image. For instance, maps can be printed in raised dots and lines along with Braille labels on swell-paper, and simple line charts can be sonified by representing each data point with a musical pitch. Domain-dependent approaches are however very restrictive, as specialized software and/or hardware which may be expensive have to be employed and a sighted specialist usually needs to produce them. Moreover, the lack of individual autonomy for the user along with the steep learning curve and loss of display resolution that these approaches present hinder their widespread take-up by the visually impaired population.

Graphics may also be described by means of natural language through verbal descriptions or dialogue interfaces. Graphics on the World Wide Web (WWW) are usually described by means of alternative verbal descriptions using the `alt` and `longdesc` attributes of HTML image elements that can be accessed by screen readers and refreshable braille displays. This is a sufficient approach for simple graphics whose meaning can be conveyed by means of short textual descriptions. However, in the case of very detailed or highly specialized graphics interactive and often domain-specific approaches are more appropriate. In any case, up to now alternative descriptions are not present for most graphical contents on the Web, and automatic image description methods are not entirely reliable yet.

Communicative graphics are a novel approach to the problem of non-visual access to mainstream graphics, in which semantically annotated graphics are integrated with a Natural Language Interface (NLI) so that users can then fetch or modify their content by means of dialogue via an accessible interface. Recent research shows that NLIs are a promising approach to non-visual access to photographs, diagrams and charts; in addition, they may be employed to perform useful tasks in heavily sight reliant domains such as maps. Additionally, NLIs do not require of specific software or hardware, as they can be accessed online the same way blind people are used to navigate the Web. Therefore, communicative graphics have the potential of emerging as a holistic approach to the accessibility of many conventional graphics, freeing users from having to employ cumbersome, ill-suited, and domain-dependent methods. They also present many potential uses beyond the scope of accessibility for visually impaired people, such as adapting graphics to be better understood by people with a developmental disability or semantically enhancing educational graphics to be accessed by regular students. However, being a novel approach, communicative graphics have not been thoroughly evaluated, and so far their domain of applicability has been limited to simple photographic content.

This thesis aims to broaden the concept of communicative graphics by expanding their domain of applicability beyond that of real-world pictures. Thus the main contribution of this work is the *definition of a framework for authoring and accessing knowledge representation graphics (i.e. diagrams) non-visually by means of natural language with a special focus on statistical charts*. The resulting framework will advance and stimulate the take-up and further research of semantically-enhanced graphics for accessibility to mainstream images on the Web. This thesis characterizes the process of designing and validating such framework, a process which is preceded by a problem investigation phase in which the problem is explored, knowledge questions posed, and currently available treatments for the problem studied and their limitations noted.

## 1.1 Background

### 1.1.1 Graphics in the Information Society

The amount of information and its availability to the general public have experienced a drastic increase in recent times. In the last thirty years, especially with the widespread take up of the Internet, contemporary societies are being conceived in a new way, in which information is a distinguishing feature of the modern world [Web06]. Nowadays all sorts of published

information are readily available to the general public, from printed books and newspapers to online academic papers, weather forecasts, social networks, discussion forums, etc.

Citizens in the so-called ‘Information Society’ need to be able to access this information in order to live ordinary lives. Access to knowledge has become a central resource in the economy, as nowadays the majority of jobs involve working with knowledge [Ste02]; but also in education, politics and most people’s personal life. Lack of access to information contributes directly to social exclusion [Moo00].

Especially relevant is the sea of information contained in the World Wide Web (WWW), as Internet usage everywhere soars. In 2016, close to one out of two people worldwide (47%) had access to the Internet. In the developed world, this figure was 81% [Int16]. Almost one billion households in the world have Internet access. The Web permeates almost every aspect of many persons’ lives, who use it for social, educational and entertainment purposes [DEM11]. Digital publishing is expected to soon become the dominant publishing format, and it has been steadily overthrowing traditional print publishing in the past decade [Mor12].

It has been estimated that human beings absorb as much as 80 per cent of information about their immediate environment by means of sight [Wila]. As a direct consequence thereof, much of the available information on the Web is based on or supported by visual means. Visual aspects of communication such as photographs, graphs, diagrams or maps are usually placed in print and online documents in order to enrich them with information that is better presented in a visual manner or for decorative purposes.

Broadly speaking, graphics can be divided into two main categories according to their intended use by readers according to their author. Some graphics are characterized by their intention to amuse, delight, persuade, invigorate, provoke or otherwise stimulate their audience. On the other hand, there are graphics which intend to describe, explain, inform or instruct [Ric84]. Oftentimes, graphics as used as supporting media for the neighboring text. The main content of a document may also be purely graphic, such as in personal pictures shared on social networks or in knowledge representation graphics which, although often accompanied by a textual legend, can communicate complex ideas with clarity, precision and efficiency on their own [Tuf97]. This is especially true on the WWW, in which websites can be richer in graphical content as there are fewer space and modality limitations compared to print media. Only on Facebook in 2013 there was an upload rate of 350 million photos a day [Int13], as they are the primary type of content posted and shared on the social network [Red14]. Photos posted by users are the most engaging content on both Facebook and Twitter, boosting user interaction between 35% and 87% compared to text-only posts [Red14]. These facts clearly highlight the importance of graphical content nowadays on the Web.

Moreover, the advent of the WWW has given way to a whole new class of images in the form of highly interactive, dynamic graphics whose responsive capabilities make it possible for enhanced didactic benefits to be exploited. Dynamic graphics are becoming widespread in digital publishing, where they are used to aid the readership in navigating dynamic structured information. Interactive (i.e. dynamic) graphics can be defined as images that will change their appearance in response to actions taken by an external agent [Kea14]. A whole emerging

class of visualization attempts to combine narratives with interactive graphics in order to create “data stories” [SH10] in which visualization designers combine the skills of computer science, statistics, artistic design and storytelling [Cuk10].

### 1.1.2 Visualizing Information

The previous section advanced the notion that certain graphics can aid people with problem solving and understanding, or, as the well-known aphorism goes: “a picture is worth 10,000 words”. The pedagogical benefits of attaching meaning to familiar visual patterns have been known since ancient times. Nevertheless, abstract, non-representational pictures that display information were not used until the advancement of early modern chemical knowledge demanded it, when chemical theories were first represented by diagrams [Edd14]. Only after the publication of Descartes’ *La Géométrie* in 1637 was the principal tool for the creation of most statistical charts, the Cartesian coordinate system, available, and with it the behavior of mathematical functions and theoretical relations among physical variables could be explored by means of analytical geometry [LS89]. Statistical charts were invented not before 1750 – 1800, long after other advanced mathematical aspects such as logarithms, the calculus or the basics of probability theory [Tuf01].

Problem solving by humans beings (defined as any goal-oriented sequence of cognitive operations) has two critical attributes [Jon00]: first, problem solving requires the mental representation of the situation in the world. This means that the problem solver must construct a mental representation, or mental model, of the problem. Second, problem solving requires some activity-based manipulation of the problem space. Building a mental model of involves using both *internal representations*, stored in the problem solver’s brain, and *external representations*, recorded on a paper or some other medium. External representations act as an external memory that can be used in parallel with the internal mental representation to maximize computational offloading in solving the problem. In addition, external representations can be divided into two types [LS87]:

- *Sentential representations*, in which expressions form a sequence corresponding to sentences in a natural-language description of the problem, i.e. elements appear in a single one-dimensional sequence. Examples of sentential (also known as linear) representations are verbal and certain musical and mathematical notations [Ber11].
- *Diagrammatic representations*, in which expressions correspond to the components of a diagram describing the problem, i.e. data structures in which information is indexed by two-dimensional location. Most information visualization graphics, such as statistical charts, belong to this category.

The fundamental difference between diagrammatic and sentential representations is that the latter does not preserve explicitly the information about the topological and geometric relations among the components of the problem [LS87]. Even though both representations of the same problem are informationally equivalent, they are not computationally equivalent.

The difficulty of a problem can be greatly reduced, by over an order of magnitude in some cases [KHS85], by using an appropriate representation of the same problem [CCS01]. For instance, it would take 20,000 successive instants of perception to compare two data tables of 100 rows by 100 columns. The same data displayed graphically could be easily compared, even instantaneously if properly displayed [Ber11]. Another example is how much easier detecting cycles in a link diagram (for example, a mathematical graph) becomes when visually represented. Let us consider the link diagram shown in Figure 1.1.

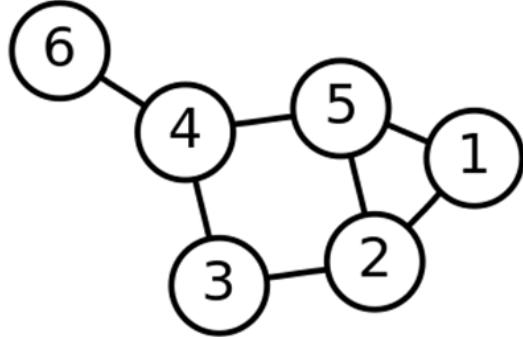


Figure 1.1: A mathematical graph containing three cycles<sup>1</sup>

It becomes immediately noticeable that this graph contains three cycles i.e. a path of edges and vertices wherein a vertex is reachable from itself. These are the cycles 1-2-5-1, 1-2-3-4-5-1, and 2-3-4-5-2. Let us now take a look at the following equivalent sentential representation of the graph in which the edges between two vertices are informally listed:

- (Vertex 1, Vertex 5)
- (Vertex 1, Vertex 2)
- (Vertex 2, Vertex 5)
- (Vertex 2, Vertex 3)
- (Vertex 3, Vertex 4)
- (Vertex 4, Vertex 5)
- (Vertex 4, Vertex 6)

Both representations of the graph are informationally equivalent in that they contain the same amount of information i.e. the structure of a mathematical graph. However, whereas on the diagrammatic representation of Figure 1.1 the three cycles could be immediately inferred by means of sight, on the sentential representation this task demands much more cognitive processing from the reader. This example illustrates the fact that “understanding” information means simplifying it. In turn, simplification involves reducing a vast amount of data to the small number of categories of information that we are capable of taking into account in dealing

---

<sup>1</sup>Image source: [https://en.wikipedia.org/wiki/Cycle\\_\(graph\\_theory\)](https://en.wikipedia.org/wiki/Cycle_(graph_theory))

with a given problem [Ber11]. In the graph example above, its diagrammatic representation allows sighted readers to rapidly perceive it as a structure consisting of two loops of vertices and an additional vertex not belonging to any loop. This example goes to show how graphic representations exploit the natural perceptual, cognitive, and memorial capacities of human beings [LS89], and that the brain finds it easier to process certain information if it is presented as an image rather than as words or numbers [Cuk10]. Therefore, properly graphically displaying information simplifies it, assisting the reader in building his or her mental model of the problem, thereby easing the problem solving process.

Nevertheless, graphics are not always appropriate to represent certain problems. The decision whether to transcribe the information graphically should be based on an evaluation of the specific properties and efficiency of each “language” i.e. each sign-system [Ber11]. This decision can also be influenced by other factors such as acquired habits, personal aptitudes, and even fashion. Moreover, the spectrum of choices when deciding which kind of graphic representation to use is much wider than most graphic designers suspect, which makes choosing the right one a very demanding task [Ber11].

In short, visualization deals with the inhuman scale of the information and the need to present it at the very human scale of what the eye can see [Cuk10]. Many cognitive accounts of the understanding of graphs and charts can be found in the relevant literature e.g. [LS87, CM85, Kos89, Pin90, Loh93], that analyze the understanding of knowledge representation graphics in terms of perceptual features, memory structures and the processing of visual information. Due to these cognitive benefits, data visualization techniques that support storytelling to improve its understanding are nowadays commonplace in science, education, and journalism

### 1.1.3 A Taxonomy of Diagrams

In this section the main concepts related to graphics and graphically displayed information that will be used throughout this thesis will be listed and defined. Many of these concepts are as a matter of course instinctively understood by most, since interacting with graphics is a common occurrence in modern life. However, before any further analysis of the nature of diagrams can take place, a suitable conceptual framework is needed to define the problem context. The need for such framework becomes even more imperative when taking into account how ambiguously many of these terms are used throughout the literature. The specific meanings of terms such as “diagram”, “chart”, “graph”, etc. are certainly not categorical, and their interpretation varies highly according to the author of the text at hand.

Commencing from the most fundamental definition, an *image* or *graphic* is a static, two-dimensional artifact that depicts visual perception. Decomposing this definition into parts, images possess the following characteristics:

- They are *visual* i.e. perceptible by means of the human sight. Images may, however, be extended with help of other sensory information e.g. the auditory system in interactive sound diagrams.

- They are *two-dimensional* i.e. found on a more or less flat surface, be it a piece of paper, a computer screen, a wall, etc. Whereas a three-dimensional scale print of a car would not be considered an image, a photograph or drawing of the same car would be regarded as one.
- They are *static* i.e. not changing in time. This means that moving images found in film, interactive online graphics and other animated displays are not included in this definition. Each of the still images that make up the dynamic graphic would however fall under this definition.

Proposals concerning the classification of graphics in the literature are varied. At their broadest level they take into account the main purpose of the image i.e. the expected general use of an image by its audience. One of these classifications is given by Mayer and Gallini [MG90]. According to them, illustrations that accompany text can serve five different purposes:

- *Decorative*. They help the reader enjoy the content by making it more attractive, without being relevant to the text.
- *Representative*. Illustrations that help the reader visualize a particular event, person, place or thing, such as found commonly in narrative passages.
- *Transformative*. Graphics that help the reader remember key information in a text.
- *Organizational*. These illustrations help the reader organize information into a coherent structure.
- *Interpretative*. Illustrations that help the reader understand the text.



Figure 1.2: Grand Staff, a component of Western musical notation and a classic example of diagram<sup>2</sup>

Richards [Ric84] makes a more general division. He states that some graphics are created with the goal to delight the audience in some way. This is the case of most artistic graphical

---

<sup>2</sup>Image source: [https://en.wikipedia.org/wiki/Staff\\_\(music\)](https://en.wikipedia.org/wiki/Staff_(music))

## 1. INTRODUCTION

---

pieces such as paintings and photography. On the other hand, there are graphics which are created with the intention of informing the audience about a certain piece of knowledge. According to this differentiation, images can be broadly divided into two main categories; namely, **decorative** images and **informative** images. In this thesis, the focus will lay on the latter group, which many authors simply call **diagrams**. An example of diagram is the Grand Staff, a form of time chart, used in music notation and shown in Figure 1.2.

Many different definitions of diagram can be found on the literature e.g. [Eng02, Ric84, Max10, Gar79]. On its most basic definition, a diagram consists in “marks on a surface forming a static record, where the relations in the diagram are used to illustrate relations between other objects” [Ric84]. In this thesis, the terms *diagram*, *diagrammatic representation*, *graphic representation* and *knowledge representation graphic* are used interchangeably to refer to any image which has been created to express information, most commonly numerical information stemming from data tables, for organizational or interpretative purposes. Section 6.1.1 expands on the cognitive benefits of graphic representations to transfer knowledge.

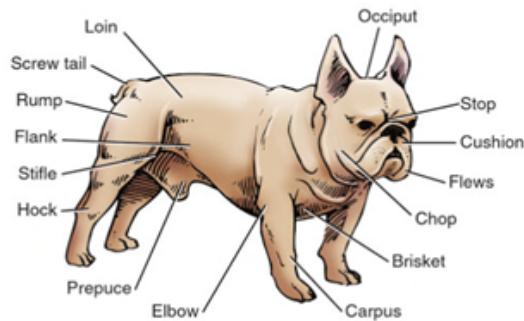


Figure 1.3: External topography of a French Bulldog [EM13]

It is important to note that often there is no clear cut distinction between categories, as pointed out by Richards [Ric84]. For instance, Figure 1.3 could be considered an informative diagram, since it is placed on a book about dog anatomy and informs the reader about the external topography of a certain dog breed. However, if the image was placed on a children's book with its arrows and their labels removed, then it could be employed as a decorative image. The same image without labels could be placed next to another dog breed's image in order to compare certain characteristics of both breeds, in which case said image would have an informative purpose again and could be considered a diagram. In Richards' words:

“One would not normally be disposed to use the term ‘diagram’ to refer to, say, a picture of a naked man. [...] However, one’s disposition might change if some labels were added and these were linked by means of connecting lines to various parts of the body shown in the picture. [...] The point I would like to argue is that the picture of the human figure is not itself a diagram. It becomes a diagram when we use the relations between the labels and the various parts of the pictured body to stand for other relations existing externally to the display” [Ric84].

Diagrammatic representations can be analogously classified into different types. Both the exact way of categorizing them and the terminology used throughout the literature varies greatly. Nevertheless, it is possible to identify common motifs among available diagram taxonomies. The main criteria used in most classifications of diagrams are based on combinations of either the type of syntactic structure that is involved in them or the type of information that is expressed by the diagram [Eng02]. Distinguishing between different types of graphic representations is important not only because of the obtained descriptive uniformity, but also because the disparate syntactic structures that underpin them change the manner they are processed by both human readers and computer systems, and therefore different Human-Computer Interaction (HCI) approaches need to be undertaken in the context of access to graphics of this thesis.

Engelhardt, on his analysis of existing graphic theories [Eng02], diagnoses recurrent concepts concerning how to classify graphic representations, and lays out a principled description of their commonly distinguished types. Accordingly, ten primary types and six hybrid types of graphic representations are identified. Namely:

- Primary graphic representations. These are the *map*, *picture*, *statistical chart*, *time chart*, *link diagram*, *grouping diagram*, *table*, *symbol* and *written text*.
- Hybrid graphic representations. These are made up by simultaneously combining two primary graphic representations. They are the *statistical map*, *path map*, *statistical path map*, *statistical time chart*, *statistical link diagram* and *chronological link diagram*.

We will not delve into the detailed descriptions for each graphic type here, but, given the focus of the present work on diagrammatic representations, a short summary of each diagram type will be given next. These definitions are an indispensable addition to the conceptual framework of the Natural Language Interface (NLI) to diagrams that will be developed in later chapters. The framework of Engelhard will be considered as the basis from which the definitions emanate and will be compared against other existing interpretations, especially the classic works of Bertin [Ber11] and Tufte [Tuf97]. Examples found throughout the literature will be given as well.

## Map

Maps are symbolic representations of physical geography. Their syntactic structure is based on an integral metric space that serves to represent a physical arrangement on a geographical surface [Eng02]. Maps are also considered an independent primary graphic type in most of the relevant literature, including the works of Tufte [Tuf01] (although he only considers “data maps” i.e. diagrams that combine “cartographic and statistical skills” and might have considered purely geographical maps as a subtype of the picture type [Tuf97]), Holmes [Hol93], Kosslyn [Kos94], Bertin [Ber11], and Lohse et al. [LBWR94]. The space within a map can be distorted to generate what some authors consider a distinct type of graphic representation, the *cartogram*, in which the size of the map’s divisions are related to the

## 1. INTRODUCTION

---

total quantities of the studied variable in said division instead of their actual geographic areas [LBWR94, Ber11].

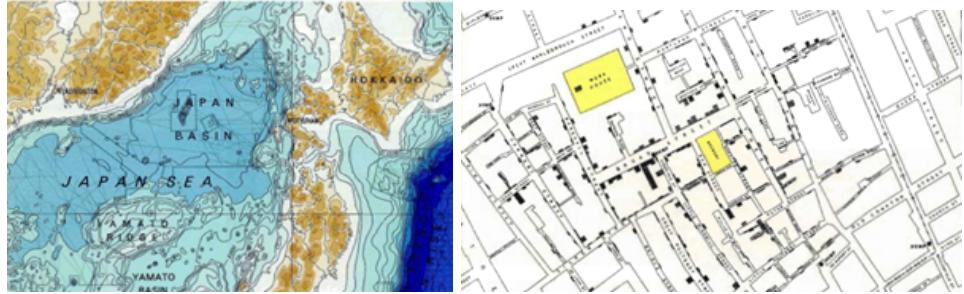


Figure 1.4: Examples of map [Tuf97]

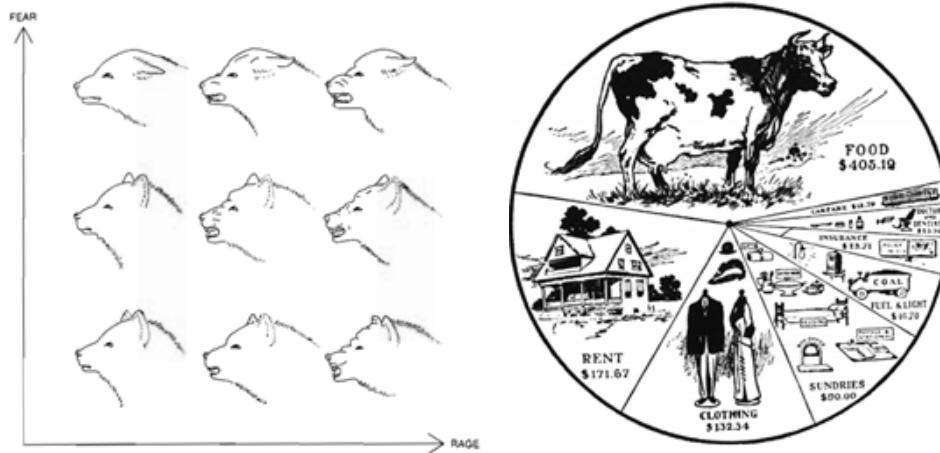


Figure 1.5: Examples of statistical chart [Tuf01, Eng02]

### Statistical Chart

A statistical chart is a graphic representation in which the syntactic structure serves to show quantities and allows comparing them [Eng02] i.e. diagrams encoding quantitative information using the position and magnitude of geometric objects [LBWR94]. This type of primary graphic representation is also acknowledged in the works of Kosslyn [Kos94] (where they are simply named “graphs”) and Tufte [Tuf01] (“relational graphics”). Usually, the numerical data is plotted in a Cartesian or polar coordinate system labelled by marks on metric axes. Proportional division of graphic objects may also be used to display the data, such as in pie charts or stacked bar charts. Finally, variations in visual attributes such as size or brightness among the graphic objects of the diagram may be employed as well. Examples of statistical charts include the *scatterplot*, *bar chart*, *line chart*, *histogram* and *Chernoff face graph*.

## Time Chart

Time charts, also known as *time series*, are graphic representations that display temporal data [LBWR94]. Their syntactic structure usually is based on a metric axis whose length stands metaphorically for length in time, or may be an ordered lineup of graphical objects [Eng02]. In any case, at least one of their dimensions marches along with the passing of time [Tuf01]. Most authors consider the time chart as belonging to the statistical chart category, even though some others consider it the most frequently used form of graphic design and therefore assign it to an independent category [Tuf01].

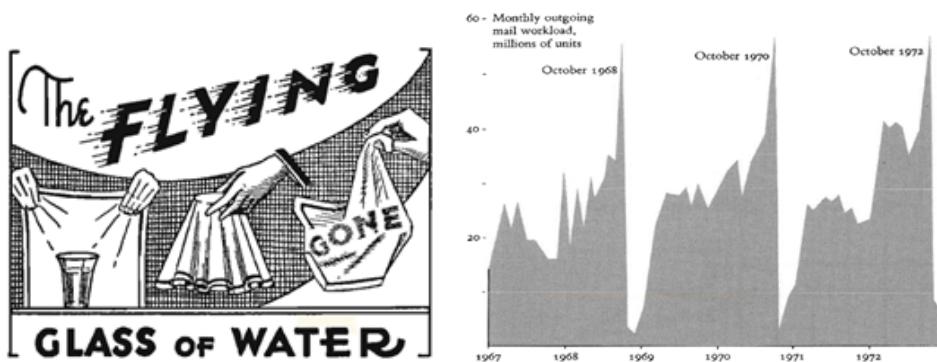


Figure 1.6: Examples of time chart [Tuf01, Tuf97]

## Link Diagram

Link Diagrams show the relationships among components [Loh93]. Their syntactic structure consists of linking relationships in which a connector is anchored to two nodes [Eng02]. Sub-types of link diagrams include [Eng02]:

- *Linear chains*: a linking configuration in which no branching occurs.
- *Circular chains* i.e. a linear chain forming a closed loop.
- *Trees*: a linking configuration that involves branching from one root and without closed loops.
- *Networks*: A linking configuration that involves one or more closed loops i.e. there is more than one possible way for moving from one node to another

Link diagrams are considered a primary graphic type by most authors, including Bertin [Ber11], Kosslyn [Kos89] and Lohse et al. [Loh93]. They are often referred to as *networks* throughout the literature. In this thesis the term *link diagram* will be the preferred one.

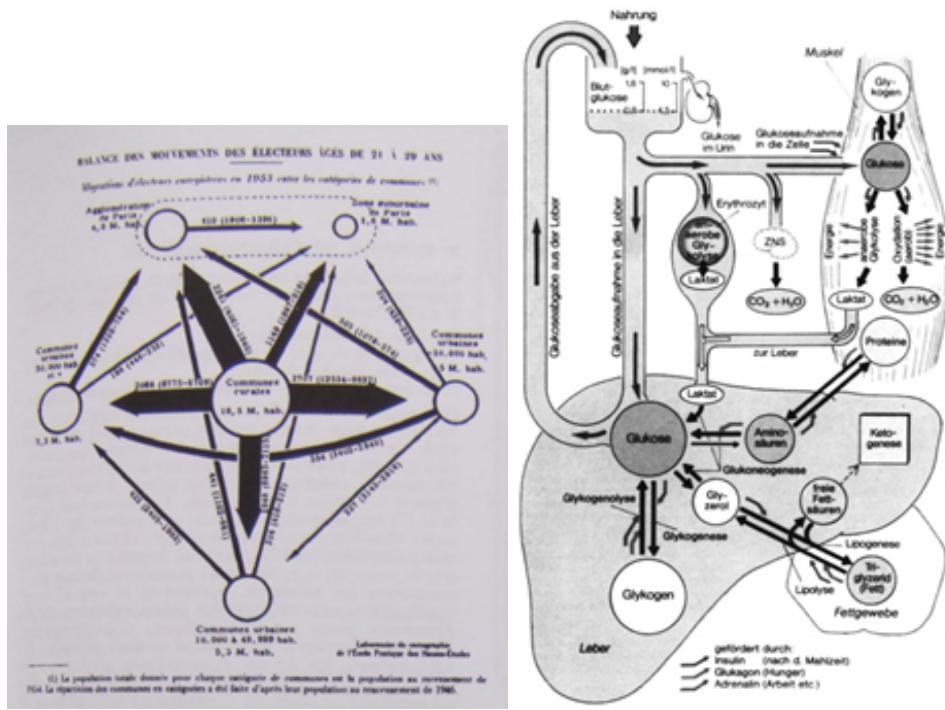


Figure 1.7: Examples of link diagram [Ber11, Eng02]

### Grouping Diagram

Grouping diagrams categorize elements into (possibly overlapping) sets. Their syntactic structure may make use of spatial clustering, separation by separators or containers in order to tell the different categories apart [Eng02]. Most authors consider this type of graphic a sub-type of the statistical chart primary type.

### Table

A table is a two-dimensional arrangement of words, numbers, graphical objects or combinations of them to exhibit a set of facts or relationships in a compact format [Loh93]. Its syntactic structure consists of a simultaneous combination of horizontal and vertical separations and/or simultaneous combination of horizontal lineups and vertical lineups [Eng02]. Tables are considered a primary graphic type by Lohse et al. [Loh93] and Engelhardt [Eng02]. However, most authors do not take them into special consideration. Some may acknowledge them as special kind of statistical chart (named “matrix”, [Ber11]) whereas others regard them as a simple arrangement technique more suitable for small numerical sets [Tuf01].

## Hybrid Graphic Representations

In addition to the primary types of graphic representation previously outlined, some graphic representations belonging to primary types may be integrated by means of simultaneous combination in order to create so-called *hybrid graphic representations* [Eng02]. Examples of hybrid graphic representations are the *statistical map* (a combination of statistical chart and map), the *statistical time chart* (a combination of statistical chart and time chart), the *path map* (a combination of link diagram and map), the *chronological link diagram* (a combination of link diagram and time chart), the *statistical link diagram* (a combination of statistical chart and link diagram), and the *statistical path map* (a combination of link diagram, map, and statistical chart). Some examples of hybrid graphic representations are shown in Figure 1.8. For more information, the reader is directed to Engelhardt's work on the matter [Eng02].

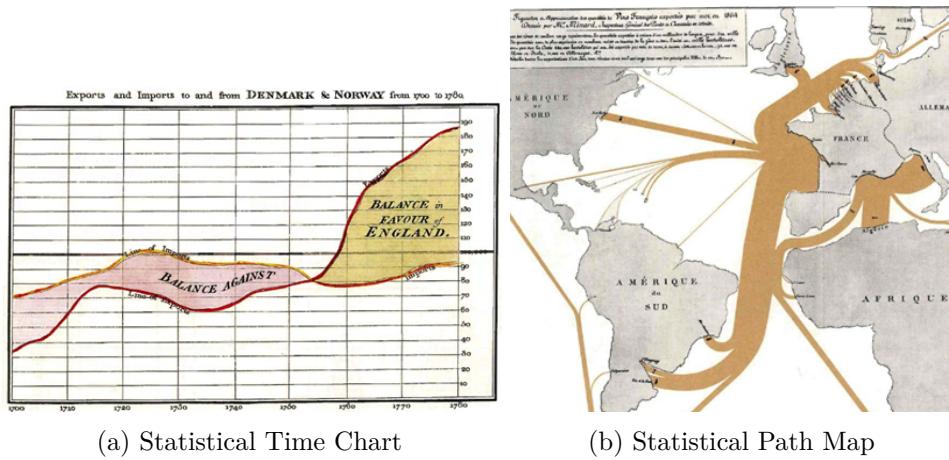


Figure 1.8: Examples of hybrid graphic representations [Tuf01]

## Nested Graphic Representations

Graphic representations of different types, including primary and hybrid types, may also be nested in order to generate more complex graphic representations [Eng02]. For example, a map can contain nested graphic representations such as other maps, statistical charts or pictures to express certain information about its geographical features, as seen in Figure 1.9. One of the most common ways of nesting graphic representations consists on a multi-panel display, usually arranged as a lineup or a table [Eng02].

## Dynamic Graphics

A final important consideration is that the scope of this taxonomy is limited to static diagrammatic representations. Even though, as introduced in the previous section, dynamic and interactive graphic representations are a growing field within the also growing amount of available electronic media, their analysis and adaptation for natural language access lays outside the scope of this thesis. Before such exercise can take place, static graphic

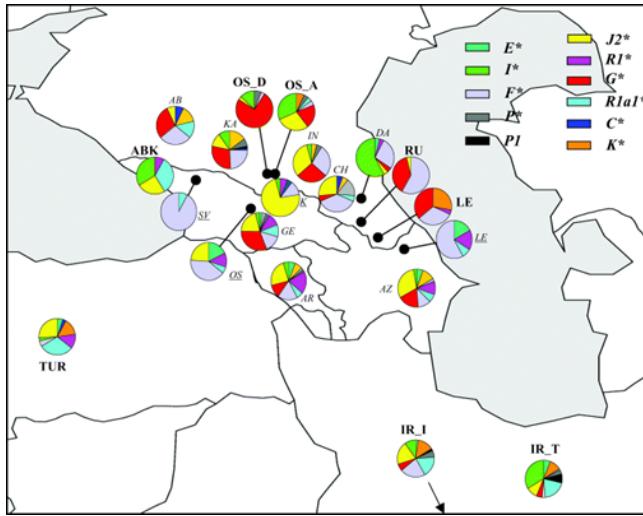


Figure 1.9: Chartmap of the Caucasus showing Y-SNP haplogroup frequencies [NLQ<sup>+</sup>04]. An example of a nested graphic representation, which contains a map and several statistical charts (pie charts).

representations need to be considered first, which the present work focuses on. In any case, the amount of static diagrams keeps growing [Eng02], and they still make up the majority of diagrams found on the Web.

## 1.2 Problem Statement

Graphics have been described as “the last frontier in accessibility” [Gar14]. Visually impaired persons need to make use of assistive technology and alternative representations of visually perceptible information in order to make up for the loss of the advantages that the ocular medium provides. But before continuing, let us characterize what is meant by “visually impaired”, “blind”, and so forth. These definitions characterize the main stakeholders of this research; namely, those persons with an existing or potential interest in accessing diagrammatic representations non-visually.

Vision loss is described in detail by classifying it into multiple ranges based on visual acuity. There are several methods for measuring visual acuity, one of the most popular being Snellen letter charts. According to the categories defined in ICD-9 and ICD-10 by the World Health Organization (WHO), a person is considered blind if he or she has a visual acuity of less than 0.05 [Int]. Someone with a visual acuity of more than 0.05 but less than 0.3 is considered to have low vision. The term “visually impaired” persons as it appears in this text refers to those who are either blind or have low vision.

In 2010 there were an estimated 285 million visually impaired people in the world, of which 39 million were blind and 246 million had low vision [PM12]. Table 1.1 displays the detailed distribution of visually impaired people by world regions in 2010. The principal causes of

		Blindness	Low Vision	Visual Impairment
WHO Region	Total Population (millions)	No. in millions (%)	No. in millions (%)	No. in millions (%)
Africa	804.9 (11.9)	5.888 (15)	20.407 (8.3)	26.295 (9.2)
Americas	915.4 (13.6)	3.211 (8)	3.401 (9.5)	26.612 (9.3)
Eastern Mediterranean	580.2 (8.6)	4.918 (12.5)	18.581 (7.6)	23.499 (8.2)
Europe	889.2 (13.2)	2.713 (7)	25.502 (10.4)	28.215 (9.9)
South-East Asia (except India)	579.1 (8.6)	3.974 (10.1)	23.938 (9.7)	27.913 (9.8)
Western Pacific (except China)	442.3 (6.6)	2.338 (6)	12.386 (5)	14.724 (5.2)
India	1181.4 (17.5)	8.075 (20.5)	54.544 (22.2)	62.619 (21.9)
China	1344.9 (20)	8.248 (20.9)	67.264 (27.3)	75.512 (26.5)
<b>World</b>	<b>6737.5 (100)</b>	<b>39.365 (100)</b>	<b>246.024 (100)</b>	<b>285.389 (100)</b>

Table 1.1: Number of people visually impaired and corresponding percentage of the global impairment by WHO region and country, 2010 [PM12]

visual impairment worldwide are uncorrected refractive errors (43%) and cataracts (33%); 18% of the causes being undetermined. The main causes of blindness are cataracts (51%), glaucoma (8%), AMD (5%), childhood blindness (4%) and corneal opacities (4%), with 21% of undetermined causes [PM12]. These causes greatly vary from region to region according to their degree of industrialization, which in turn affects the quality of the available public eye care services. In any case, efforts to combat preventable blindness ought to be made in every region [KR01].

In Europe, an average of 1 in 30 persons experience sight loss [Eur17]. Permanent sight loss is third most frequently mentioned disability in Austria, with 3.9% of the Austrian citizens reporting it according to *Statistik Austria* [LB08]. According to other reports, in 2007 about 3,000 people considered themselves as completely blind in Austria, while 101,000 presented profound vision impairment [Bli07].

Full access to information is paramount if visually impaired persons are to take a full role in society. Visually impaired people need all the information that fully sighted people need: information about their rights and entitlements, information that enables them to make rational choices as consumers, and information to support them in their work, their learning and their leisure. In addition to these common informational necessities, visually impaired people need information that relates to their position as disabled people and as visually impaired people e.g. information about their specific rights and entitlements, their condition, available aids and self-help groups, etc. [Moo00].

Blindness can greatly influence the educational and professional decisions of affected persons.

## 1. INTRODUCTION

---

The Treaty of Lisbon, signed by all European Union member states, underpins the importance of equal access to education for economic growth, mobility, inclusion, active citizenship, employability, European integration and democratic development [Cou07, PM15]. In spite of this, blind students are less likely to choose a field related to science, technology, engineering, and mathematics (STEM) for their graduate studies than their sighted counterparts, as studies such as law or social sciences are regarded as “easier to handle and finish” [PM15]. Meanwhile, STEM-related occupations were expected to grow 7.2 percent more than non-STEM-related occupations in the period from 2008 to 2018, while being better paid on average, having a significantly lower unemployment rate, and bearing a higher impact on a nation’s competitiveness, economic growth, and overall standard of living [LMB<sup>+</sup>].

Graphics are widely used as didactic materials supporting teaching STEM disciplines and are central to their clear understanding [Sha14]. Examples are countless and may include diagrams used to give the structure of chemical molecules in chemistry, charts employed to summarize results of scientific research, and candlestick charts employed in financial analysis. Therefore, proper access to visual information is vital in order to fully integrate students in the education system and society. Lack of access to diagrammatic representations has been identified as one of the major reasons why students who are blind or visually impaired are kept out of science classes [Sha14]. Closely related to the problem of accessing graphics in educational settings is the problem of access to mathematics. Many mathematical expressions are also represented in two dimensions in a way that aids students with their recognition and understanding [BMS08]. Moreover, teaching math in primary schools generally involves using visual didactic information in the form of pictorial and implicit spatial representations, negatively affecting the learning ability of blind and visually impaired students [JPSM16]. Other aspects of mathematics, such as geometry or graph theory, are inherently visual, and are learnt more efficiently with the support of graphics in both the context of a classroom and in self-study.

This negative trend could however be corrected by providing students with materials, methods and strategies following their specific needs [PM15, Fer06]. Even though it is still a commonly held opinion that diagrams are not suitable for presentation in a non-visual manner, research suggests that the opposite is the case [Ben02]. The outcomes related to academics and learning of many visually impaired students can be improved by the use of assistive technologies and best practices with foundations in scientifically based research. However, teaching practices undertaken in public schools are more often than not based on tradition, superstition, anecdote, and common sense rather than science [Fer06]. Finding alternatives to traditional Braille-based accessible educational materials is paramount in order to improve the aggregation of visually impaired students into inclusive educational settings. In turn, this would lead to better opportunities of these students for inclusion in society [GSG<sup>+09</sup>].

Lack of access to visual information does not only negatively affect blind and visually impaired persons during their education, but also in their subsequent careers. Visually impaired persons are much less likely to be employed than sighted persons, with an average unemployment rate of over 75% for blind and partially sighted persons of working age [Eur17]. For instance, blind programmers face exclusion from the software analysis and design process [BE98].

Even though programming languages are largely text-based and therefore fairly accessible, graphical methods are commonplace in the design and analysis stages of software systems development, such as Unified Modeling Language (UML), Ward-Mellor RTSA, and other structured diagrams. Braille alternatives of these diagrams, even of the simplest ones, may run for several pages of Braille, requiring blind programmers to possess a higher deal of skill and handle a greater cognitive load than those of their sighted colleagues. Moreover, the supporting models change very frequently during the review stages of a software project, rendering traditional approaches useless, as they take a long time to be produced. These disadvantages can greatly affect the careers of blind professionals in software engineering and other technical fields.

### 1.3 Purpose

The Problem Statement section has manifested the need for efficient, effective, and user-friendly mechanisms so that Visually Impaired Persons (VIPs) may access graphical information, and, more specifically, diagrams, in a usable manner. Nowadays, VIPs must resort to external help from sighted persons, assistive technologies and domain-specific methodologies in order to obtain a non-visual alternative to diagrams; approaches which, as it will be discussed in Chapter 2, are currently insufficient for their access needs.

Assistive technology (AT) in the context of this thesis refers to technologies that promote better communication between humans with impaired exteroceptive senses perception (such as sight or hearing) and computing devices [Mur08]. Examples of assistive technology include Braille printers and embossers, screen magnification software, mouth-operated joysticks, and educational assistive software for people with learning disabilities. Broader definitions might include all kind of assistive, adaptive, and rehabilitative devices for people with disabilities, including wheelchairs, hearing aids, and prostheses<sup>3</sup>. The visually impaired make use of assistive technology that almost always concentrates on the senses of touch and sound to compensate for limitations with sight. Those that focus on sound are referred to as *auditory devices* (the process of adapting visual information to sound is called *sonification*), whereas those focusing on touch are named *haptic* and *tactile devices* [Mur08]. Both approaches may also participate together and support each other in the same device, in which case they are named *hybrid* or *multi-modal* devices.

Assistive technologies that support visually impaired persons have the capability of greatly improving their academic and professional outcomes in multiple disciplines, e.g. math [BFJ<sup>+</sup>11], software development [BE98], and even the visual arts [Arn90]. However, as we will discuss in greater depth in Chapter 2, current assistive technologies are an efficient means of accessing linear information only. When it comes to nonlinear information, such as diagrams, current assistive technologies are either too focused and domain-specific, expensive or cumbersome to produce, or present a way too steep learning curve. Additionally, they usually lack the interaction and customization capabilities that would allow their users to employ them in an autonomous manner i.e. without the external support of a sighted person.

---

<sup>3</sup>[https://en.wikipedia.org/wiki/Assistive\\_technology](https://en.wikipedia.org/wiki/Assistive_technology)

*Communicative images* are a novel approach to accessibility of graphics that may help solve many of the drawbacks of traditional Assistive Technology (AT) for a sizable number of graphic domains. They are defined as “two-dimensional objects integrated with a dialogue interface and being equipped with an associated knowledge database” [KO11]. These semantically-enhanced graphics can communicate with the user and also learn from the communication process and enlarge its underlying knowledge base accordingly. Image semantics are described in a structured way by means of ontologies using semantic categories, their properties and relationships, which prevents semantic inconsistency, duplication and generally preserves the reliability of the underlying knowledge representation.

Ontologies classify and confer meaning to the depicted objects in the graphic by linking them with concrete semantic categories and assigning concrete values to properties [KO11]. For example, an individual of the ontological class `Person` can annotate a certain graphical object in the graphic (e.g. a photograph), depicting a person. This individual may then be assigned the value `John` for its name property. The individual could then be related other individuals in the image through object properties (e.g. `IndividualA → hasFather → individualB`), and so forth. When every graphical object and relationship in the graphic is annotated in such manner, the complete image semantics can be embedded into the image itself and accessed by external applications to perform complex inference on the image contents. A dialogue interface may then enable users to efficiently retrieve the underlying knowledge by means of queries in natural language. Ontology-supported dialogue systems separate domain knowledge and linguistic features, reducing the complexity of the rest of their modules (e.g. the semantic analyzer and dialogue manager modules) thereby improving the efficiency of the Information Retrieval (IR) process [Plh14]. The dialogue manager subsystem processes the user queries and returns the relevant requested data to the user. Communicative images are reviewed in depth in Chapter 2 of this dissertation.

The grammatical structure of language and the graphical structure of diagrams present many parallelisms that suggest an innate likeness between the two of them [Ric84], making natural-language based access to diagrams a promising approach. The serial nature of speech has an obvious disadvantage over well-designed alternative approaches which do not impose such a cognitive load on the reader’s processing capabilities. Interestingly enough, despite of this fact blind readers much prefer speech feedback when reading e.g. link diagrams, and other means are preferred only rarely [FLST13]. However, research on natural-language-based access to graphics has been limited to real-world photography, leaving out diagrammatic representations from its domain of applicability.

Therefore, this dissertation suggests a natural-language based access approach to knowledge representation graphics stemming from the principles of Communicative Images to solve the problem of a lack of an effective, autonomous, non-visual access method to diagrams outlined in the previous section. Figure 1.10 displays a high-level overview of the proposed framework. Each of its constituent elements will be thoroughly described in the remaining of this document. An abridged and Web-based prototype of the proposed framework has been developed and evaluated with VIPs.

The findings of this work will hopefully stimulate further research towards a holistic, natural-

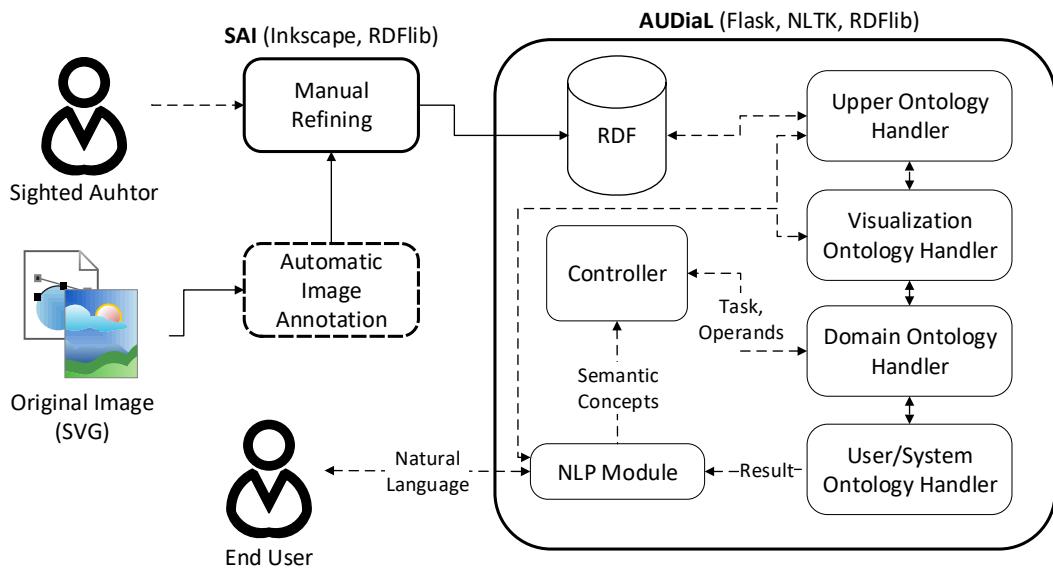


Figure 1.10: High-level block diagram of the proposed framework for natural-language-based access to graphics

language based, ontology-supported framework to accessibility of general graphics. The potential future integration of improved computer vision and automatic object recognition methods into such a framework, in addition to powerful state-of-the-art Natural Language Processing (NLP) capabilities, presents an exciting opportunity for blind and visually impaired people worldwide for their greater inclusion in the information society.

## 1.4 Research Questions

Having defined the problem context and the purpose of this work to investigate it and to come up with potential solutions, this section states the aims of the present research as specific questions and hypotheses. It must be noted that, even if the questions are listed here together for convenience, there exists a temporal interdependence between them, as some questions follow from answers to other questions. For instance, by following the Design Science methodology described in next section, a design activity (e.g. designing a NLI prototype) may be motivated by the need of answering a knowledge question. The potential contributions of this thesis are the derived answers to these questions, each of which constitutes a novel contribution to the current state of accessibility research.

- 1. What are the benefits and drawbacks of current approaches to non-visual access to diagrams?** Before embarking on the pursue of new approaches, the state-of-the-art literature on non-visual accessibility to knowledge representation graphics needs to be studied. Which domain-specific approaches are currently set into place? Are they

## 1. INTRODUCTION

---

sufficient to solve the problem? Does any effective holistic approach exist? This question corresponds to the Problem Investigation phase prescribed by the Design Science Methodology, and is the focus of Chapter 2: State of the Art.

2. ***Are Natural Language Interfaces (NLIs) a suitable means for accessing statistical charts non-visually?*** Until now, the universe of discourse of communicative images has been constrained to photographic domains, in which a NLI is used to retrieve information about the visual features, such as position or size, and other characteristics of real-world objects depicted on a semantically-enhanced graphic. The main hypothesis of this thesis is that NLIs are also an usable means for accessing statistical charts by blind persons. This is the fundamental knowledge question of this research, and it concerns this thesis in its entirety.
3. ***How can the semantics of diagrams be formalized for their efficient natural language-based retrieval?*** All relevant information characterizing a diagram and its underlying data must be formalized so that it can be stored and interpreted by a computer. In the case of real-world photography, ontologies have proven to be a fitting manner of storing the semantics of what they depict and retrieving them through a NLI. Can an analogous approach be employed in the case of diagrams? How should it be adapted to allow effective non-visual access to diagrammatic representations, including the undertaking of high-level inference and navigation tasks? The aim of Chapter 3: Formalizing Visualization Semantics is to investigate and answer this knowledge question.
4. ***How can semantically-enhanced graphics be authored effectively?*** The formal underpinnings of a so-called semantically-enhanced diagram generally consist of a large amount of ontological resources (object instances and properties), since every significant graphic object of the diagram, their constituent sub-elements and the relevant relationships between them have to be formalized and associated to visual primitives. Automatic segmentation and object detection techniques may be employed to annotate them in an autonomous and fast manner. However, these techniques are still much less reliable than the human visual system, and some sort of manual authoring is thus required. Accordingly, it is necessary that authors of semantically-enhanced graphics can create them in an efficient manner using appropriate tools. This knowledge question involves the design of such an authoring tool, a process described in Chapter 4: Semantic Annotation of Diagrams.
5. ***How should a Natural Language Interface (NLI) be adapted to the specific needs of blind persons accessing diagrams?*** Diagrams provide sighted readers with perceptual, cognitive, and memorial benefits that make them such a commonplace means for depicting data (c.f. section “Visualizing Information”); benefits which evidently cannot be exploited by blind persons. This knowledge question is concerned with finding out how, and to which extent, can the lack of these cognitive benefits be compensated for in the design of an accessible NLI to diagrams. Designing such a system with these considerations in mind would result in accessible diagrams which are as functionally

equivalent as possible to their visual counterparts. This knowledge question is examined in the first half of Chapter 6: Accessibility and Validation of NLIs to Diagrams.

## 1.5 Method

This research follows closely the *Design Science* Methodology described by Roel J. Wieringa [Wie14] for both finding answers to the research questions posed in the previous section and designing the artifacts necessary for their resolution, such as the pivotal Natural Language Interface (NLI) to diagrams software prototype. Design Science is briefly defined as “the design and investigation of artifacts in context” [Wie14]. As displayed in Figure 1.11, design science research iterates over two problem-solving activities; namely, solving design problems and answering knowledge questions. These two activities may also be referred to as *designing* and *investigating*, respectively. In the context of this thesis, the main design problem (i.e. designing an accessible NLI to diagrams prototype) is motivated by the need to answer some of the knowledge questions posed in Section 1.4.

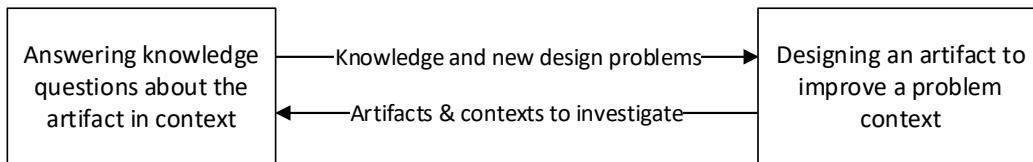


Figure 1.11: Design science research iterates over solving design problems and answering knowledge questions. Figure adapted from [Wie14].

As a matter of fact, the entirety of the knowledge questions previously listed can be understood as being encompassed by the seminal, generic design problem of *improving current approaches to non-visual access to diagrammatic representations*. Such undertaking commences with a *problem investigation* phase, which is the first task of the three-phase *Design Cycle*, depicted in Figure 1.12. In turn, the Design Cycle is part of the greater *Engineering Cycle*, which adds a fourth phase to the cycle, namely the *treatment implementation* task. However, given that design science projects are always restricted to the first three tasks of the engineering cycle (i.e. the Design Cycle) [Wie14] we will not discuss it further, and the focus of this section will be placed on the three tasks of the Design Cycle in the context of the present research.

The first task of the Design Cycle (namely, *problem investigation*) is made up of knowledge questions about the problem and its context. The research goal of this initial task is to “identify, describe, explain and evaluate the problem to be treated” [Wie14] before any artifact has been designed and when no requirements for an artifact have been identified yet. Some knowledge questions of this initial phase have been already answered in this chapter. Namely, the principal stakeholders (visually impaired persons) were defined in Section 1.2; whereas existing high-level goals have been discussed in Section 1.3. The expected main contributions of this thesis are the answers to the research questions listed in Section 1.4.

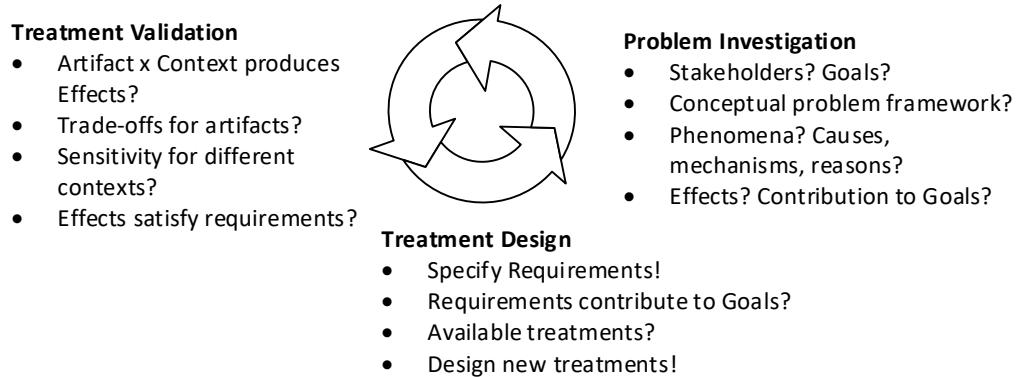


Figure 1.12: The Design Cycle of the Design Science methodology is made up of three main tasks; namely, Problem Investigation, Treatment Design, and Treatment Validation. The question marks indicate knowledge questions, and the exclamation marks indicate design problems. Figure adapted from [Wie14].

The remaining of the Problem Investigation phase, detailed in Chapter 2: State of the Art, consists in an in-depth literature review of existing approaches to non-visual access to graphics. This process was preceded by the design of a small taxonomy of existing assistive technology according to their output modality. Namely, concrete ATs were classified according to whether their output was offered in linear or multidimensional form, and by the main sense employed by users to access said output (touch, hearing, or both). Current methods to accessibility of graphics were then categorized accordingly and explored and analyzed; lastly, their benefits and drawbacks with relation to our problem context were described. The conclusion of this task can be briefly resumed as follows: available methods are insufficient for effective, efficient, and user-friendly non-visual access to diagrams by visually impaired persons due to a lack of user autonomy, low output resolution, inaccuracy of output, working memory overload, high cost, complicated use requiring training, or a combination thereof.

Problem investigation is followed by a *treatment design* task, which is the major undertaking of this research. Problem investigation acknowledged the potential of NLIs as a suitable approach to enabling VIPs access diagrams. However, to the best of our understanding, no such system had been developed yet. This outcome provoked the main design problem of this thesis, already introduced at the beginning of this chapter. Namely, to *define a framework for authoring and accessing diagrammatic representations of data non-visually by means of natural language, with a special focus on statistical charts*. This design problem, in turn, consists for the most part in creating the artifact that will be employed to investigate the principal knowledge question laid out in Section 1.4, namely, whether NLIs are a suitable means for accessing statistical charts non-visually. The design of this research artifact, named *Accessible*

*Universal Diagrams through Dialogue (AUDiaL)*, is the main undertaking of Chapter 5.

This resolution gave rise to two additional design problems, narrower in scope than the main design problem, which stem from the need of answering two additional knowledge questions. Namely, these are questions 3 and 4 from Section 1.4. These sub-problems are shown in context in Figure 1.13. The first one (Chapter 3) consists in the design of a suitable conceptual model underpinning the semantics of diagrams, at different levels of abstraction, so that high-level inference and analytical tasks could be carried out by AUDiaL in an effective manner. As usual, this design problem started with a Problem Investigation phase (Section 3.1), where existing visualization formalization mechanisms were studied. After resolving that ontologies are the most suitable means for encoding diagram semantics, the relevant conceptual framework was outlined in Section 3.2 and Section 3.3. The requirements that the conceptual model had to adhere to were defined in Section 3.4. Lastly, the design of the suitable conceptual model (a hierarchy of ontologies for visualization) is laid out in Section 3.5.

The second design sub-problem, tackled in Chapter 4, is related to the problem of efficiently authoring graphics enhanced with formal semantics. The conceptual framework of this thesis is first expanded with concepts from digital image formats in Section 4.1 and Section 4.2, as well as concepts specific to the proposed *semantically-enhanced graphics* in Section 4.3. A semantically-enhanced graphic consists in an image with associated semantics in the form of ontologies which are transparently embedded within the graphical content, thereby enabling cognitive and inference tasks supported by visual information to be carried out effectively. The Problem Investigation phase related to existing methods for semantically annotating images is laid out in Section 4.4. It was concluded that current methods are insufficient for authoring semantically-enhanced graphics efficiently in the context of this thesis. This conclusion resulted in the design and validation of *Semantic Annotator for Inkscape (SAI)*, an authoring tool for semantically-enhanced graphics described in Section 4.5.

Chapter 6 is concerned with completing a full iteration of the main Design Cycle by conducting two distinct tasks. First, the Treatment Design phase of the main research artifact (the AUDiaL prototype) is completed by adapting it to the special access needs of its target group, namely Visually Impaired Persons (VIPs). Section 6.1 commences this process by analyzing the cognitive benefits that diagrams provide to sighted readers. Several HCI techniques are proposed in the subsequent sections that aim to make up for the lack of these benefits when navigating diagrams non-visually through a NLI. The technical accessibility features of AUDiaL as a Web application that must be accessible to screen reader users are laid out in Section 6.8. Lastly, the Treatment Validation phase of the prototype, in which AUDiaL is validated with end users is described in detail in Section 6.9 and sections following it.

Chapter 7 concludes this thesis. Section 7.1 gives the reader once again a short general overview of the thesis and its problem context. Section 7.2 summarizes the main contributions of this research, which are the inferred answers to the knowledge questions originally posed. Their answers are gathered from the results of the relevant research tasks undertaken throughout this thesis. However, the execution of this work also gave raise to a number of issues that could not be tackled or solved successfully, or which present room for improvement.

## 1. INTRODUCTION

---

These aspects are discussed in Section 7.3. Lastly, Section 7.4 introduces a number of interesting avenues for future work based on the contributions on this thesis.

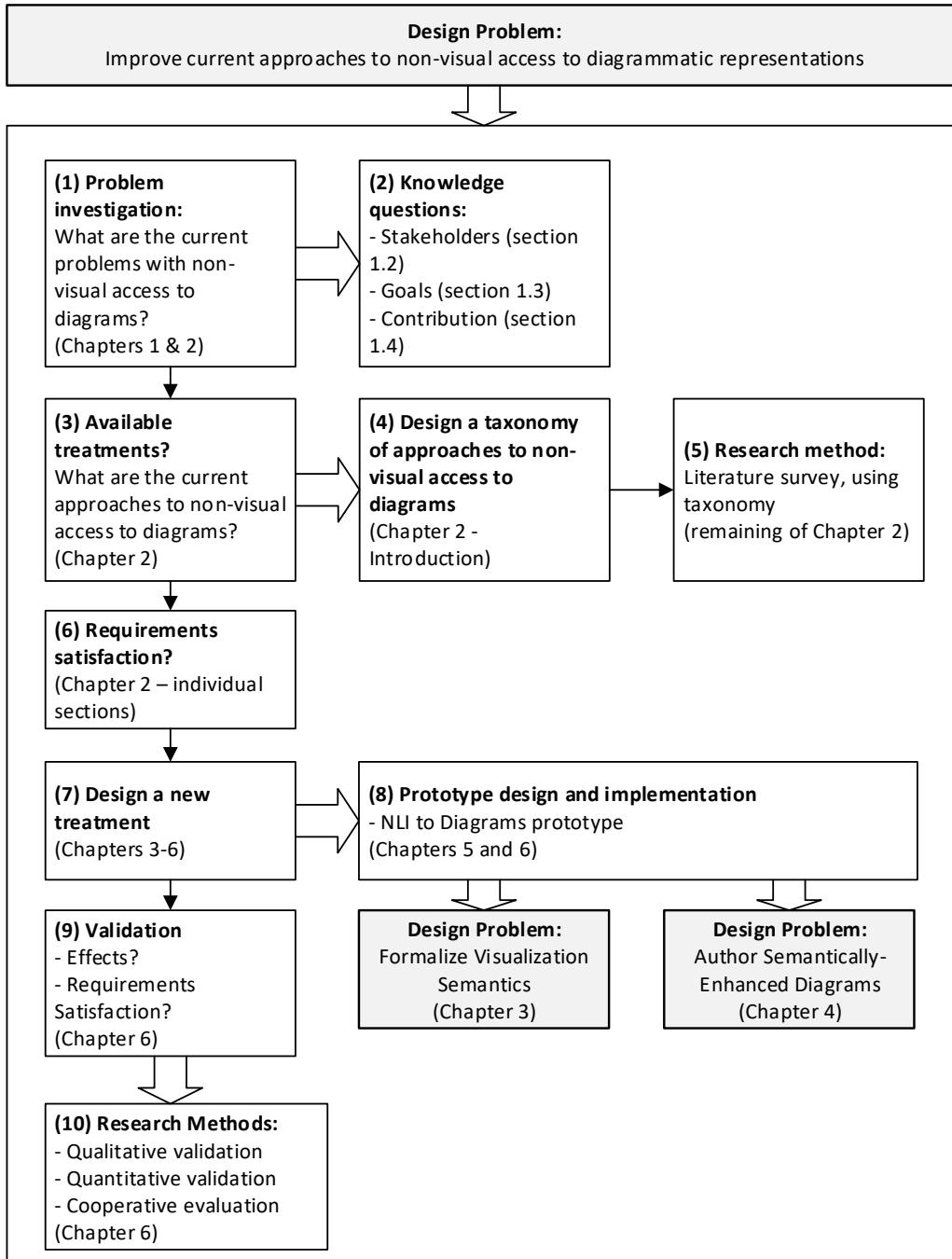


Figure 1.13: Methodological structure of this thesis following the Design Science approach. Note that not all knowledge questions are shown, as some arose from problem solving tasks. Figure adapted from [Wie14].



# CHAPTER 2

## State of the Art

This chapter analyzes existing approaches in non-visual access to generic graphics with a primary focus on diagrammatic representations, noting their benefits and drawbacks in the context of the research problem described in the previous chapter. This process takes up the majority of the *Problem Investigation* task of the primary *Design Cycle* of the present research, an endeavour that was described in Section 1.5. The conceptual framework used to frame the present research problem will be augmented with concepts related to human physiology, Assistive Technology (AT), and accessibility. In turn, generalizations about patterns ascertained during this task will be employed to justify the design of the main artifact of this thesis; namely, a Web prototype of a Natural Language Interface (NLI) to semantically-enhanced diagrams.

Sense	Limit bandwidth (bits/s)
Eye	$10^6$
Ear	$10^4$
Skin (vibrotactile)	$10^1$

Table 2.1: Limits of information bandwidth for three senses [Kok87] (with revised data from [WB97b]).

Humans possess two types of sense, exteroceptive and interoceptive. Among the exteroceptive senses, external senses allow us to perceive information from the external world. External senses are known individually as sight, taste, smell, touch, and hearing. Taste and smell are extremely specialized senses with a limited informational bandwidth (loosely defined as the capacity of a sense to receive and perceive information) [DB74]. Moreover, they are prone to grow accustomed to frequent stimulus and become less sensitive to it over time [WB97a]. Therefore, alternative approaches to visually perceived information have to be

confined to the remaining senses of touch and hearing to compensate for sight limitations. The approaches considered in this chapter will thus be divided into those that make use of touch (*tactile* and *haptic approaches*, also known as *tangible media*), those that leverage sound (*auditory approaches*, including *sonification* and *speech interfaces*), and those that combine both means of perception (*hybrid* or *multimodal approaches*).

Table 2.1 displays the upper limit in information bandwidth for the three exteroceptive senses discussed in this chapter, showing how other senses fall way below the informational capacity of the visual sense. Non-visual alternatives of graphics are therefore limited in nature, and they generally benefit from a simplification of the visual content by some bandwidth-correlated scaling factor [FWB96, WB97a]. Because, as introduced in Section 1.1.2, problem solving involves the use of both sentential and diagrammatic representations; and linear information can be efficiently acquired by tactile and auditory means, this chapter will begin by spelling out current sentential (i.e. linear) alternative representations of graphics for accessibility purposes. In addition, approaches supported by means of dialogue are a special case of linear-auditory approaches that will be discussed separately. Communicative images illustrate an interesting dialogue-based approach that will be described in greater detail as they comprise the foundation upon which the suggested framework described in this thesis (Section 1.3) is built.

## 2.1 Linear Approaches

Text and other linear information, such as linear mathematical formulae, can be accessed by blind persons with relative ease using Braille prints or modern assistive technology such as text-to-speech (TTS) synthesizers and refreshable braille displays. These technologies are described in Section 2.1.3 and Section 2.2, respectively. Typically, Braille copies of books and other print materials are individually ordered by blind readers or sighted professionals working with blind students. Nowadays, the availability of accessible electronic documents has significantly increased access to textual material to blind and Visually Impaired Persons (VIPs). Electronic documents are especially useful for those with vision-related cognitive disabilities, as they can be presented in a simplified layout containing special fonts, word highlighting and audio synchronization [LPSZ08]. On the other hand, speech synthesis is most commonly used as a non-visual way of accessing on-screen text in virtual environments with the support of AT. The linear nature of text makes converting it to Braille characters or speech an almost trivial task. Likewise, reading accessible text can be achieved in an efficient and natural manner by VIPs, in spite of the comparatively low rate of information perception by means of touch and hearing when compared to sight [KMS<sup>+</sup>06].

### 2.1.1 Linearizing Graphic Information

Some spatial information can be linearized and displayed as text to a blind person in a relatively satisfactory manner. For example, modern screen readers (c.f. Section 2.1.5) can linearize the two-dimensional HTML markup present in well-designed Web sites (excluding pictures and other graphical information) by stripping it of its markup tags and displaying

the textual content line by line on a refreshable braille display or by speaking the different parts out through a speech synthesizer. Graphical User Interfaces (GUIs) present in modern operating systems can also be accessed with relative ease by means of speech and Braille through the use of a screen reader, as it will be discussed later in this chapter. Besides sentential information and user interfaces, all kinds of information can be at least partially described by means of verbal descriptions, such as oral explanatory narrations given by a sighted person or Braille legends supporting works of art in museums. Natural language can be generally regarded as a suitable means of describing information at all levels of abstraction, and therefore most of what we know and perceive (including photographs, diagrams, and other visually displayed information) may be described, not without limitation and in varying degrees of realization, in static text or through human conversation.

Regarding diagrammatic representations, there are very few available standards and best practices for the creation of accessible verbal descriptions. Some of them are included in the Section 508 standards of the US General Services Administration's Office of Government-wide Policy [U.S], the HTML5 specification by the World Wide Web Consortium (W3C) [Wor11b], the WebAIM guide on techniques of providing alternative textual descriptions [Web17], the National Center for Accessible Media's guidelines for description of science content within digital talking books [GOF08], and the Description Key guidelines from the Described and Captioned Media Program [KS08]. For further information, a summary of best practices suggested by these guidelines can be consulted in [Kea14].

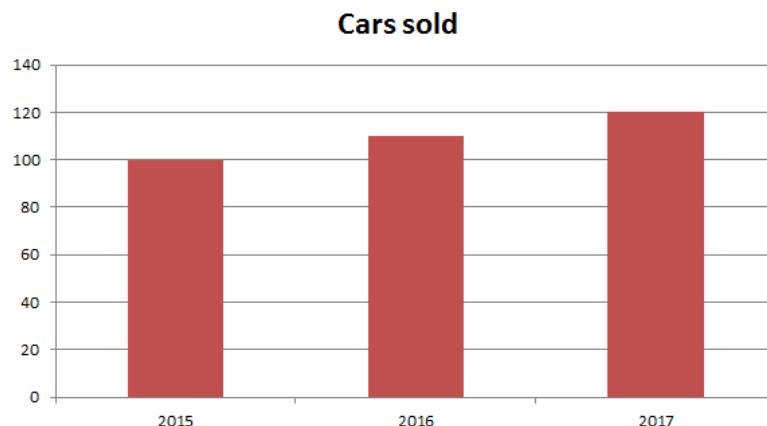


Figure 2.1: A simple bar chart. Simple diagrams such as this one are sufficiently described by short textual descriptions or by data tables.

Textual descriptions are however not functionally equivalent to their graphic counterparts [GM11], even when best practices are followed. Most of the cognitive benefits of diagrammatic representations for sighted readers (characterized in Section 6.1.1) vanish when they are textually represented, and with them the purpose for using the graphic that its author had in the first place. Verbal descriptions of diagrams can be confusing and difficult to understand, easily overload short-term memory, and do not provide any means of gaining an overview

## 2. STATE OF THE ART

---

of data [BPS03]. Nevertheless, photographs, pictures, and very simple diagrams may be sufficiently described by means of textual descriptions satisfying the informational needs of blind readers [Gar14]. For instance, for the majority of the population it would not be difficult to grasp and mentally retain the following information: “Number of cars sold by year: 100 in 2015, 110 in 2016, and 120 in 2017”. Figure 2.1 displays an equivalent visual representation of this information in the form of a bar chart. In this case, the previous text would suffice as an alternative for the graphic, as none of the cognitive advantages stemming from its visualization would make a big difference in its understanding due to the small mental model that is required to analyze it. For example, figuring out that there has been an increasing trend in the number of sold cars can be almost immediately perceived by either seeing the graphic, or reading its equivalent textual description or data table.

As statistical charts are used to represent data, this raw data itself may be provided in a structured form, such as tables or outlines, as an alternative to the diagram. However, complex diagrammatic representations may require thousands of words and numbers to be described to its full extent, which would make performing any sort of typical analytical task supported by it extremely challenging if not impossible. Current research suggests that an average adult is able to mentally process a maximum of 10 data points [Kea14], a number which certainly falls way below the amount of individual data points present on the average diagram. Alternatively, data analysis techniques which employ data reduction may be used to simplify the amount of information that needs to be processed by the reader. For example, classical statistical procedures widely used in science and technology take all data points and reduce them to a small number of statistics, such as means, standard deviations, correlation coefficients, variance components, and t-tests. The reduced information may then be more easily exploited in order to make useful inferences about the data as a whole. However, this is a very limiting approach that cannot convey the wealth of information that exists in data [Cle94]. Even though numerical reduction methods are useful in giving a quick overview of certain graphic representations, they are not a universally suitable means for non-visually accessing them. Moreover, many diagrammatic representations, such as maps, cannot be summarized using these methods. Generally speaking, if the graphic would not make sense if it were represented as a visual table, then it should not be presented as a data-only version [Kea14].

Consequently, complex diagrams are converted into domain-specific accessible counterparts that employ haptics, sound, interactive software, or a combination thereof, aiming to preserve some of the cognitive benefits that would vanish otherwise when using a linear approach, with the goal of enabling the reader form a better mental model of the graphic. Nevertheless, linear approaches have certain advantages that sometimes might justify their use over more intricate alternatives. The main advantage of textual representations is that they are easily displayed through sound output (through speech synthesis) and haptic means (using Braille characters), as we have already discussed. They are therefore the fastest, least expensive, and most straightforward way of displaying non-visual alternatives of graphics, making them a commonplace device on the Web, personal computing devices, museums and exhibitions, etc.

The rest of this section will describe current linear approaches to accessibility of graphics, as

well as common AT and methods for interpreting them. These technologies include Braille, screen readers, and text-to-speech technologies, which make up the basic foundation upon which almost every approach may be accessed.

### 2.1.2 Print Braille

Braille is a tactile writing system named after its creator, Louis Braille. Characters in Braille text are generally made up of six raised dots arranged in three rows of two dots each (see Figure 2.2 for an example), although eight-dot Braille is also in limited use. These raised dots can then be felt by means of touch through sequential gliding of the fingertips over them. Due to their greater reliance on touch, blind persons have a greater tactile acuity than sighted persons [WGG11], which makes learning Braille a relatively easy task for them. Moreover, tactile acuity in blind persons does not decline much with age, in contrast to that of their sighted peers [LMV<sup>+</sup>08], possibly making Braille documents more suitable than speech for older blind persons.

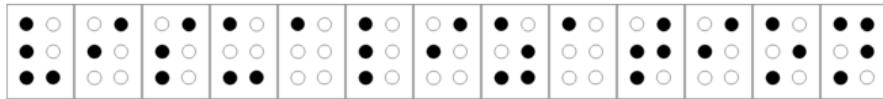


Figure 2.2: The word ‘visualization’, transcribed in Grade 1 English Braille.

Each dot of a Braille character can be set (raised) or cleared (flat), giving a total of 64 possible combinations per Braille character (63 plus the whitespace character). This means that not every character can be directly represented using this system, as just the 26 uppercase letters, the 26 lowercase letters, and the punctuation characters present in the English language plus 10 numerals would already greatly exceed this number. Therefore, different systems of contractions and abbreviations are used and need to be learned by Braille readers. Braille cells are thus context-sensitive, depending on the absence or existence of previous, following or symbol characters in the string being read [Mur08]. For instance, digits are usually represented by lowercase letters (e.g. the ‘a’ character is used to represent the number one) preceded by a special breakout character indicating that what follows next is a number and not a letter. Moreover, separate Braille codes exist for notations systems other than natural language, e.g. music, mathematics, and even highly specialized domains such as chess. The dimensions and spacing of Braille characters are standardized by the majority of Braille producing countries, each character in a text always having the same fixed width. Since the International Congress on Work for the Blind in 1878, the French values of the basic Latin alphabet are used for Braille notations in all languages. However, beyond these twenty-six common letters, national conventions are disparate, and each language might use Braille characters differently to fit their particular structure.

Braille literacy has a positive correlation with employment rates, educational levels, financial self-sufficiency, and reading habits in VIPs [Ryl96]. Reading Braille is, in many ways, similar to reading print. Readers can innately adjust their reading speed to match the comprehension of the text being read, whereas speech rates need to be manually adjusted by the user of the

## 2. STATE OF THE ART

---

given AT producing the output. Moreover, Braille has less ambiguity than speech, as many consonants sound similar and might be hard to tell apart at higher speeds (e.g. the /t/, /s/, and /d/ sounds), and it is more efficient at communicating complex documents [Hal99]. On a personal level, many VIPs might prefer Braille to other reading methods because of the following reasons [Jen16]:

- Greater independence and privacy, with many Braille readers volunteering that Braille is a key aspect to their personal autonomy.
- Braille works to counteract negative stereotypes of blindness among sighted persons.
- Many Braille readers equate the ability to read with unmediated visual or tactile learning, not considering synthesizers and tape recorders as a form of reading.
- Provision of a more intimate relationship with the text and its author. Braille has the ability of giving the reader a sense of “closeness” to the text that cannot be provided by speech output methods.

In spite of its many advantages, Braille literacy rates have been plummeting in the last decades. The National Federation of the Blind in the United States of America estimates that Braille literacy of school age children was at just 12% in 2008, compared to 50% in 1960 [Mur08]. No unanimous consensus on the reasons for this decline exists. However, a number of factors might explain it, including the rise of in the number of visually impaired children with additional disabilities who are non-readers, disputes on the utility of the Braille code, the decline in teachers' knowledge of braille and methods for teaching, negative attitudes toward Braille, and the greater reliance on speech output and print magnification [Ryl96]. In what some call the “paradox of technology” [Nat], while advances in technology have made Braille more available than it has ever been in the past, text-to-speech and audio text technology is perceived to obviate the need for Braille, leading to a widespread “crisis in Braille literacy”.

Braille documents can certainly be less advantageous than speech in many aspects. First, they are cumbersome and costly to produce. Converting typical reading or instructional materials to Braille requires the use of very specific Braille translation software (BTS) and special Braille printers known as Braille embossers [MGH<sup>+</sup>14]. Even though most BTS supports a variety of input formats such as Microsoft Word, Portable Document Format (PDF), or HTML, special requests made by teachers or students might make it necessary to re-implement certain features [MMMR16]. Braille embossers require a special thick kind of paper that supports the printing of raised dots. Moreover, they are generally very expensive, with a cost of between \$1,500 and more than \$80,000 [MGH<sup>+</sup>14]. Because of these reasons, most blind persons cannot produce their own Braille documents, and they depend on local libraries, schools and other associations to provide them with accessible books and documents in this format. Moreover, Braille books are much bigger than their print counterparts, making them difficult to store, and the quality of printed Braille degrades easily with time. Braille deterioration is a common problem in, for example, pharmaceutical packaging, where a single

corrupted dot might completely change the meaning of the package's label e.g. from '£75' to 'large' [Gil06]. Degraded Braille slows reading for meaning significantly more than for letter search, even more so in the case of fast readers [Mil87].

Some of the drawbacks of printed Braille can be overcome by using some forms of Assistive Technology (AT) that provide refreshable Braille output, thereby removing the physical constraints of print documents. These technologies are discussed in Section 2.2.2.

### 2.1.3 Speech Synthesis

Speech synthesis is the artificial generation of human-like speech sounds given an input text. It is the most commonly employed Assistive Technology (AT) by VIPs [AL09], especially in virtual environments (see Section 2.5). A major advantage of the human auditory sense is that it is inherently parallel, allowing changes in auditory stimulus to be noticed regardless of direction and even in the listener's attention is elsewhere [Mur08]. Speech is therefore regarded as a more natural alternative than print Braille to read text. Speech output enables easy access to textual content in an instinctive manner via text-to-speech (TTS) engines, and is almost exclusively the only form of audio employed in everyday non-visual user interfaces. Speech output is adopted not only by VIPs but by sighted users as well, especially in certain situations such as interacting with virtual personal assistants or while driving.

Text-to-speech (TTS) synthesis involves the computation of a speech signal from input text [vSOH97]. Many different publishers offer their own TTS engines as either hardware or software products with different available languages, dialects, accents, and vocabularies. A synthesizer is actually only one part of a full-featured TTS engine; modern TTS solutions are divided into three main parts (which may be in turn split up into a number of sub-modules each), namely [vSOH97]:

1. Conversion of the orthographic input into a string of phonetic and prosodic symbols, including information about phrasing, intonation, and duration.
2. Concatenation i.e. conversion of this string of symbols into a continuum of speech parameters.
3. Acoustic synthesis i.e. the conversion of the parametric representation into speech.

Speech synthesizers are a common addition to screen readers (see Section 2.1.5 and Figure 2.3) allowing blind persons to read electronic documents on their personal computers, as speech output generally outperforms Braille in reading tasks. Compared to the 100 words per minute reading rates commonly achieved by skilled Braille readers, preferred rates for synthetic speech are around 200 words per minute, and experienced readers may prefer an even higher output rate [Mur08]. Among people who use both Braille and speech equally, speech is generally preferred for reading and reviewing short documents, whereas Braille tends to be the chosen alternative when reading more complex documents, either in format or content [Hal99].

However, many capabilities of the auditory system are not exploited when listening to synthetic speech [Mur08]. For instance, small changes in frequency, intensity and rhythm that are naturally perceived by hearing persons and could potentially convey extra information are generally not employed in speech interfaces. Some of these auditory features may be employed to convey, for example, spatial information, which linear approaches such as speech synthesis are inherently not able to transmit. More about current non-speech-based auditory and hybrid approaches can be read in Section 2.3 and Section 2.4, respectively.

### 2.1.4 Multimedia Books

An accessible alternative to print and Braille versions (Section 2.1.2) of books and magazines are electronic documents, which are more readily available than Braille books as they can be accessed with a portable electronic device [LPSZ08]. Beyond traditional audio books, which are simple records of a book being read by a person or through a speech synthesizer, electronic multimedia books or Digital Talking Books (DTBs) employ more advanced interactive features, such as search, bookmarking, speech regulation, video and animation playback, accessible tables and references, etc. Multimedia books can be read in an efficient manner by VIPs using AT such as TTS, magnification devices, and Braille displays.

An open data format for the representation of interactive books that are accessible to VIPs is described by the Digital Accessible Information System (Daisy)<sup>1</sup>. Daisy is a format based on MP3 and Extensible Markup Language (XML) that includes added functionality to that of a standard audio book. In most countries, Daisy is the main or only distribution format of books for VIPs [LPSZ08]. A navigation center is embedded in every Daisy publication that allows the reader to quickly go to any place in the document through a hierarchy of headings by pages or by other significant constructs [Ker03]. The playback systems and production tools can be customized for any language, and all known character sets are supported. A resource file specifies text, audio clips, or images for notifying the reader when encountering certain book features or structures. For example, page numbers can be spoken aloud if the user so desires [LPSZ08].

A great advantage of Daisy books is that they can be read using Daisy players on standard commercially available devices such as personal computers, MP3 players, and smartphones [LPSZ08]. This is especially significant as the smartphone is the fastest growing computing platform used by VIPs, since they provide dynamically-updated information, are multipurpose, and include a good number of built-in universal design parameters (e.g. speech input/output, magnification, high contrast, and vibration) [OGGL15]. Daisy books can also be converted to other formats for playback by popular tablet computers, Web browsers, and even converted into paper Braille [LPSZ08].

Accessibility to graphics in Daisy books is supported by two different means [LPSZ08]. First, an alternative textual description of the image can be provided, quite in the same way accessible graphics are handled on the Web (Section 2.7). Second, structured markup may be used instead of a raster graphic, such as Mathematical Markup Language (MathML) or

---

<sup>1</sup><http://www.daisy.org/>

Scalable Vector Graphics (SVG) (c.f. Section 4.2). In this manner, every individual element in the image can be navigated and heard as spoken words or converted to alternative tactile versions.

### 2.1.5 Screen Readers

Perhaps the most widespread Assistive Technology (AT) for blind users on virtual environments is the screen reader. A screen reader assists people with little or no functional vision in interacting with computers by capturing the (textual) information being displayed on the screen, usually with keyboard assistance, and representing the information as speech or Braille [AL09]. Speech is generated by a TTS engine (Section 2.1.3), and the addition of a refreshable braille display allows it to generate real-time Braille output. Refreshable braille displays, also known as Braille terminals, are electro-mechanical devices for displaying Braille characters, usually by means of raising bumps through holes in a flat surface [AL09]. They are expensive devices usually limited to displaying a single row of 40 to 80 characters at a time. Most users, not being familiar with Braille, prefer speech to Braille output when accessing information through screen readers.

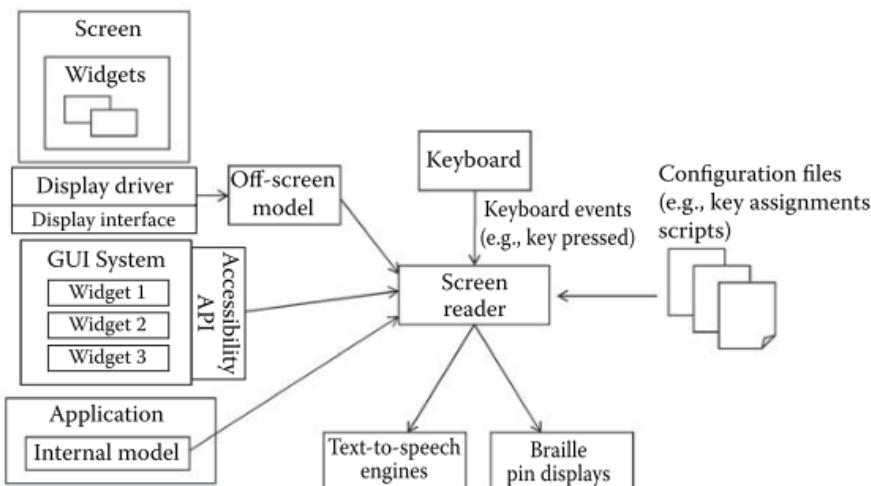


Figure 2.3: Architecture of screen readers and Graphical User Interfaces (GUIs) [AL09].

Capturing the on-screen information and scanning the keyboard input is highly dependent on the running operating system. Therefore, a number of screen readers have been developed to support different operating systems e.g. Microsoft Windows, Linux or Macintosh OS X. Some popular commercially available screen readers are JAWS<sup>2</sup>, Dolphin<sup>3</sup>, Narrator (bundled with every copy of Microsoft Windows since Windows 2000), and VoiceOver (built into most Apple Inc.'s macOS operating systems). In the case of UNIX systems, some open source screen readers such as GNOME's Gnopernicus (no longer under development) or

<sup>2</sup><https://www.freedomscientific.com/products/software/jaws/>

<sup>3</sup><https://yourdolphin.com/products/individuals/screen-reader/>

ORCA<sup>4</sup> are available. Finally, a very popular and open source screen reader for Windows is NVDA<sup>5</sup> (NonVisual Desktop Access). Similar solutions exist for mobile platforms, in which the auditory feedback may be enhanced with vibratory stimuli. Popular examples are Google TalkBack<sup>6</sup> for Android systems and VoiceOver<sup>7</sup> for devices running iOS.

The main benefit of screen readers is that they allow non-visual exploration of software interfaces, including GUIs, which are mainly visual, including not only text but graphical parts like icons, images, menus and control elements. A screen reader analyzes and interprets the content structure of a GUI, such as headings, paragraphs, tables, lists, etc., and its rich text information, such as font styles, sizes and colors, and presents it in a suitable manner to the user. It is important to note that graphical elements will be left out if no alternative description of them is given by the interface designer. Each screen reader provides its users with a set of both predefined and personal navigational keyboard commands that allow them to interact with applications effectively and obtain information quickly and accurately [AL09]. In addition, most popular screen readers offer support to other applications running alongside them in the operating system so their user interfaces can be made accessible as well. When third party applications' GUIs are properly designed with accessibility in mind they become much easier to access for users employing a screen reader (see Figure 2.3). Commonly used applications through a screen reader are Web browsers and multimedia book readers (Section 2.1.4). Section 2.7 discusses accessibility of graphics in the context of the WWW.

Occasionally, some applications offer a self-voicing solution as an alternative specific means for non-visual access. A self-voicing application provides an aural interface without requiring a separate screen reader [AL09]. In this manner, applications can try to offer a non-visual interface suited to their specific functionality. However, this is a not very common approach and it may nevertheless confuse persons who are used to their screen reader's interactive features to access every other application.

### 2.1.6 Automatic Summarization

One of the most recent approaches to textual alternative descriptions of graphics consists in automatically computing a summary of its contents and, more rarely, communicative purpose. Systems that generate textual summaries of non-textual input data are generally referred to as data-to-text (D2T) systems, and have been an active field of research for more than twenty years, albeit usually with goals unrelated to accessibility. They generally aim to enable efficient processing of large volumes of numeric data, supporting traditional visualization modalities, and to reduce the effort that is required by sighted human experts on analyzing the data [DCM12b]. Some examples of D2T approaches in the literature include systems that automatically compute verbal summaries of weather forecast data [GDK94], stock market information [Kuk83], and geo-referenced data [TSR11]. For example, the BabyTalk project

---

<sup>4</sup><http://projects.gnome.org/orca/>

<sup>5</sup><https://www.nvaccess.org/>

<sup>6</sup><https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback>

<sup>7</sup><https://www.apple.com/accessibility/iphone/vision/>

[PRG<sup>+</sup>09] generates multi-paragraph verbal summaries of large quantities of heterogeneous clinical data collected for babies in a neonatal intensive care unit. These summaries, which first introduce the detected potentially most relevant key event followed by related events, are then used to support medical decisions, having the potential of improving human decision-making.

However, the input of the big majority of D2T systems is heavily domain-dependent and confined to numerical data and, before them, textual data used by document summarization techniques existing since the late 50s (see e.g. [Luh58]). Systems which summarize diagrammatic representations directly without accessing to its underlying raw numerical data are much harder to come across in the literature. Early work by Kurze [Kur95], now unreachable, has been reported [ESC<sup>+</sup>07] as generating text conveying the basic content of business graphics, including verbal descriptions of the diagram's number of data sets, axes' labels and ranges, etc., information which is then spoken aloud to the user by a speech synthesizer.

The most noteworthy work in this category though is probably the SIGHT (Summarizing Information GraphHics Textually) system developed by Demir et al. [DOS<sup>+</sup>10]. SIGHT provides a brief initial summary of bar charts along with the chart's most significant features. It is thought for situations in which the reader of a graph is interested in the high-level content of the graphic and not in its details. According to the authors, conveying all informational content of the graphic would overwhelm readers, and moreover many uninteresting details would likely be provided. In case the user wishes to obtain further information about the graphic, the system then suggests a number of more focused follow-up requests, upon which history-aware answers are provided. However, users are forced to follow the system's guidance and suggestions in choosing follow-up questions, and are not allowed to ask free-formed questions. The system is made up of several cooperative modules, the two most notable ones being the following:

- *Intention Recognition Module*: it is responsible for recognizing the graphic's intended message by reasoning about the communicative signals present in the graphic. Three kinds of communicative signals present in bar charts are considered: the relative effort required for various perceptual and cognitive tasks, whether certain elements of the chart are highlighted, and the presence of certain verbs and adjectives in the graphic's caption.
- *Generation Module*: its task is to select the appropriate content for the current response according to the graphic's representation, its communicative message as computed by the Intention Recognition Module, and the kind of user request. The selected content is then organized and realized into a response by using a bottom-up generation approach. A user model is collected in order to generate tailored answers to the user's interests.

SIGHT demonstrates a very powerful approach for automatically summarizing bar charts that may be employed to give VIPs a detailed overview of simple charts when they are not interested in the details. However, such summaries might become too large and difficult to mentally process for more complex charts. Moreover, in spite of its potential for being

## 2. STATE OF THE ART

---

interactive, SIGHT does not provide in-depth navigational capabilities to its users, leaving the interpretation of the graphic to its author and the implemented summarization algorithms, falling short in giving blind persons the autonomy they require to obtain an individual judgement of the information [FLST13].

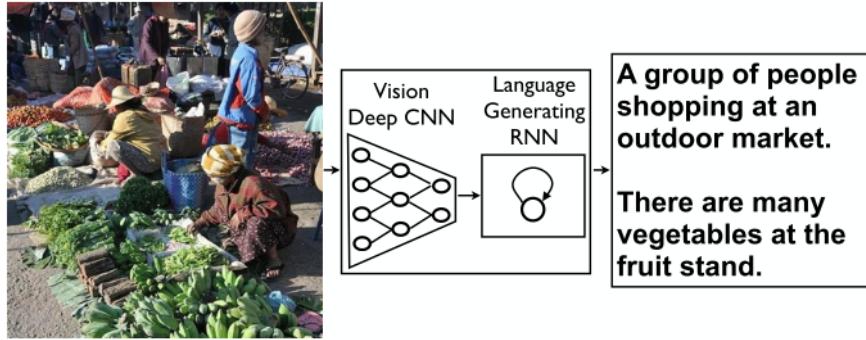


Figure 2.4: Automatic captioning of a picture employing a neural network approach from [VTBE15]. Verbal descriptions are generally a sufficient means for non-visually describing simple real-world scenes.

In contrast to the lack of research regarding automatic captioning of diagrams, generating natural language descriptions from real-world pictures (a problem known as *automatic image captioning*, see e.g. Figure 2.4) has been receiving much research attention in recent years, achieving very promising results e.g. [KPO<sup>+</sup>13, VTBE15, FL10]. These approaches combine techniques of computer vision and natural language processing and generation to create human-readable descriptions with applications in image-retrieval algorithms and accessibility. This is a problem that stems from image classification and object recognition tasks (described in greater detail in Section 3.3), which are well-studied fields by the computer vision community. However, generating natural language descriptions from images introduces additional challenges, as a description must not only describe objects but also express how these objects relate to each other as well as their attributes and the activities they are involved in. Moreover, the semantic knowledge needs to be expressed in a natural language, making it necessary to include a language model in addition to visual understanding [VTBE15].

Generally, automatic object detection methods are first used to identify and/or describe the visual properties of objects present in real-world images, generating a list of keywords containing the main elements of the image and their attributes. Recently, as image understanding problems become more dependent on contextual knowledge, more nebulous definitions of what constitutes an object are being used. For example, the object class of ‘toys’ is significantly abstract and depends heavily on context [AT13]. After the main components of the image have been identified, smooth sentences containing the keywords and their (usually spatial) relationships are generated using a broad range of text statistics and language models, making up the verbal description of the image.

Natural language generation constitutes one of the fundamental research problems in natural

language processing (NLP). It is core to many of its applications including machine translation, summarization, dialogue systems, and machine-assisted revision, and remains an open research problem. Traditional approaches to generation of natural language captions of pictures have consisted on matching whole images to a database of images with captions e.g. [OKB11], or automatically retrieving and summarizing relevant documents (for example, from a corpora of Web sites containing images) e.g. [FL10]. More recent approaches have been however based on automatic generation of text from scratch, by detecting multiple objects, modifiers, and their spatial relationships, and generating novel sentences to fit these constituent parts [KPO<sup>+</sup>13]. Recurrent Neural Networks (RNNs) have been also explored to output meaningful natural language sentences, deriving inspiration from their use in machine translation problems [VTBE15].

In any case, current automatic scene understanding approaches produce static, non-interactive alternative versions of graphics which moreover might not always accurately describe the contents of the image, specially in the case of diagrams. None of these image summarization methods offer an interface to their readers, which is generally required to facilitate exploration of the information space, even more so when navigating diagrams non-visually [FLST13]. Communicative images and Visual Question Answering (VQA) systems, described respectively in Section 2.6 and Section 2.5.3, present an interesting cognate alternative to automatic image captioning that let users obtain more fine-grained information according to their specific goals and interests, while in addition letting them modify or add missing or incorrect information from the image.

### 2.1.7 Image Metadata for Accessibility

Some of the metadata that is transparently stored in many image formats has the potential of being used to generate or improve their accessible counterparts [Hak12]. Embedding the alternative textual description of an image within the image itself has the benefit of simplifying the authoring workflow of accessible graphics, especially considering how most expert publishing-tool users are not typically the subject matter experts who would be most capable of authoring useful descriptions, and how cumbersome the addition of alternative text in most publishing software is [DIA11]. Authoring tools may embed the equivalent of HTML alt text descriptions (Section 2.7) directly inside an image file, stored as description fields on its metadata. When the image is then imported into the publishing tools or extracted from some other source, the embedded description could be extracted by the publishing mechanism and re-wrapped in the target data format, such as Daisy (Section 2.1.4) [DIA11].

This may be particularly useful in the case of real-world photographs, as even the most basic digital cameras typically record and store some metadata inside each image, such as the date and time at which the picture was taken, camera model and manufacturer, several camera settings, and even geographical information (GPS coordinates) in the case of many higher-end cameras [DIA11]. Furthermore, additional information, such as the approximate location where a picture was taken, may be inferred from the exposure information and other camera settings stored within the image's metadata [San10]. Temporal and spatial features

## 2. STATE OF THE ART

---

contained in raster graphics' metadata have also been exploited to automatically classify images into categories according to the user's preferences [JLLC07].

Metadata Format	Supported by
EXIF	TIFF, JPEG
IPTC	TIFF, JPEG
XMP	TIFF, JPEG, JPEG 2000, PNG, GIF, PDF
SVG	SVG

Table 2.2: Support of metadata formats by image formats [DIA11].

The standard metadata format for images taken by digital still cameras is the Exchangeable Image File Format (EXIF), maintained by the Japan Electronics and Information Technology Industries Association (JEITA). It is supported by the .tiff and .jpeg image formats and is currently in use by most digital cameras. Metadata tags defined in EXIF can be divided into four different categories: date and hour information, technical characteristics of the camera configuration, information about location, and copyright description and information. The latter section has been traditionally used by digital librarians to add image descriptions for cataloguing purposes [LGS<sup>+</sup>08]. In addition to EXIF, several different image metadata formats exist, including pure metadata specifications e.g. XMP, and function specifications such as MWG. Table 2.2 shows a few of these metadata formats and which image formats support them.

Even though most metadata formats share some common aspects, such as the 'description' and related fields which may be used for accessibility related purposes, each specification is currently independent of the others, and therefore different metadata formats cannot be interchangeably adopted [DIA11]. Moreover, the metadata fields can only be used to describe the image as a whole, whereas its constituent elements and relationships cannot be individually outlined, which is a necessary feature to properly describe diagrammatic representations in a non-visual manner, as discussed in the previous section. Finally, another drawback to using these fields as a non-visual alternative to the graphic is that no currently available assistive technology can locate and read descriptive metadata embedded within raster images [DIA16].

Vector image formats, particularly SVG, have the benefit of providing some built-in support for description metadata in either verbal or graphical form, as the image content itself is text based. Specifically, at the basic level the SVG format defines a `title` element and a `desc` element that do not visually render by default but are intended for human consumption. `title` provides a human readable heading for the whole document or any element within that is automatically recognized and voiced by most modern screen readers. Each individual graphic element of the image may be added a `desc` element containing a more in-depth textual description of it (see e.g. Listing 2.1). This potentially allows vector graphics to be sequentially navigated through their constituent elements. For example, each slice of an SVG

pie chart can be augmented with a `desc` element describing its labels, color, share of the total aggregate, etc. These elements may then be grouped into a container element that is correspondingly described in order to outline the chart as a whole. However, general-use screen readers are still not able to read `desc` tags of SVG elements [DIA11]. Moreover, this approach would only allow a sequential exploration of the elements in the graphic, yielding similar shortcomings to those mentioned earlier in this chapter. The accessibility of simple SVG images may also be enhanced by using a supported subset of attributes from the WAI-ARIA technical specification, namely `role` and `aria-labelledby` [Wat13]. These attributes however only provide a small improvement from the canonical accessibility features of SVG.

```
<g>
  <rect
    style="fill: #000000; fill-opacity:1; stroke:none"
    id="rect3764"
    width="4.621418"
    height="117.33674"
    x="125.25249"
    y="686.80524"
    <desc>Vertical Metric Axis describing population</desc>
  >
</g>
```

Listing 2.1: An SVG `desc` metadata tag used to describe a graphic element as the vertical axis of a statistical chart

Besides these built-in elements, SVG files might include all kinds of non-graphical information embedded within the image itself but which are not visually displayed to sighted readers. Enhancing static graphics found on the Web with multimodal descriptive metadata about the image in SVG and other dynamic graphic formats is the main idea behind the concept of *Smart Images* [Hak12]. They have been briefly considered as an upgrade to traditional JPEG and GIF image formats on the WWW carrying the potential for descriptive information about the image to be closely coupled and even contained within the image format itself. Smart Images put forward an approach where images directly contain sufficient information to allow access in a user's preferred modality through a standard interface, including text, sonification, and haptic elements that may be explored through available assistive technologies. The embedded semantic enhancement of a graphic may include a full-fledged axiomatic knowledge base underpinning the elements on the graphic, their properties and relationships. This is the core concept behind communicative images, which are extensively described in Section 2.6.

## 2.2 Tactile and Haptic Approaches

Haptic alternatives to graphics have been in use for over 200 years [Eri98]. They take advantage of the haptic subsystem of human beings to represent spatial information in an

analogous fashion to printed Braille (section 2.1.2) does for textual content, and, more recently, through the use of kinesthetic (e.g. force-feedback) and vibratory devices. Cutaneous sensors on the skin detect touch and temperature, while the kinesthetic sensors on the muscles and joints of the body sense motion [GM11]. In any case, the human haptic subsystem requires physical contact with objects to acquire information. As a result, the haptic perceptual field is considerably reduced compared with the visual field and large exploratory hand movements are necessary in order to perceive objects in their entirety, making haptic perception much more sequential than visual perception [Hat93]. Tactile devices have applications apart from accessibility as well, for example, as controllers in tele-operated manipulation systems [HKP95].

### 2.2.1 Hard-copy Tactile Graphics

Traditionally, haptic graphics have been presented in Braille and raised dots and lines on swell-paper, known as *tactile hard-copies*, which are commonly authored by sighted experts following a set of guidelines. Several different guidelines for the design of hard-copy tactile graphics exist. Their purpose is to provide transcribers, educators, and producers with information about best practices, current methods, and design principles for the production of readable tactile graphics [The12]. Among them, the guidelines and standards for tactile graphics by the Braille Authority of North America [The12], and the handbook for the creation of tactile graphics by the Karlsruhe Institute of Technology [Uni11] are specially worth mentioning.

Tactile perception does not vary much from person to person; moreover, blind persons explore tangible media generally following the same strategy, divided in two main stages [WB97a]. The first stage consists of a broad exploration of the image as whole, which provides the reader with a general overview of the graphic. This stage can be further divided into a detection stage and a discrimination stage. These stages are achieved by identifying the presence of an object and its independence from other objects. Then, after the general overview has been obtained and understood, the details of the graphic are individually explored in two stages: identification and comprehension [Mur08]. Three main exploratory procedures are used during free exploration of the graphic [OGGL15]: lateral motion (moving the fingers back and forth across a texture or feature), contour-following (tracing an edge within the graphic), and whole-hand exploration of a global shape. However, even with the use of these techniques and experience in reading tactile graphics, a person may easily forget previously explored elements if many of them are present. Therefore, it is very important to avoid information overload when designing tactile graphics. Transcribers of tactile graphics often expand the image scale, remove small features, and divide more complicated diagrams into multiple separated tactile graphics [OGGL15]. A one-to-one correspondence between the original graphic and its tactile counterpart needs to be avoided except for very simple diagrams, as doing so leads to perceptually cluttered and unusable displays [OGGL15].

The main benefit of the employment of the haptic subsystem as a substitute of the visual subsystem is that both can provide much the same information, including characteristics of physical objects such as shape, size, texture and position. Tactile alternatives are especially

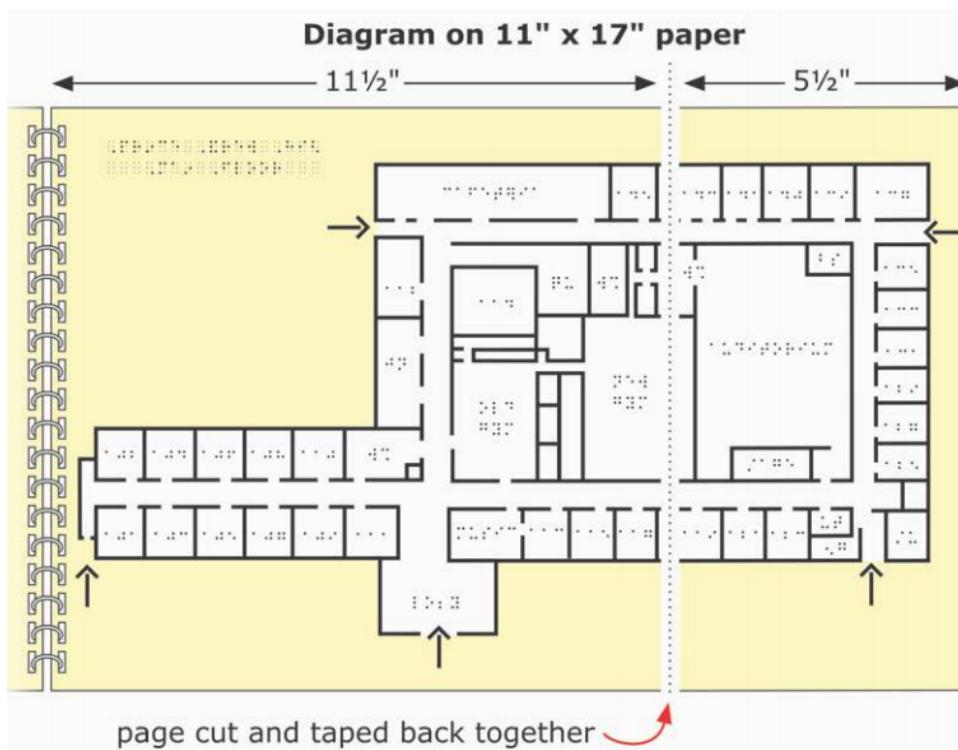


Figure 2.5: A hard-copy tactile high school floor plan [The12].

practical when the topology and geometric relationships of the diagram need to be preserved for its full understanding, including the relative size and position of its graphical objects [GMA10]. Tactile approaches provide direct perceptual access to spatial information, which is much harder to generate and interpret using spatial language or audio [OGGL15]. Therefore, tactile alternatives are traditionally employed most often when the spatial characteristics of the diagram are particularly relevant, the most typical case perhaps being tactile hard-copies of geographical maps and floor plans (see e.g. Figure 2.5). Tactile maps have been in widespread use since the 1960s, when they were first introduced to teach mobility to blind children [GWJ65]. Nowadays, they are the most common way of teaching geographical features to blind pupils. Many production methods for tactile maps have been developed, the two dominant ones being thermoform-based maps and microcapsule (i.e. swell-paper) maps [ZMW15]. Besides maps, other kinds of diagrammatic representations have also been traditionally presented to blind students in Braille hard-copies [YRB01] or by verbal descriptions in the simplest cases. Deciding the appropriate variant of a graphic to use in learning materials is a highly individual task that almost always requires human intervention. Methods and tools that help with this decision have been developed, such as the image sorting decision tree developed by the DIAGRAM center initiative, shown in Figure 2.6.

In spite of their popularity, this kind of presentation technique presents many drawbacks

<sup>8</sup>Image source: <http://diagramcenter.org/decision-tree.html>

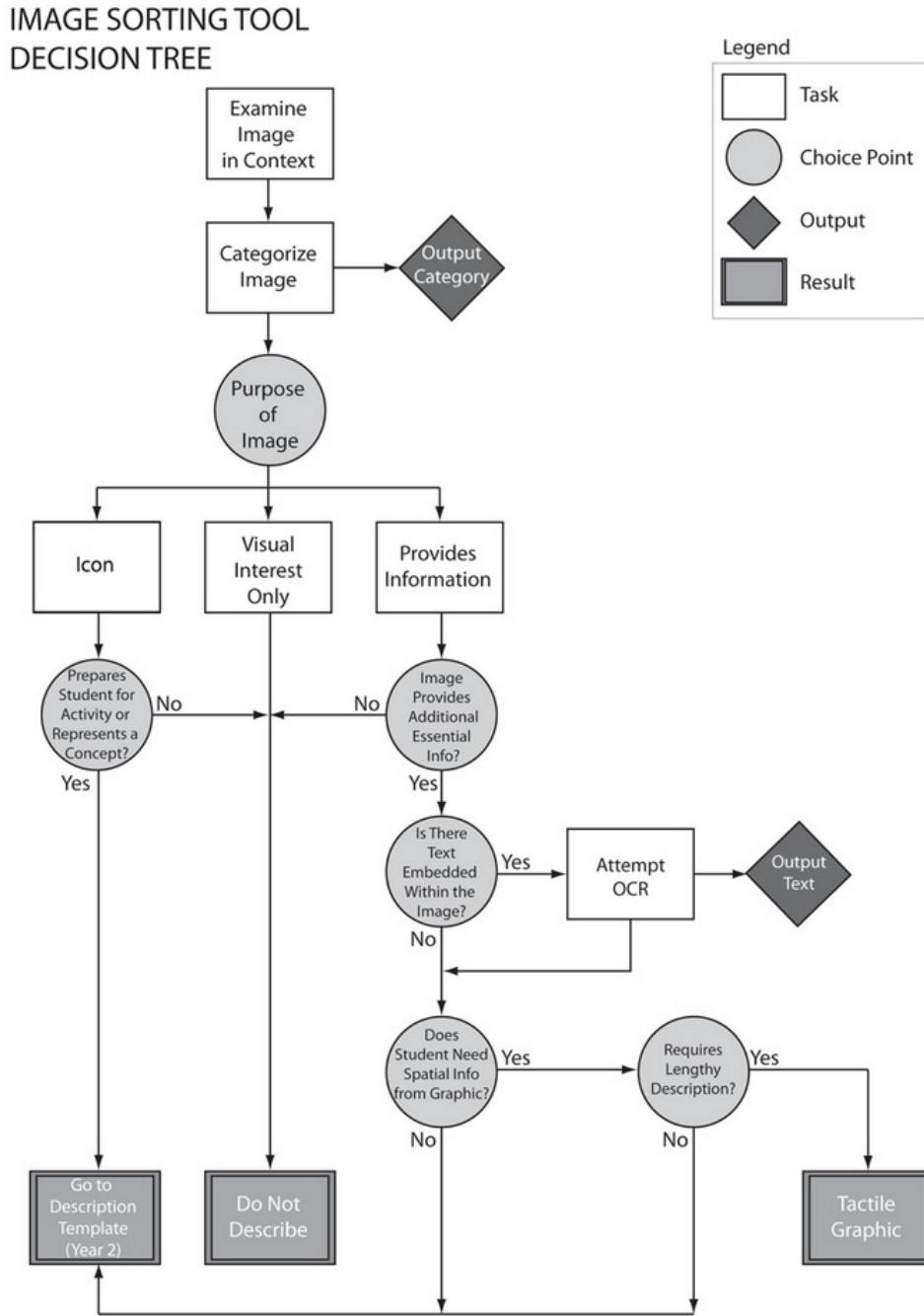


Figure 2.6: The decision tree developed by the DIAGRAM center initiative is a tool for choosing which print images need tactile alternatives and for which ones a textual description suffices<sup>8</sup>

that restrict their take-up by the blind population. In the first place, the sequential nature of haptic perception introduces a significant memory load when storing and integrating the spatial attributes of tactile graphics, whereas the visual system can infer these relationships almost instantaneously [GMA10]. Furthermore, as discussed in Section 2.1.2, only a small percentage of blind persons can read Braille text, and therefore every tag, legend, and any other textual content of the graphic has to be delivered to the user through non-haptic means. Not only textual content may be unclear: blind readers can only get a rough idea of the graphic's content itself due to the lower resolution and accuracy of raised graphs [YRB01] and the need to simplifying the content, as previously stated. As a result, complex details on the graph are difficult to perceive and tend to confuse the reader. Dynamic data cannot be represented using traditional tactile print documents either, which furthermore need to be drafted by a trained tactile graphics transcriber and are expensive to produce [Gar14]. Hard-copy tactile graphics are also prone to degradation with time, which slows their comprehension, more so by fast braille readers [Mil87]. Tactile documents are cumbersome to store and transport too, as they are much bigger in size than their print counterparts, and expensive to produce. Cost and availability issues are particularly significant in the developing world [Sha14]. Consequently, most blind persons do not even attempt to read a tactile graphic. Even when taking into consideration only those who can understand Braille, it has been estimated that no more than 10% to 25% of them can read a braille diagram. This has led to a widespread conviction that word descriptions provide the best access to graphical information [Gar14]. Many students report to preferring verbal descriptions and auditory representations of diagrams over tactile alternatives [Sha14].

### 2.2.2 Tactile Displays and Haptic Devices

An increasing number of alternative approaches to hard-copy tactile graphics that can produce diverse tactile effects have been developed over the past 10 years. Currently, there are six main classes of available tactile and haptic devices [OGGL15]: *surface haptic displays*, *lateral skin displacement displays*, *vibrotactile displays*, *tactile shape displays*, *bubble displays*, and *force displays*. Moreover, tactile means may be used to enhance traditional and novel user interfaces. For instance, so-called *hapticons* were integrated in the Kevin system [BE98], designed for blind software engineers to construct diagrams. Hapticons are the tactile equivalent of icons in a graphical user interface, and are displayed on a tactile overlay connected to the host computer. They make interface buttons non-visually distinguishable from each other, and act as an *aide-memoire* when the user is learning the system. Integration of haptic and non-haptic means will be discussed in Section 2.4. Let us now briefly look over the current approaches to computerized tactile graphics, laying out their main characteristics, working examples, benefits, and drawbacks.

#### Surface Haptic Displays

*Surface haptic displays* rely on modulating the friction that is created when a fingertip is moved across a flat surface such as a glass or plastic screen. Friction may be modulated through different techniques, for example electrovibration [XIP<sup>+</sup>11] or electrostatic force

modulation [WGCP07]. Modulating friction allows these devices to create the illusion of geometric features on flat surfaces. However, these approaches still present important drawbacks [OGGL15]. First, one finger must be in contact with the display at all times. Second, no clear perception of corners or image boundary edges can be obtained. Finally, the image beneath the fingertip disappears as soon as the fingertip stops moving. This signifies that only simple shapes can be explored, and complex diagrams cannot be represented using these devices.

### Lateral Skin Displacement Displays

*Lateral skin displacement displays*, e.g. [LPHL05], apply differential stretch to the surface of the skin within the contact path [OGGL15]. As in surface haptic displays, there must be relative motion between the fingertip and the surface in order to discern the graphic elements. But, in contrast, small features such as small dots can be rendered beneath the fingertip. Their main drawback is the reduced size of the display surface, no larger than the fingertip. Therefore, navigating tactile graphics by these means is analogous to a small lens which is moved across a virtual image, and complex diagrams can also not be represented with these devices.

### Vibrotactile Displays

In *vibrotactile displays* a flat surface, for example the screen of a tablet or smartphone, is caused to move driven by some form of vibrating source [OGGL15], such as an embedded motor or a piezoelectric transducer. They may be used to render tactile graphics by creating the illusion that the screen is vibrating only at the point where the user's finger comes into contact with a region of the screen corresponding to a graphic object, such as a line in a chart. A classic example of vibrotactile display is the *Optacon* device developed by Linvill and Bliss [LB66], which consists of an optical scanner and an array of pins that vibrate according to light and dark areas under the scanner. Even though reading speeds using the Optacon are very slow, it has proved to be an invaluable device for activities such as reading labels on supermarket items [Mur08]. Some of these approaches have shown to provide a comparable level of access to graphical material as is possible from their traditional hard-copy counterparts [Pal13]. Accurate extraction of graphic information such as the orientation and thickness of the lines, regions, etc. is anyhow not as directly perceivable as in hard-copies and pin-array displays, demanding a higher cognitive load from the reader than in traditional tactile approaches [KGBL14].

Retaining orientation on curved or jagged lines using these displays is a challenging task, making this technology not appropriate in situations where precise interpretation of high-resolution, irregular lines is required, for example when dealing with maps [OGGL15]. Moreover, when the represented diagram is bigger than the available screen size, complex diagrams become too hard to navigate even when panning and zooming mechanisms are incorporated. A number of additional cues may be added that aim to solve this issue, such as audio cues for specific graphic objects, different vibration patterns, etc. [Pal13]. However, since modern commercial tablets and smartphones generally possess a single vibration actuator, the signal strength of

the single motor degrades across the display extent, making the vibrotactile stimulus hard to perceive [OGGL15]. Devices with several independently controlled vibration actuators, such as the Optacon [LB66], allow for more detailed images to be perceived. However, display size limitations still restrict the complexity of the graphic. Moreover, specific devices like the Optacon would be expensive, not portable and hard to learn, especially compared to everyday devices such as tablets and smartphones. Even though they are not a sufficient means for autonomous non-visual navigation of complex diagrams, vibrotactile displays have proved to be successful in supporting other modalities in order to reinforce them e.g. by improving edge detection [OGGL15] and remain an interesting device in accessibility research.

### Bubble Displays

One of the most recent approaches to presentation of tactile information are *bubble displays* e.g. [BPM09], in which small cavities beneath a flexible surface can be selectively inflated to create tangible surface features such as bumps and edges [OGGL15]. The main motivation beneath this new technology is the creation of so-called 'soft' buttons for flat screens on tablets and phones. Therefore, they are more likely to become commonplace in everyday devices, which make them a hopeful new approach to widespread accessibility of graphics. Current bubble displays are limited to showing static graphics, as the shapes of the objects they can display are fixed by the position of the cavities below the surface. As these kinds of display are still in their infancy, they do not possess the expected capabilities for displaying complex diagrammatic representations.

### Tactile Shape Arrays

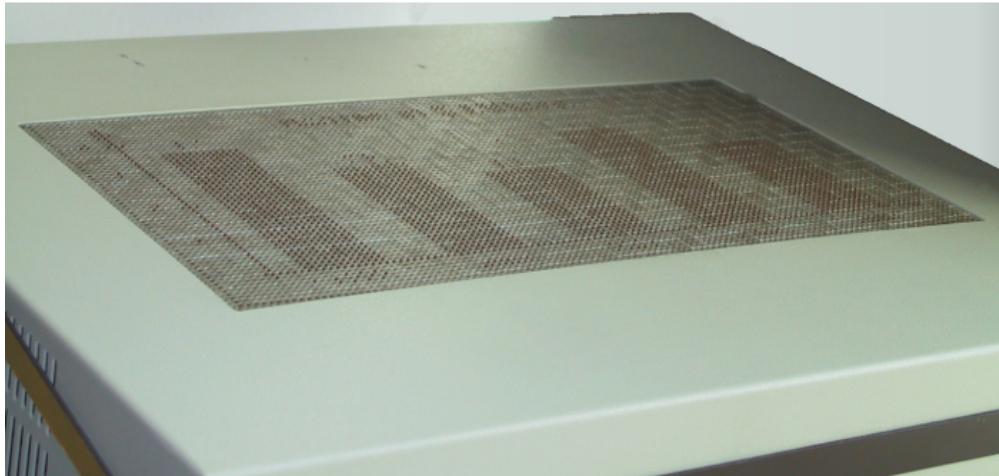


Figure 2.7: A tactile matrix display (the *HyperBraille* from Metec), displaying a bar chart [ROE04].

The most widely used and researched tactile alternative to graphics besides hard-copies are *tactile shape arrays*, also known as *pin array devices* [OGGL15]. These devices consist of a

large rectangular set of pins that can be raised and lowered like those in a refreshable Braille display (see e.g. Figure 2.7) or use vibrating pins or plastic bumps [Mur08], rendering an equivalent of a hard-copy braille document (discussed at the beginning of this section). In contrast to other computerized approaches, the dots in a pin array device apply both normal and tangential force to the fingertip as it slides across its surface, preserving both the force and geometry cues of real surface features [OGGL15]. They are therefore the kind of device that allows the most similar experience to reading traditional braille embossed graphics, and are by far the most suitable haptic technology for rendering graphic images [OGGL15].

Their main benefit is the support for multipoint interactions, which allows maximum use of the small bandwidth capabilities of the sense of touch [Mur08]. Reading performance is greatly enhanced by the possibility of exploring the graphic by using both hands and multiple fingers simultaneously [OGGL15]. Another important feature of these devices is that movable dynamic information can be displayed (see e.g. the concept of “Braille movies” laid out by Fricke and Baehring [FB94]). Some of these devices even allow a third dimension to be represented. For example, the tactile table developed by Shinohara et al. [SSM98] consists in a  $64 \times 64$  array of pins, each of which can be raised in 0.1mm steps to a maximum height of 10mm. The height difference between areas was used successfully to represent shade in a monochrome image. A similar device, developed by Wagner et al. [WLH02], carries a  $6 \times 6$  array of pins that can be raised to between 0 and 2mm, rendering a resolution of around 3mm.

Despite their potential, commercialization of tactile shape arrays has been severely hindered by the large cost associated with their manufacturing [Mur08]. The price for a pin array display starts at around \$2,000 for a single line 16 character unit, and goes up to over \$56,000 for a touch tablet composed of 7,200 dots [OGGL15]. These kinds of device are moreover generally bulky and difficult to store and carry. Furthermore, the size of most of pin array devices is not sufficient to display complex diagrams [BE98], and their dot resolution is lower than that of hard-copy tactile graphics.

### Force-feedback Devices

An alternative approach to cutaneous haptic displays are *force-feedback devices*, which make use of the kinesthetic sensors of the body to render virtual three-dimensional shapes through force feedback. Usually, users need to insert one of their fingers into a thimble or hold a stylus attached to one end of the mechanical linkage, coupled with electrical motors at the other end, to feel the reaction force. Contact and interaction with virtual objects can be represented by the appropriate force generated from the motors by tracking the position of the thimble or the tip of the stylus [YRB01]. Most current force displays can provide movements in either two- or three-dimensional space, the most popular device of its kind being the *PhanToM*<sup>9</sup>, developed by Geomagic. Another example of force display is the *Immersion Impulse Engine*, which has been used to perceive virtual textures and explore 3D objects [CPK<sup>+</sup>98].

---

<sup>9</sup><http://www.geomagic.com/en/products/phantom-premium/overview/>

Force displays allow adequate manipulation of virtual objects as well as shape and weight inspection, but they only provide a single point of contact with the surface, which (as in the case of surface haptic displays) is a very inefficient means of navigating complex haptic graphics [OGGL15]. Due to the huge bandwidth limitation the single contact points imposes, which is further exacerbated by the mediation of the device's end-effector (e.g. the PhanToM's stylus), they are not a suitable means for exploring most diagrams [YRB01]. Detailed and realistic texture of graphic objects is very difficult to reproduce using these devices. For instance, friction-based feedback may be employed to distinguish different lines on a simple line chart, but become less effective as soon as there is more than one line in the chart [YRB01]. Keeping the pointer on a line of the chart is a difficult task, especially at the corners and the end points of the lines. Some auxiliary devices such as haptic grids may be employed to help with these tasks; however, not every user finds them useful [YRB01].

### Augmenting Haptics

All the aforementioned tactile alternatives to graphically displayed information usually need some kind of extraneous explanation before a user can make sense of it, even if they provide Braille legends or other explanatory symbols [LW03]. When no assistance is available, autonomously reading a tactile diagram becomes a very time-consuming and tiring process [YRB01], which discourages blind persons from carrying it out. External assistance to the blind reader has traditionally been given by a sighted person; however, a personal assistant might not always be available and furthermore it hinders the ultimate goal of self-sufficiency for blind persons accessing print materials. To make up for this crucial limitation, many hybrid approaches combining tactile graphics with sonification, audio or speech cues have been explored. Hybrid concepts are discussed in Section 2.4.

## 2.3 Sonification Approaches

*Sonification* is defined as the representation of data into the sound domain using non-speech audio [KWB<sup>+</sup>], allowing the user to make judgements regarding nominal/symbolic (e.g. a category), qualitative (e.g. comparing relative sizes of data values), or quantitative (e.g. telling the exact value of a data point) aspects of the information being heard [NR07]. Interfaces based on data sonification are known as *auditory displays*. Classic examples of this type of interface are manual telegraphy (Morse code) or the audio output routinely used in hospitals to keep track of physiological variables such as those measured by electrocardiogram (ECG) machines [PBB<sup>+</sup>08]. Even though traditional sonification methods such as these are purely linear, contemporary techniques allow more complex information to be communicated, including two-dimensional graphics, some of which will be examined later in this section. Nowadays, sonification approaches are routinely employed to meet a number of needs including [PBB<sup>+</sup>08]:

- presenting information to Visually Impaired Persons (VIPs)

## 2. STATE OF THE ART

---

- providing an additional information channel for people whose eyes are busy attending to a different task
- alerting users to errors or emergency states of a system
- providing information via devices with small screens such as PDAs or cell phones that have a limited ability to display visual information

There is currently a substantial and growing body of work regarding sonification, stimulated by the increasing availability of powerful technical tools for creating auditory displays. Nevertheless, sonification is still a field somewhat in its adolescence, which is considered new by most engineers and designers [PBB<sup>+</sup>08].

### 2.3.1 Auditory Icons, Earcons, and Spearcons

In regard to accessibility, auditory interfaces that aim to (partially) replace GUIs are well studied in the HCI literature. An important design challenge that this kind of interface presents is that the auditory cues must convey the contents of the interface in addition to their more traditional role of conveying feedback about user actions and application events [Myn95]. To support quick recognition and learnability of the auditory cues, auditory icons may be used to represent GUI objects. *Auditory icons* map attributes of everyday sound-producing events to attributes of the model world of the computer [Gav89]. For example, selecting a file might give user feedback by playing the sound of an object being tapped. Auditory icons are similar to visual icons in that both rely on an analogy between the everyday world and the model world of the computer. The power of these organizing metaphors makes it possible for auditory icons to be learned by users as with a similar amount of effort as visual icons [Gav89]. The use of auditory icons has been suggested to represent otherwise hidden information to sighted users as well, such as information out of sight due to screen size limitations [BWE].

Similar devices to auditory icons are *earcons*, which use a more traditional musical approach that does not offer an intuitive link between the played sound and what it represents [Bre02]. Earcons are built up by a series of short, rhythmic sequences of pitches known as *motives*. More complex earcons may be constructed by means of concatenating individual earcons or by structuring them in a hierarchical manner in order to represent more intricate structures [Bre02]. For example, musical earcons have been explored to display the spatial layout of simple objects in a graphical area in the AUDIOGRAPH tool [AR98]. Music feedback alone is employed to communicate the current position of the cursors on the graphical display, the nature of the graphical objects therein (i.e. their type, size, and shape), and their overall position using various scanning techniques. In their approach, coordinate points were described by using a musical mapping from distance to pitch, a higher note describing a larger coordinate value. Horizontal and vertical coordinates were distinguished by timbre (organ and piano, respectively). This same metaphor was employed to musically describe graphical shapes. Finally, the authors also carried out some experiments to test whether users could estimate the size of graphical objects. Control actions were implemented using meaningful

earcons, such as playing a tune with a “fault” in it followed by the correct tune to represent the undo command.

Results from these experiments were mostly positive, with most users being able to identify shapes and their approximate size, use the tool to move around the graphical area, and adjust their size. However, users gave negative feedback regarding the long size of the musical messages and the effort involved in interpreting them. Moreover, when no explicit semantic guidance was given the results were often lacking a meaningful interpretation of the graphical objects, which implies that the perceptual context has a direct and contributing role in the interpretation of the music used to communicate graphical information. It is thus noted that in a practical interface it would be sensible to use speech in addition to music to allow for exact information to be conveyed [AR98]. The advantage of auditory icons over earcons is that they are easy to learn and remember because they are based on sounds that contain a semantic link to the objects they represent. However, their association to more abstract actions or objects becomes more difficult, since when the meanings of auditory icons must be learned they lose some of their advantages [Bre02]. Moreover, auditory icons are not as effective in displaying hierarchical information [WNL06]. On the other hand, earcons require more learning and potentially more explicit training due to their arbitrary mapping [WNL06].

A third, more recent class of auditory display to communicate symbolic information that aims to tackle some of these issues is the so-called *spearcon*. Spearcons are created by speeding up a spoken phrase until it is not recognized as speech, with the resulting sound being used in a likely manner to an auditory icon [WNL06]. They are created automatically by converting the text of a menu item to sped-up speech, without changing pitch, via text-to-speech synthesis. This way, spearcons are unique to a specific item on the interface, while related items tend to have similar sounding spearcons (e.g. the spearcons for ‘save’, ‘save as’, and ‘save as Web page’ are all unique while being acoustically similar at the beginning of the sound, which allows them to be mentally grouped together). The non-arbitrary mapping results in a shorter required learning phase for the user than in the case of earcons [WNL06].

In any case, auditory icons, earcons, and spearcons are mapped to a single object or action. Therefore, they are a suitable means of representing nominal and, in the case of hierarchical earcons, qualitative information, but are incapable of describing quantitative information, which greatly hinders their potential use in non-visual displays of statistical charts and other knowledge representation graphics. It is however important to note that these techniques may be employed in combination with other interpretation levels to create more powerful auditory displays (for example, in some daily weather sonification approaches, water sounds have been employed to give an iconic link to rain, while the duration of the sound indicates the amount of rain per unit of time) [PBB<sup>+</sup>08].

### 2.3.2 Diagram Sonification

Beyond their use in user interfaces, sonification is said by some authors to be the best evaluated technology for non-visually communicating information about statistical charts [FLST13]. Most of these technologies employ sonification as a means of supporting some

## 2. STATE OF THE ART

---

other interface, such as traditional GUIs or haptic graphics, and do not rely purely on non-speech sounds to represent graphical information. In those instances where they are purely aural, these displays are known as *audification interfaces*. In audification, the data “speaks for itself” by e.g. using every data value as a sound sample in a sound signal  $s(t)$  [PBB<sup>+</sup>08].

An important benefit of sonification approaches in comparison with tactile methods, discussed in section 2.2, is that responses to auditory stimuli are often faster than those of any other sense. This is furthermore aided by the omnidirectional nature of sound, in that a person will notice a sound even if their attention is elsewhere [Mur08]. Specially, changes in rhythm and frequency are very noticeable. Moreover, the auditory system is very sensitive to changes in status. For instance, even if a listener can get used to background noise and ignore it, it becomes immediately noticeable the moment said noise stops [Mur08]. This makes sonification a very interesting approach for representing quantitative information that changes often in time (or along some other variable), such as the values of the data points in a line chart, as discussed later in this section. A number of mathematical concepts such as asymmetry, monotonicity, and line slopes can be determined more efficiently using sound than through tactile means [Bre02]. Moreover, given that humans are capable of determining with more or less accuracy the location of an individual sound source in space, *spatial sonification* approaches may also be employed to non-visually portray spatial information in the three-dimensional space.



Figure 2.8: A sonified line chart being displayed by the *Sonify* iPhone application<sup>10</sup>

### Line Charts

Time charts are likely the most studied type of diagrammatic representation in the audification literature, with early work reported by Mansur et al. in the 1980s [MBJ85]. This kind of diagram may be sonified by representing each data point with a musical note, with a continuously-varying pitch being used to represent the amplitude of a curve in the y-direction

<sup>10</sup>Image source: <https://hcii.cmu.edu/mhci/capstone/2016/bloomberg/product.html>

(the higher the y-value of the data point, the higher the musical note being played), whereas time is used to represent motion in the x-direction [MBJ85]. For example, data values can be mapped to different Musical Instrument Digital Interface (MIDI) notes which are created by a smooth interpolation function going through a sample of (time, value) pairs for all data items [PBB<sup>+</sup>08]. Recognition of the potential utility of sound as a data-display medium has led to software packages that can produce different types of simple so-called “auditory graphs” (i.e. sonified line charts), which are occasionally intended for assisting blind and visually impaired students and scientists [WM10]. Examples of sonification software for statistical charts include the *Sonification Sandbox* by the Georgia Institute of Technology [DW07], *Triangle* by Oregon State University [GLS96], and the Accessible Graphing Calculator (AGC) from ViewPlus Software [Ost02].

In its simplest form, audification of a line chart may be implemented by just using the data values directly as values in the digital sound signal [PBB<sup>+</sup>08]. A line chart containing two data series may be sonified by using either the same type of sound (e.g. piano notes) for both data series or a different type of sound for each of them [BPS03]. Both data series are sonified and simultaneously output to the user through headphones, panning one data series to each ear. In this manner, both data series can be perceptually separated and compared at the same time. Using the same instrument for each data series makes it slightly easier to identify interactions between lines such as intersection points, whereas using different instruments might be better for identifying individual line features [BPS03].

Mapping of changes in data onto changes in sound (e.g. pitch or volume) is a crucial, far from trivial aspect which is understudied and for which very few guidelines exist [WM10]. Moreover, designing such a mapping may be counter intuitive at times, especially if the designer is sighted and the intended listeners are visually impaired, and the perceptual and cognitive expectations of the end-user need to be considered without making decisions simply on what sounds good to the designer [WM10]. Frequency is the most common used sound dimension in sonification of diagrams; however, different sound dimensions might be better for representing certain data types. For instance, it has been observed that frequency is better for representing temperature, whereas tempo (onset) is a more suitable representation for size (see e.g. [WK05] for an experimental assessment of a number of mappings).

Once a specific mapping has been chosen, a polarity and scaling need to be selected by the sonification designer as well [WM10]. Polarity refers to how the data dimension and the display dimension co-vary. For instance, a positive polarity indicates that if a data dimension increases, then its assigned display dimension must increase as well. A more specific example would be the frequency of the output sound getting higher as the underlying temperature represented by the chart increases. Scaling refers to how much change in a data dimension is represented by a given change in the display dimension. The slope on a line in a statistical line chart is the visual equivalent to sound scaling. As in polarity, choosing the best scaling value for representing data with sound depends on the exact type of data and display dimensions in use [WM10]. The actual slope of the scaling function also depends on both the sound attribute that is being varied, and the type of data that the sound is representing (this means that calling the same sound temperature, velocity, or number of dollars may change the way

it is interpreted by the listeners). Interestingly, blind and sighted users prefer very similar mapping configurations, with only a few exceptions. However, there are no guidelines on preferred data-to-sound mappings, as most of the research done in the field has been done with sighted users and very constrained [WM10].

### Other Sonified Diagrams

The aforementioned mapping of numerical data to sound is however a very restricted method that cannot be applied to many types of diagrammatic representation, especially those who encode non-numerical data, such as link diagrams and maps. This is because it is harder to define a direct mapping between the conceptual information represented in these types of diagram and acoustic dimensions such as frequency [MBKS08]. Nevertheless, a few approaches have made use of sonification to convey conceptual information in diagrams, albeit they are very limited and mostly focus on a single feature of a given diagrammatic domain. For instance, Brewster et al. [BCH98] have employed compound earcons to provide non-visual navigation cues in tree diagrams representing a menu hierarchy. Compound sounds, made up through the concatenation of a set of atomic sounds representing the numbers zero to found and the dot, are attached to nodes in a tree to represent their location in the hierarchy. Their research showed that participants could identify their location with over 97% accuracy, and were also able to recognize previously unheard earcons with a similar degree of accuracy.

Franklin and Roberts suggest using positional sound to represent aspects of pie charts [FR03]. The user, who is wearing headphones or is surrounded by a speaker system, is placed at the center of the virtual two-dimensional world representing the pie chart, facing its ‘zero percent’, and sounds are generated around letting the listener determine the percentage values of the chart’s slices. However, their most successful design iteration did not result in a genuine audification interface, since it involved using a codification of slice values akin to Morse system which uses long beeps to represent 10% value increments and short beeps for 1% value increments for each slice. Slices were distinguished by using different sound pitches for each one. This representation technique allowed for 100% accuracy and a low average of replays. Moreover, the authors noted that a similar design could be employed to sonify other type of diagrams, such as bar charts.

Ramloll et al. [RYRB01] explored numerical two-dimensional (i.e. tabular) data discovery using both speech and non-speech sounds. In their non-speech-output-only approach, every number in a cell of the table is mapped to piano notes whose pitch correlates to the number in a similar way we have already discussed for data values of line charts (in this case, the higher the cell’s number, the higher the sound’s pitch). Panning of the sound is used to map the position of the cell within the table’s row and column in order to allow navigation of the table. For example, when moving down a column the first value is associated with a pitch heard in the left ear, the subsequent values are mapped to pitches between the left and right ear, and the last value is perceived only in the right ear. Even though users cannot make correct numerical value estimations based on pitch perceptions and therefore speech has to be included in this kind of system, the results from the evaluation of their approach

shows that non-speech sounds can have a significant impact on making traditional non-visual interfaces more accessible to VIPs.

### Spatial Sound

*Spatial sound*, created through stereo, loudness, Doppler, or environment effects, can provide an interesting upgrade to auditory interfaces for VIPs by describing the locations of real world objects and interface modules with positions of virtual sound sources [ST15]. For example, spatial sonification has been employed to help blind persons build a mental model of an unfamiliar area of a city by using an auditory city map [HWB06]. Relevant objects, like a park or a public building, are represented as a two-dimensional area associated with a distinct sound. Sounds of nearby objects according to the user's current position are played concurrently, with the volume of each object's sound adjusted according to its distance to the user. Moreover, the auditory map uses planar three-dimensional acoustic model in order to mediate also the direction of a geographic object compared to the user's orientation (see Figure 2.9). In this manner, the listener may gain an idea of his or her surroundings. In order to get a cognitive model of the entire presented area, the users can alter their own position on a tablet and virtually walk through the city, while listening to the objects around him or her.

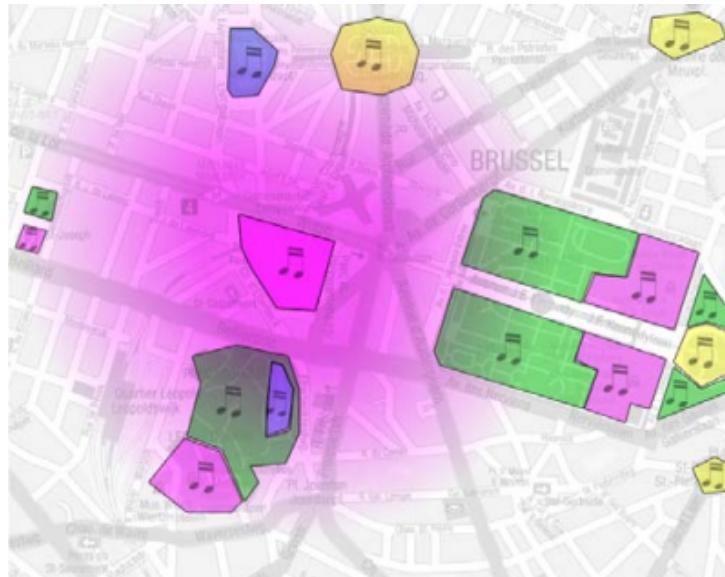


Figure 2.9: Visual representation of a landmark's sound irradiation used in an auditory city map [HWB06].

However, these approaches present three major drawbacks that restrict their usability [NR07]. First, research on spatial sonification is still in its infancy. Second, mapping visual information to spatial sound is not inherent, and its interpretation is not precise. For example, sound that is mapped to a position in the azimuth plane cannot be accurately located, in contrast to information located in an equivalent graphical visualization, whose position can be retrieved

with great precision. Third, spatial data in geographical visualizations are mapped to two-dimensional spaces, while this need not be the case for sonification. Therefore, diagrams represented through spatial sound remain a very limited approach that does not satisfy the access needs of VIPs.

### 2.3.3 Sonified Diagram Summaries

Usually, due to their lack of precision when communicating quantitative and nominal information, methods that employ sonification for the navigation of diagrams do so in order to support and enhance a more traditional user interface (see Section 2.4 for an analysis of combined methods such as audio-tactile approaches). Sonification may be an especially interesting addition to other kind of interfaces in order to provide users with a quick glance of a diagrammatic representation before they delve into a more detailed exploration using other techniques. Diagram audio summarization techniques are particularly useful for VIPs, as they improve subsequent navigation and problem-solving activities supported by the diagram. Audio summaries have been shown to help VIPs build a mental model of link diagrams more easily by constraining the possible structure (size, node types, etc.) to a small part of the complete graph-space (i.e. the range of all possible graphs the diagram of interest might match) [Bro08]. Nevertheless, summarization techniques based on sonification are hard to come across in the literature.

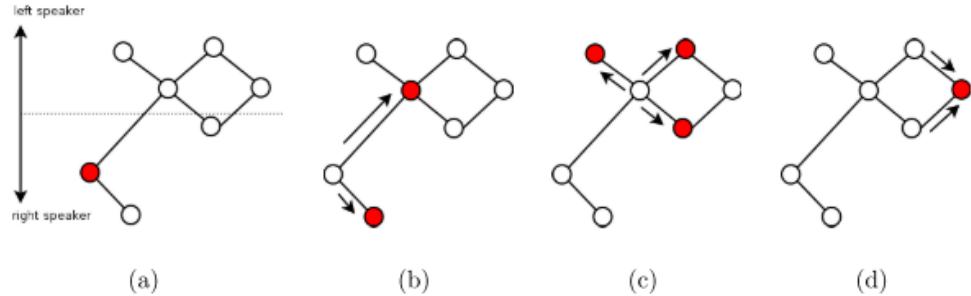


Figure 2.10: Visual representation of a sonification algorithm that provides a quick summary of the topology of a link diagram [Bro08].

Brown [Bro08] developed an audio glance algorithm of link diagrams based on earcons that describes both the topology (i.e. to which other nodes each node is connected) and layout (i.e. node coordinates). It consists in a breadth-first algorithm that follows the connections in the diagram from the left-most node (see Figure 2.10). This node's earcon is played first, and, after a pause, all nodes connected to it are played, with a shorter pause in between them. Then, all nodes connected to them are played in a similar fashion, and so on, until the last node is reached. Each node's earcon also represents its vertical spatial dimension by using a stereo axis: the further above the center of the y-axis a node is, the further left its sound is played, and vice versa. Evaluation shows that the size and complexity of a graph can be conveyed to the listener by mean of these audio glances. Some limitations with this algorithm were found though: first, it does not show the diagram's layout, which may be

necessary for its proper understanding. Moreover, the chosen starting node has a big impact on the sound of the glance and how much information can be gleaned from it.

### 2.3.4 Augmenting Sonification

In spite of its many benefits, sonification by itself is usually an insufficient means for telling nominal and quantitative information apart. Research shows that the number of different categories that can be represented by means of sonification is very small. Miller [Mil56] estimated that only a maximum of six different pitches can be employed so that the listener will never confuse them. In other words, no matter how many alternative tones the user is asked to judge, the best they can be expected to do is to assign them to six different classes without error. This crucial limitation does not only apply to pitch; the channel capacity for absolute judgements of loudness is approximately the same, allowing persons to distinguish only around five perfectly discernible alternatives [Mil56]. In comparison, the channel capacity for judgements of visual positions seems to be significantly larger, with an estimated 10 to 15 perceivable distinct positions along a linear interval [Mil56]. It is nevertheless important to note that there are other variables of visual stimuli that cast a lower number of distinguishable categories, as it is the case when judging hue and brightness levels, or the size of a number of squares (estimated at about five categories according to [EH55]). In any case, presenting absolute data values by means of sonification is extremely difficult unless the listener has perfect pitch, which is not the case for most persons [Bre02].

Non-speech sound presents some other drawbacks when used to display information. Among them, its lack of orthogonality is worth noting [Bre02]. This means that changing one attribute of a sound may affect the perception of other sound attributes by the listener. For example, changing a note's pitch may affect its perceived loudness, and vice versa, which might easily confuse the listener. Moreover, information displayed through sound is transient i.e. it disappears as soon as it has been presented [Bre02]. Remembering previously played sounds when many of them have been already reproduced requires a big mental effort which makes building a complete mental model of the information a hard task. To cope with this issue, some sort of replay method should be available in sonification technologies [Bre02]. Moreover, perceiving and understanding a large number of simultaneous sounds is generally not possible [ST15], making sonification approaches suffer from many of the same drawbacks that traditional linear (i.e. text-based) approaches do. The benefits and drawbacks of linear approaches were already discussed in section 2.1.

Another very important concern expressed by users of sonification technologies is annoyance due to sound pollution. While some sounds are functional and indulged by users, others may be regarded as noise that impedes proper comprehension [BGB94]. This may apply to the user of the assistive technology as well as to other persons in the same environment who overhear it. Last, current sonification devices are difficult to work with for diagrammatic representations other than very simple statistical charts, as they generally do not provide means for summarizing information and highlighting salient data features [FLST13]. Some authors note how sonified diagrams do not work well when they are complex e.g. line charts

containing more than three lines, or when data is not time-dependent, such as in pie charts [FPRL06].

In conclusion, due to its many limitations sonification is generally accepted as a supplemental tool to convey visual information, not as the central mode of communication [Kea14]. Furthermore, other techniques are commonly favored over sonification approaches by VIPs performing tasks on diagrams. For example, research on non-visual access to statistical charts shows that most users show a strong preference for speech feedback over sonification, which is used only rarely [FLST13]. Most sonification methods in use today are thus employed in multimodal devices that combine it with tangible media or speech. Some of these hybrid approaches are discussed in the following section.

## 2.4 Hybrid Approaches

Hybrid (also known as *multimodal*) methods leverage the benefits of both tactile and audio techniques to create all-round non-visual alternatives to graphical information, in which each modality supports and enhances the other. Research has shown that the difficulty of reading a tactile diagram becomes greatly reduced if the graphic can explain itself to the reader [Gar14]. Labels and other textual descriptions of a diagram's elements, as well as some other features of important graphic objects themselves can be spoken out to the visually impaired reader, usually when its associated element is touched, pressed, tapped, or clicked on a tactile adaptation. Besides reducing the cognitive load on the reader's working memory, multimodal approaches also let authors of accessible graphics convey much more information about the graphic than its feasible using a stand-alone tactile copy citeGardner.2014. Moreover, they can be configured so that the system can speak different messages for the same area according to the system's current operating mode [BE98]. Audio-tactile diagrams are also known as "talking tactile diagrams" [BE98].

One of the first multimodal approaches for non-visual exploration of graphic information was the *Nomad* tool conceived by Parkes in the late 80s [Par88]. It consisted in a briefcase-sized standalone device with a touch sensitive surface enabling finger selection of points or areas for sound output, onto which an A3-sized tactile graphic could be superimposed. In response to haptic input from the user on the touch surface, Nomad would give speech feedback or respond with other structured sound such as a constant pitch generated along a preferred path on a map. However, this early approach could not be favorably employed to navigate everyday graphics, as the resulting auditory feedback presented much noise and interruptions that impeded its correct usage, and had to be finally discontinued.

A similar approach was carried out by Landua and Wells in their *Talking Tactile Tablet* (TTT) [LW03]. The TTT is an aluminum and acrylic tray, connected to a host computer via USB, with a hinged frame into which a collection of tactile sheets can be mounted. These tactile sheets must be produced according to a specific layout compatible with the TTT interface (see Figure 2.11). Users then may proceed to close the frame to hold the sheet motionless against a touch-sensitive interface. Blind readers then interact with the computer by pressing on the tactile drawing; the computer registers the position of each pick and

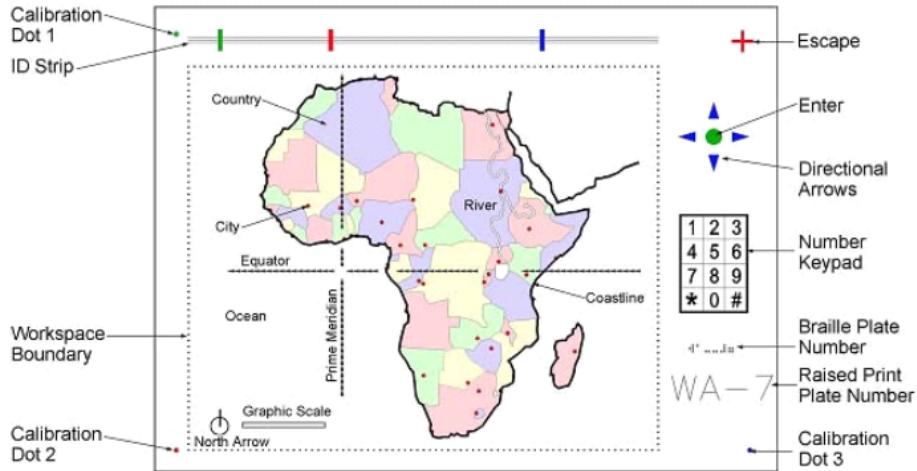


Figure 2.11: The Talking Tactile Tablet (TTT), a multimodal device combining haptic and speech output, here shown with a mounted tactile map [LW03].

then returns an appropriate audio response. Different programs allowing specific interaction with a set of graphics can be created using the provided TTT Authoring Tool. Even though the tactile sheet is static, the TTT can be used to display some changing information by assigning variable values to place-holding tactile shapes, which allows the user to focus on only those structures relevant to the current discussion without losing a sense of their spatial distribution. The TTT's approach to making maps accessible has been unanimously praised by its users (even though they have been limited to children in a general classroom context), who had the impression that the combination of touch and speech provided them with a cromulent overview of the data. Moreover, their approach is presumably extensible to other diagrammatic representations such as statistical charts and tables [WB06].

A classic hybrid approach to accessible link diagrams was introduced by Blenkhorn and Evans in 1998 [BE98]. Their work focused on a tool, named *Kevin*, which provides access to one particular type of software engineering diagrams, namely RTSA data flow diagrams, using tactile maps either augmented with Braille labels or speech. A spatial simplification of the diagram needs to be previously undertaken, as to provide a generic tactile representation so the same tactile display can be used to represent any diagram of the given domain. In this specific case, RTSA diagrams are transformed into their equivalent tabular representation, known as N2 chart. This equivalent but simplified diagram is then overlaid on a touch-sensitive tablet. Below and to the right of the chart is the control area, which is divided into a number of smaller areas that are used to control the operation of the tool. These areas contain tactile icons (hapticons) allowing users to perform common operations such as opening a model, navigating through the hierarchy, creating new charts, etc. As the user moves along the diagram, appropriate speech messages are spoken to display the current node's content or navigational cues. Being an early approach, the Kevin tool is very limited in scope as it only allows navigation of a specific domain of link diagrams. Newer approaches, such as the

## 2. STATE OF THE ART

---

Talking Tactile Tablet previously discussed, provide much more flexible capabilities to their users.

A prototype device developed by Wall and Brewster [WB06] consisted on a combination of a graphics tablet and a VTPlayer device (a mouse based apparatus that incorporates two 4x4 pin arrays providing basic tactile feedback), controlled with one hand each, allowing sight impaired users to interact with bar charts in a haptic-supported point-and-click style. The graphics tablet worked as an absolute positioning device which provides a spatial frame of reference (about the size of a landscape A4 piece of paper) for grounding exploration and communication, whereas the dynamic data contained in the bar chart was rendered using the VTPlayer device. The tablet was augmented with tactile  $X$  and  $Y$  axes, providing a quick spatial reference for the exploration of the chart, and the VTPlayer device offered tactile feedback for navigation and indirect access to data values, similar to the tactile representation of areas in a hard-copy tactile graphic. Speech output was offered to the user through buttons placed on the stylus used to interact with the graphics tablet. These speech commands provide information about the bars (titles and values), preventing the tactile representation from becoming too cluttered while providing details on demand to the user. Evaluation of the prototype yielded very favorable reactions from its users, who were able to easily perform a number of tasks and felt that the representation was equivalent to hard-copy paper diagrams.

Goncu and Marriot [GM11] presented another multimodal approach for presentation of graphics in their *GraVVITAS* (Graphics Viewer using Vibration, Interactive Touch, Audio and Speech) device. Their method is based on a touch sensitive tablet PC that tracks the position of the reader's fingers. Graphic content specified in SVG may be displayed on the tablet, and haptic feedback is given to users by small vibrating motors attached to their fingers and controlled by the tablet PC. This way, the user can haptically determine the position and geometric properties of graphic elements. Audio feedback is given in addition to help the user with navigation and to allow the user to query a graphic element in order to obtain non-geometric information about it. They place a special focus on attaining what they call "functional equivalence" of the graphic, this is, a strong form of equivalence in which the spatial and geometric nature of the original graphic is maintained. The benefits of retaining the spatial features of graphical elements were already discussed in Section 2.2. Using SVG as the image format in their tool allows them to use invisible metadata associated with the shapes to control the tool behavior (see Section 2.1.7 for a concise discussion on use of metadata for accessibility purposes). Each shape must have the following metadata specified for the GraVVITAS tool to function: ID, vibration level (for the shape's edges), audio volume level (for its interior and boundary), text description to be read out when the shape is queried, and the name of a non-speech audio file for the generating the sound associated with the shape during navigation. Their prototype was evaluated for navigation of floor plans and line charts, with most participants reporting being satisfied with the prototype, even though many expressed that more training would be desirable in order to feel more comfortable with its usage.

Another similar multimodal approach is the *GPLUMB* system, developed by Calder et al. [CCLX06]. It displays a link diagram on a tablet PC and uses auditory cues that help a blind

user navigate it. PLUMB provides two different navigation interfaces to blind users. The first one consists in a tablet PC based navigation system that allows the user to trace the graph by receiving audio cues from the device about his or her current location in the diagram. This interface works well for teaching users the geographical and geometrical relationships between the nodes in the link diagram, but is often slow and depends on the precision of the user's hand movements. The second navigation interface is keyboard-based, in which a series of keyboard shortcuts have been developed that allow users to quickly obtain information about edges and vertices and navigate through the diagram, but sacrificing information about the topology of the diagram. An interesting feature of the PLUMB tool is the auto-complete search function for choosing a starting point in the graph. In this way, users may jump to any vertex from where to start exploring, or they can use the pen to choose a location on the tablet screen where the diagram is redrawn.

This section has only scratched the surface of current available multimodal approaches to accessibility of diagrams. Many other multimodal devices can be found in the pertinent literature, see e.g. [ZMW15, APS<sup>+</sup>10, MB06, SDMRG03, RKPP02, PSB<sup>+</sup>02, SMLG03, Gao13]. What these approaches have all in common is their requirement for specific tactile or haptic devices along with a conventional computing device. As previously discussed in Section 2.2, pin array devices, force feedback (haptic) devices, and other computerized tactile equipment are not a convenient means of displaying graphical information in most environments due to their high cost, specificity, and cumbersome set-up and portability. Even though empirical evidence has demonstrated the many benefits of supporting audio output by tactile means in accessible diagrams, the presence of these crucial drawbacks is likely to dissuade those who purchase tactile devices (as opposed to those who actually use it) in schools, colleges, and public libraries, to buy them and support their development [LW03]. Moreover, only simple graphics may be represented in all these approaches due to the reduced sized of the available haptic displays and their low resolution, and, furthermore, it is not feasible to represent dynamic scientific graphics through tactile displays given the status of current technology [Kea14].

Therefore, until usable zoom-in and panning capabilities are available in inexpensive tactile devices, complex diagrams are more suitable to be represented non-visually by means of sound and speech. Recent methods have been thus shifting to web-based, audio-only approaches that do not require of any specific hardware or software beyond a traditional screen reader and therefore do not interrupt the usual workflow of reading an electronic document or website. Besides sonification approaches, already discussed in Section 2.3, non-visual access to graphics based on dialogue pose an interesting alternative that does not require of haptic means to convey the meaning of the graphic. Dialogue-based approaches to accessibility of graphics are discussed in next section.

## 2.5 Speech-supported Approaches

### 2.5.1 Speech Interfaces

Speech-based technologies (c.f. Chapter 5) allow users to communicate with a computing device or related services without any physical interaction device by leveraging a skill that is mastered early in life, the use of *natural language*, having thus the potential to be more natural than interactions using other technologies [FS09]. They have been steadily growing in popularity and have currently reached a level of maturity where speech-based interactive systems are technologically possible and commercially viable [Jok09]. Speech interfaces are being currently used in a widespread spectrum of applications, including for example information-providing systems, speech translation, Web navigation interfaces, and call routing software [Jok09]. A classic example of dialogue-based application is the so-called How May I Help You (HMIHY) technology, introduced by Gorin et al. in the late 90s [GRW97]. Their focus laid on a system whose goal was to understand speech-based user input sufficiently to route the caller to an appropriate destination in a telecommunications environment. Applications in which spoken interaction is particularly effective are characterized by complex domains that are difficult to model in the detailed and exhaustive way required by traditional technologies [Jok09]. Some of these applications include trip planning and assistance in machine maintenance, in which the users may not know what the different options and alternatives are for a planned trip, or how to classify maintenance problems using technical terminology, but can reach satisfactory solutions through natural language queries.

Speech interactions may be categorized into three main groups [FS09]: *speech output* systems, *speech recognition* systems, and *spoken dialogue* systems, also known as *Natural Language Interfaces (NLIs)*. Speech output systems, also known as *talking devices*, speak out information while leveraging other technologies (such as the keyboard and mouse) for input, and are the most commonly researched speech-enabled approach to accessibility of graphics. Some of these technologies and their applications for accessibility, especially on the WWW, are addressed in this chapter in Section 2.1.3, Section 2.1.4, Section 2.1.5, and Section 2.7.3. Speech recognition systems include applications that make use of speech for input while relying on other technologies to display their output. Examples may include speech-based cursor control in a Graphical User Interface (GUI) and speech-based dictation systems. Finally, spoken dialogue systems, also known as question answering systems, speech interfaces, or NLIs, include applications that utilize speech for both input and output, such as telephony systems.

Most question answering systems focus on factoid questions i.e. questions that can be shortly expressed with simple facts such as a personal name, temporal expression, or location [JM18]. In turn, factoid answering systems generally fall under one of two categories: Information Retrieval (IR)-based question answering and knowledge-based question answering [JM18]. Whereas the former fetch their answers from a vast collection of textual information such as Web documents, the latter builds a semantic representation of the query, which is then used to query databases of formal facts. Dialogue systems typically integrate many components, such as a speech recognizer, a database (or some other knowledge base) back-end that stores the

information to be retrieved or modified, and a dialogue strategy [SKLW00]. In turn, dialogue strategies range from simple approaches designed to follow a step-wise interaction script and to direct the user to produce utterances that fit into predefined utterance types, to advanced textual question answering tasks based on semantic parsing, symbolic representation, and deduction systems. More recent, state-of-the-art approaches may employ deep language understanding components that are capable of dealing with a greater amount of open-ended, free-formed questions, such as neural networks and Reinforcement Learning (RL), often combining both technologies.

### 2.5.2 Speech Interfaces in Accessibility

Despite their increasing popularity, research on this kind of technology is rarely motivated by the goal of accessibility [FS09]. Moreover, within the field of universal access the majority of speech interaction research has focused on speech input as an alternative to traditional computer input methods for persons with motor impairments, whereas other researchers have laid their focus on issues relevant to older users, children, and people with hearing, cognitive, or communication impairments [FS09]. This means that research on NLIs has generally ignored the needs of VIPs. Speech in the field of non-visual universal access to graphics has been most commonly employed in combination with tactile means in hybrid technologies (see Section 2.4), but very few pure dialogue systems can be found in the literature, in spite of the fact that verbal communication is considered by some authors as the most suitable means of presenting visual information to blind users [Kea14].

The most common example of speech-enabled assistive technology for the blind is mobility systems. These technologies are essential components in the lives of many VIPs, who require support in autonomous outdoor and indoor navigation activities as part of their daily routines beyond the traditional use of the cane, mainly in public places away from familiar environments like home or the workplace [DK08]. A speech-enabled device may act as a virtual tour guide to the user, providing them with guidance for travelling around an area and allowing them to discover beforehand the detailed navigation information relevant to the intended travel. These technologies are generally enhanced with user-tracking technologies that allow the system to know the current position of the user and with traditional devices such as tactile marks on the ground [DK08].

Other authors note how blind persons are curious about the pictures they take and ask questions about them to their sighted peers [GLS<sup>+</sup>18]. Moreover, studies show how blind people are able to process information given in natural language (more specifically, oral information) better than large type-readers, performing significantly better on delayed recall [Bro71]. Therefore, dialogue has the potential of being a holistic non-visual alternative to graphics that can be presented and accessed by VIPs. However, given the lack of research in this area, a more profound evaluation of dialogue-enabled systems to graphics needs to be initiated in order to confirm or dismiss this hypothesis. The rest of this chapter thus focuses on existing speech interfaces to graphical content, especially in the field of accessibility. Systems in which input is given by the user via keyboard commands or some other non-haptic modality will be considered as well, as long as output is given by means of natural language. We will

start out by discussing the VQA challenge, which entails answering free-formed questions about photographic content and is currently being investigated in the deep learning, computer vision, and knowledge representation and reasoning (KR) communities. Afterwards we will move on towards accessibility-oriented technologies, placing a special focus on Communicative Images.

### 2.5.3 Visual Question Answering Systems

The problem of automatically answering open-ended questions posed in natural language about visual content is known as the Visual Turing Test [MF14] and methods that aim to tackle it are called Visual Question Answering (VQA) systems [AAL<sup>+</sup>15]. This is a problem which has been receiving great amounts of research attention after the “deep learning revolution” of the early 2010s, in which advances in hardware and artificial neural networks research boosted the progress of machine learning methods for improved automatic object recognition and general image processing. Section 2.1.6 already discussed current Artificial Intelligence (AI) methods for automatic generation of verbal descriptions of images. Even if VQA approaches may employ similar computer vision strategies for object detection and neural language models, these methods require deeper multi-domain knowledge and a potentially different, more advanced set of AI capabilities including knowledge-based and commonsense reasoning [AAL<sup>+</sup>15] given that both the possible questions and their answers are unrestricted. In order to tackle this challenge VQA methods seek to progress directly from image features to text, as opposed to relying on high-level semantic annotations of the images [WSL<sup>+</sup>16].

VQA systems have a great potential in improving access to photographic content to blind persons, as asking questions about a photography’s content to a sighted counterpart is often the only means by which blind persons can understand it e.g. in social networking sites [Wu16]. However, these methods are not yet considered ‘deployable’ i.e. they perform significantly worse than humans on the same tasks [SCD<sup>+</sup>17]. Moreover, and more importantly in the context of this thesis, accessibility concerns are only briefly noted by their authors in potential use cases, as the current research interest of these methods mainly stems from their uses in other fields such as image search and retrieval, classification, and understanding in complex AI systems. Only very recently have some authors mentioned accessibility as a desirable next step for vision applications, who consider VQA advances critical for empowering blind persons in their daily lives [GLS<sup>+</sup>18, SHL<sup>+</sup>16]. This also entails that VQA to diagrammatic representations are almost nonexistent in the literature, as the vast majority of approaches focus on real-world photographic content. Even current curated datasets of graphics to be employed as training and evaluation sets on VQA systems), including those generated with accessibility in mind, are limited to real-world pictures and generally do not include diagrams of any kind, see e.g. Figure 2.12. Some exceptions we have been able to find are the *FigureSeer* dataset [SHL<sup>+</sup>16], containing diagrams stemming from CiteSeerX papers indexed by Semantic Scholar; and *FigureQA* [KMA<sup>+</sup>18], a synthetically-generated dataset containing question-answer pairs grounded in 100,000 images.

Most commonly, related machine learning methods are concerned with tackling the problem of automatic diagram classification i.e. assigning diagrammatic figures stemming from arbitrary

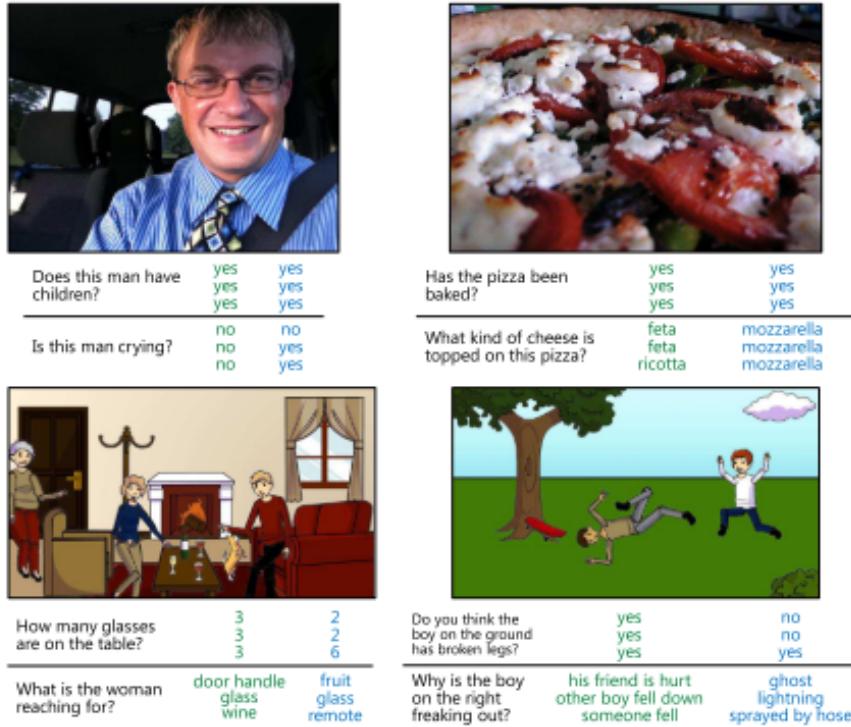


Figure 2.12: A few example images with associated questions and answers from a dataset employed to build and evaluate VQA systems [AAL<sup>+</sup>15]. Diagrams are left out in almost the entirety of VQA training datasets.

sources to their corresponding sub-category (bar chart, pie chart, histogram, etc.) as well as extracting their most salient associated metadata, such as figure captions, see for example [CMK<sup>+</sup>13] and [CD15]. However, these methods do not include interactive means for querying the parsed diagrams. There are nevertheless a few interactive VQA-inspired approaches to accessing diagrams. The most notable one probably is the *FigureSeer* framework of Siegel et al. [SHL<sup>+</sup>16]. *FigureSeer* allows to automatically detect and parse the results figures embedded in online research papers with the goal of helping researchers discover relevant information within the plethora of available research on the Web. This is especially relevant for researchers worldwide since current search engines of scientific contributions only allow for searching within the text content of the papers, whereas many crucial inferences possibly contained in their plots are not reachable. *FigureSeer* accepts PDF documents as input, automatically producing as a result a structured representation of the results-figures within it. The given representation is produced by leveraging a graph-based reasoning approach using Convolutional Neural Networks (CNN). It contains the most salient elements detected in the graphic such as axes, legend, and individual data points. These elements may moreover be queried by users through a basic query-answering system. This was implemented as a proof of concept for the suggested framework, allowing simple questions in natural language following specific templates to be inquired. Query templates are made up of two parts (slots to be filled

in): the dataset and the specific metric thereof that the user is interested in. For example, the query “best precision at 0.3 recall on BSDS dataset?” would fit the template’s slots with ‘BSDS’ and ‘precision at 0.3 recall’, respectively. When a query is asked, all relevant figures from the corpus are retrieved first, their content afterwards is parsed into a data-table representation. Finally, the query is run against the data-table and any matching quantity extracted and presented to the user.

VQA systems are the latest advance in NLIs to graphical information and have achieved very impressive results in a short time span. They are the most promising current approach to querying diagrams stemming from a diverse range of sources using natural language. However, their lack of orientation towards accessibility purposes (including lack of support for autonomous wayfinding strategies, access-related annotations and custom element tags) and their inaccurate inference results in complex diagrams (e.g. current approaches have obtained an average accuracy of 45.3% in simple numerical tasks [SHL<sup>+</sup>16]) entail that VQA systems do not satisfy yet the needs of blind users trying to access diagrams in an autonomous, effective fashion. On the other hand, Communicative Images were designed with accessibility in mind from their inception, and are thus of special interest for the goals of this thesis. They will be therefore described in a separate section of this chapter.

### 2.5.4 Other Speech-supported Approaches

Besides VQA systems (section 2.5.3) and Communicative Images (characterized in depth in section 2.6), no other current approach found in our literature review allows VIPs to access graphical content by means of natural language only. However, there exist systems, mostly interactive accessible software tools, which provide natural language support of some kind, usually in the form of speech output or verbal instructions and system support to keyboard-enabled features. This section discusses some of these speech-supported approaches.

Natural Language Interfaces (NLIs) have been briefly explored in order to address usability issues with navigating and editing complex diagrammatic representations by sighted persons. For instance, Spiliotopoulos et al developed a speech-enabled navigation and editing approach to complex graphs depicting the structure of an ontology [SDK15]. They implemented a set of voice commands covering the functionalities of an already existing ontology graph authoring software, including commands that initiate content-free interaction with the interface and those that include actual content of the model/domain expressed by the ontology. User evaluation yielded very strong preference of speech for search-related actions compared to the non-speech-enabled interface, especially when searching for nodes containing long textual contents. However, for navigation-related tasks, the visual interface was generally preferred. Even though the conclusions of this study cannot be extrapolated to the needs of blind users, they demonstrate how dialogue-based approaches may be of great effectiveness for non-visually comprehending complex diagrams, as sighted users also find them useful in some use cases even when they can already profit from the numerous cognitive benefits of information visualization.

Metatla et al. discuss an approach for providing non-visual speech- and earcon-based construc-

tion of relational (i.e. link) diagrams using multiple perspective hierarchies and two interaction (guided and non-guided) strategies [MBKS08]. Synthetic speech output is employed to communicate contextual, navigational, and content information about a link diagram to the user. Content information reflects the semantics of the graphical objects such as names and labels, links' directions and types. Navigational information reflects the outcome of browsing actions such as expanding or collapsing branches. Contextual information relates to the category of the current navigational branch the user is located in. Simple diagrams are comprised of just two category types: nodes and links. Link types and directions are communicated non-verbally via specific earcons in the form of ambient sounds. Besides simple navigation, their system also allows blind users to construct link diagrams by editing individual nodes either through a guided or an autonomous strategy. In the guided the user and the system engage in a conversation-like interaction where they collaborate by exchanging information to perform the desired task. When details to perform an action are needed, the system prompts the user for further information. By responding to each prompt, the user is guided through the necessary steps to accomplish the editing action. In this manner, the user may perform an action on any given item without having to locate it within the diagram. On the non-guided strategy, the user must autonomously explore the appropriate path to the node he or she wishes to edit before performing the action via a given short verbal command (e.g. "remove"). Evaluation performed with sighted participants demonstrated that the auditory interface with a multiple perspective hierarchy was a suitable means of constructing simple link diagrams. Moreover, the non-guided diagram construction strategy was established to be more efficient than the guided strategy regarding task completion time, diagram comprehension, and general user satisfaction. An important takeaway is that having the system act as an active intermediary agent between the user and the hierarchy was found to significantly increase interaction times.

*Kekulé*, developed by Brown et al. [BPS03], is a system for non-visual exploration of link diagrams representing the structure of chemical molecules, in which users issue keyboard commands and text strings representing output are sent to a speech synthesizer and spoken out to the user. Their authors emphasize the choice of audio as the exclusive output modality because of the "widespread use and relative cheapness of audio equipment". Input is given by means of typed commands that allow users to zoom in and out of a molecule i.e. examine it at different levels of detail, and move between connected nodes in the graph (atoms or groups of atoms). Moreover, some commands let the user fetch a summary of the graph or ask for supplementary information that helps them build a mental model of the graphic, such as spatial position. Navigation starts at the topmost hierarchy level, this is, the chemical molecule as a whole. Afterwards, the user may issue a 'zoom in' command to go to the next level of the hierarchy, which lays out a number of more detailed sub-components that make up the molecule. Then the user may move to any of these sub-components (e.g. an amino acid), and emit a new 'zoom in' command to navigate this sub-element's own sub-components in greater detail. This process goes on successively until the bottom-most level of the hierarchy is found, where individual atoms lay. The user may at any time go up one level in the hierarchy by means of the 'zoom out' command. Additional commands can be issued to know the current spatial location on the diagram (e.g. 'top-left'), to obtain a summary of the

## 2. STATE OF THE ART

---

components of the current node, and to get the current location in the hierarchy. Finally, a simpler connection-based browsing mode is also available, which allows users to move between the components of the molecule through its connections and to determine how they are connected to one another.

An interesting feature of Kekulé is its ability to recognize implicit features of the domain and to chunk information into summaries that can be requested at any time by the user, such as atom clusters acting as functional groups. This information may sometimes be explicit to a sighted expert user reading a diagram, but only implicit in the available data model of the graphic or to readers less knowledgeable in chemistry notation. Clustering of this domain knowledge allowed skilled users to reduce their efforts in understanding the diagram. On the other hand, it aided to the confusion of non-expert users. Another big challenge found during evaluation was related to understanding the current location. Navigating the graphic was shown to be confusing and keeping in mind their current position imposed an important working memory load to the users. Finally, dealing with cycling structures also posed an important usability challenge. Kekulé's authors note how adding some sort of bookmarking or annotation capabilities (perhaps through non-speech sounds) may help overcome this issue through computational offloading of the user's working memory.

One of the most salient examples of natural language-enabled access to diagrams with focus on accessibility is the *iGraph-LITE* system developed by Ferres et al. [FLST13]. iGraph-Lite is an “assistive technology designed to help blind and visually impaired individuals interact with graphical representations of line graphs using natural language through key commands and a text-to-speech (TTS) engine”, its main objective being “language-based interactivity with the user, rather than describing a graph, communicating its intended meaning, or the generation of a static text”. The iGraph-LITE tool is made up of three subsystems:

1. A knowledge representation (KR) system that enriches a basic semantic representation of line, bar, and combination graphs.
2. A natural language generation (NLG) system that produces a static description of a graph.
3. An interface that allows users to navigate the graphic by means of keyboard combinations in a likewise manner to how popular screen readers are used to navigate a GUI.

The underlying knowledge representation is specified as an XML representation of the raw data the graphic depicts. The XML counterpart of a graphic can be automatically generated from the Object Model (OM) of graphics in common plotting software, such as MS Excel™. Semantically underpinning graphs by means of Web Ontology Language (OWL) ontologies in a similar manner to that of Communicative Images (section 2.6) has also been considered. However, it appears that OWL support was finally not implemented [Fer09]. These automatically generated XML files are afterwards manually curated and enriched to fill in any missing, wrong, or incomplete fragments, and finally provided to iGraph-LITE as input. Enriched graphics may then be navigated via keyboard commands, and a language message

generator (named *NLG* module) generates relevant output in natural language to the user. The use of keyboard commands as navigational means is inspired by how natural languages navigate graphical user interfaces with help from screen reader software (see section 2.1.5).

A static high-level description of the graph may be automatically generated by the system as well if the users asks for it. Available commands allow the user to sequentially navigate each data point in a line or bar chart by issuing ‘move forth’ and ‘move back’ commands, to retrieve the diagram’s summary, and to ask for the current position. In case that the user prefers to navigate the diagram in a faster, non-sequential, fashion, a ‘skip’ command was also available, which allowed them to jump over a number of data points in either direction. However, this instruction was rarely used as it tended to confuse the participants. Finally, a ‘help’ command could also be issued in order to obtain information about available options. On a second iteration of the system’s design, new commands were added. Some of these new commands allow users to immediately request specific bits of information that were already included in the summary, such as axes information and chart titles or the first and last data points in the graphic. Moreover, navigation to a data point was broken down into sub-commands allowing the user to ask for its value or slope information only, as opposed to always offering the user full point information.

Validation of iGraph-LITE yielded very positive results with visually impaired participants reporting that it “was superior to the means they normally use for graph interpretation (tactile graphs, other people interpreting graphs for them)” and that it enabled them to autonomously access diagrams, eliminating their reliance on sighted helpers [FLST13]. Its authors remark how “the main attraction of iGraph-LITE was [...] its ability to satisfy [VIPs’] desire for autonomy”. Among the interesting insights from the evaluation is the fact that blind participants were much more likely than sighted participants to sequentially navigate the chart (i.e. moving left or right from the current position without skipping any data points), leading to a higher number of total commands used to perform the same tasks and confirming “the benefit of breaking down the graph description into single elements”. However, it is important to note that iGraph-LITE is not a dialogue system but a speech output system, and thus it presents a number of important drawbacks that constrain its potential take-up by the visually impaired population. Namely, its proficient use requires a great deal of training, in which users had to study the list of available commands and go through a number of phone training sessions with the experimenter [FLST13]. This lengthy training takes back from the declared autonomy of the users working with iGraph-LITE, whereas an habitable and usable natural language interface can require little to no training, allowing users to effectively express themselves within the constraints imposed by the NLI system using language they think the system will understand [OMBC06]. Moreover, iGraph-LITE is constrained to graphics stemming from a number of common plotting software applications, and thus it provides no means of supporting raster or vector images originating from arbitrary internet sources that users may encounter in their daily life. Communicative Images (discussed in section 2.6), on the other hand, theoretically allow for the semantic enhancement and navigation of any image, given that a suitable ontology and dialogue strategy exist for its specific domain.

A major drawback of the methods mentioned so far is that user interaction is limited to

## 2. STATE OF THE ART

---

fixed navigation and querying of the diagram, not providing users with custom annotation capabilities, something that Communicative Images have shown to be of great aid to enhance the effectiveness and user satisfaction of dialogue-based systems. We have just been able to find a single approach that enhances diagram navigation by means of user-provided annotations, that of Brown described in [Bro08] and which expands on the main ideas of Kekulé [BPS03], already discussed. Brown's approach, much like iGraph-LITE, also consists in a keyboard-enabled system for non-visual navigation of diagrams (more specifically, link diagrams depicting instances of family tree and logic circuit domains) with output given to the user in natural language. However, the enhanced approach additionally includes user annotation, summarization, and node identification capabilities that help the user with navigating complex diagrams. Link diagrams and their semantic augmentations are formally represented via Resource Description Framework (RDF) files. These augmentations included domain-dependent numeration of graph nodes in order to uniquely identify each one, user-defined labels for each node, musical notes attached to nodes, visit histories (via a non-speech sound that was played after re-visiting a node); and domain-specific annotations such as to which generation of a family a person in a family tree belongs to, the relationship between nodes and the user-selected home node (e.g. father-son/daughter relationships in family trees), and the direction of directed link connectors in logic circuits.

As in many of the already discussed approaches to non-visual access to graphics, Brown's method starts by displaying an automatically computed summary of the diagram to the user. This summary includes non-speech information about the diagram's topology and was already introduced in section 2.3.3. Summarization was also provided by reporting the complexity of the graphic in a single sentence using an average of the nodes' complexity, measured via the Augmented Valence Complexity index [RP02]; and by listing all nodes in a hierarchical fashion in a domain-dependent manner in the same manner Kekulé had previously done to aid with the navigation of chemical molecules. Users may then navigate the diagram by starting on an initial node and moving afterwards sequentially from node to node, with the focus always on one specific node and automatically shifted to the next node the user moved to, but summaries about the nodes laying below the current node in the diagram hierarchy may also be enquired at any time during navigation. Users move from node to node by issuing 'next in list' or 'previous in list' commands. While on the current node, the user may issue commands to fetch its associated properties (e.g. birth date and sex of a person) or its immediate connections (identifiers of all nodes it is liked to) in the diagram (e.g. a given person's husband's name, and daughter's name). Lastly, several search functions, intended to echo the cognitive benefits of visually grouping elements on close proximity, were implemented. Commands included search by node name or by certain annotations, such as node's numbers, labels, and musical notes, with matching results being laid out to the user in a list. Choosing one of the list's elements then lets the user move to its respective node in the diagram, and continue navigating thereupon. The prototype was implemented as a self-voiced application whose commands are accessed via a set of grouped menus mimicking the behavior of a Windows application being operated through a screen reader. Commands are also reachable via keyboard shortcuts.

Evaluation was performed by 20 participants, 2 of whom were blind, 4 were visually impaired of unspecified degree, and 14 were sighted. No visual display was given to any participants regardless of the state of their vision. Results were collected by a combination of different analysis methods, both quantitative (task success and load index scores) and qualitative about motives and strategies employed. Evaluation results showed how enhancing the graphic with the techniques earlier described led to increased task success rates, reduced mental demands, and improved user satisfaction when performing tasks supported by a diagram. Moreover, no significant differences were found in the qualitative results between sighted and blind participants. On the other hand, some of the suggested annotation techniques proved being of little use, including non-speech summaries, visit histories, and node complexity information. Brown's prototype is the sole example we could find that tries to compensate in a rigorous manner, via software means, for the lack of some of the cognitive benefits that visualization allows sighted persons to enjoy. Nevertheless, the employed software techniques were evaluated on link diagrams, and the question of whether they may be of benefit in other kinds of diagrammatic representations remains open. Some of these methods have thus been considered in the framework described in this thesis, and will be discussed in greater detail in Chapter 6.

Offering the available commands to the user in a tree list or some other traditional interface as implemented by the approaches discussed in this section presents important drawbacks when non-visually operating them, especially as the number of available options increases [PE03]. This problem is again related to that of requiring users to go through a lengthy training process before using the proposed solution, which could be at least partially solved by using a speech-based approach for user input as well as system output. Generally speaking, human-computer interaction would be preferred to traditional human-computer interfaces when interacting with complex systems non-visually, as the latter tend to focus on a requirement for users to have the ability of seeing the computer screen to enable effective communication with the system. On the other hand, human-computer interaction, such as that enabled by dialogue interfaces, demand a dynamic process of communication in which devices become central as the tools to mediate achieving communication objectives [Bab97]. Enabling users to engage in full dialogue with the image (e.g. a diagram) therefore appears to be a fitting alternative to the methods discussed thus far. This is precisely the principle under which Communicative Images, discussed next, operate.

## 2.6 Communicative Images

The only (to the best of our knowledge) purely natural language-based approach to interacting with graphics that focuses on accessibility for Visually Impaired Person (VIP) are the so-called Communicative Images devised by Kopeček, Ošlejšek [KO11], and Plhák [Plh14]. Communicative images are defined as two-dimensional graphical objects (i.e. images) integrated with a dialogue system and being equipped with an associated knowledge base, such that communication with a user exclusively via natural language is possible [KO11]. Moreover, a Communicative Image may also learn from the communicative process with the users and enlarge its knowledge base with previously unknown facts. More importantly, they have been

## 2. STATE OF THE ART

---

designed with accessibility for blind persons in mind from their very start. Communicative Images have been implemented within the Graphics Accessible to Everyone (GATE) project, whose three main goals are [KO08a]:

1. To enable users with a visual disability to easily create some limited form of computer graphics using a system for image generation by means of dialogue.
2. To provide users with a visual disability with methods of and support for investigating pictures by means of dialogue.
3. To develop utilities for effective picture annotation.

These goals, and more specifically the two latter ones, are of great relevance in the context of this thesis. We will therefore take a deeper look at Communicative Images and their elements, and will note their benefits and limitations. This technology is the foundation upon which the framework defined on this thesis is built, and thus it has to be well understood before carrying on with the present work.

The four main elements that shape a communicative image are (Figure 2.13):

- A vector (or raster) image in (A)SVG format.
- A knowledge base that formally underpins the semantics of the image.
- Image annotation: mapping the graphic elements that make up the image and their relationships to their semantic counterparts stored in the KB.
- A dialogue system that allows the user to non-visually query and navigate the image.

We now proceed to outline each of these components in detail in the following sections.

### 2.6.1 ASVG

Communicative Images exploit properties of the Scalable Vector Graphics (SVG) format so that images stored in this format may be self-described in a way that can be accessed by Visually Impaired Persons (VIPs) [KO08a]. SVG is an XML-based language for describing two-dimensional vector and mixed vector/raster graphics. Its content can be styled, and it is scalable to different display resolutions. SVG files can be viewed in a standalone manner, mixed with HTML content, or embedded using XML namespaces within other XML languages [Wor18]. Moreover, SVG has native properties that enhance the accessibility of images saved in this format; namely, the ability to represent text as text strings, to fully scale images without loss of resolution, and a rich support of geometrical primitives [KO08a]. In addition, SVG also offers a number of metadata elements that may be employed to provide alternative, human-readable descriptions of the graphical elements that make up the graphic. We will not delve into these elements here, as they were already described in Section 2.1.7.

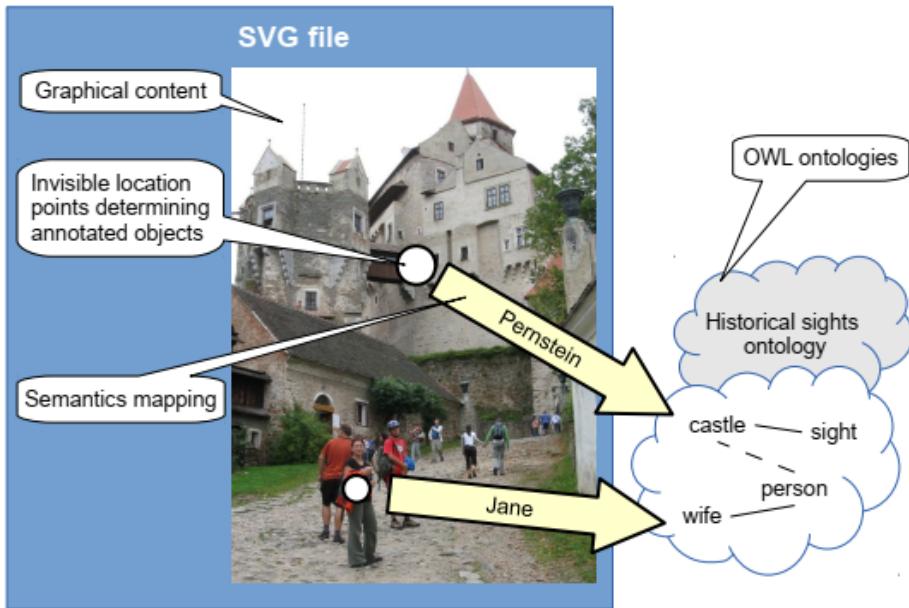


Figure 2.13: Structure of a Communicative Image [Plh14].

The formal interpretation of a Communicative Image is decoupled from the graphical data itself and is attained through an embedded ontology that supports semantic annotation of its graphical elements, their properties, and the relationships between them. In the context of this thesis, graphical annotation refers to the process of assigning meaning to graphical elements through the mapping of one or more graphical objects to individuals of an ontology (c.f. Chapter 4). From this combination of graphical vector information and its associated knowledge base a new graphical format arises, which its authors have named *Annotated SVG* (ASVG) [Ošl09].

## 2.6.2 Graphical Ontology

The *graphical ontology* is the foundation upon which the knowledge contained in Communicative Images is structured and formally characterized. An ontology is defined by Ošlejšek as a “representation of a knowledge base (KB) composed of a domain and its individuals. Domain defines semantic categories, their properties and basic relationships, while individuals represent concrete objects classified within the domain” [Ošl09]. Ontologies and their incorporation to the framework of this thesis will be discussed in greater detail in Chapter 3. This ontology is built from the perspective of Graphic Objects (GOs), their structure and visual properties, and thus it is called the graphical ontology. It offers a number of advantages over other graphical annotation methods, such as support for multilingualism, ambiguity prevention, and consistency among annotated graphics [Ošl09].

The graphical ontology is thus an upper ontology (i.e. top-level ontology) that characterizes

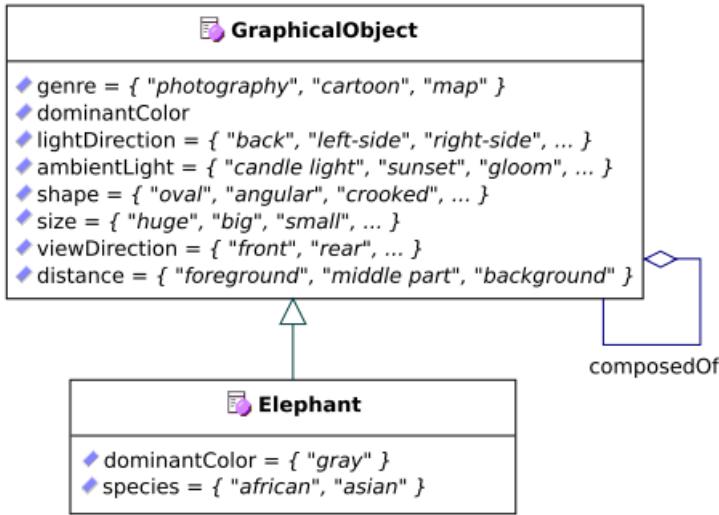


Figure 2.14: Graphical Ontology super-class, its associated properties, and example of a graphic object instance [Ošl09].

the foundational concepts of the graphic domain. This domain describes semantic concepts belonging to real-world visible objects sharing a number of defining characteristics. (e.g. domain classes such as animals, their properties such as size, and relationships between them such as relative position). The graphical ontology has a single top-level semantic category, **GraphicalObject**, underpinning any kind of graphic object and to which universal visual characteristics may be attached in the form of ontological properties. Some of these properties include the dominant color of the object, the main direction of its lightning, its shape, size, and so forth (Figure 2.14). An interesting addition to the graphical ontology is the description of graphical objects' exact color by supporting the so-called Semantic Color Model (SCM), described in detail in [KO08b], that maps any color described in the traditional RGB (Red-Green-Blue) model into a mixture of two named colors among a discrete set of extended primary colors that may be spoken out to the user e.g. "a red-green shade". The graphical ontology supports the SCM by pre-defining several primary colors as named literals (e.g. naming a given red-green SCM color as "orange"). More importantly, the graphical ontology has been designed to act as the support for exploration of a photograph by defining the absolute and relative positioning of the graphic objects (e.g. "in the top-left corner", "in front of") [Plh14].

At this point it is important to note that this formal abstraction of the graphical domain offered by the graphical ontology covers only visually perceptible aspects of graphical objects, and thus it is limited to describing explicit visual concepts that may be of little relevance for the non-visual description of graphic domains other than photography, more so when taking into account the target user group of this thesis, namely Visually Impaired Persons (VIPs). Therefore, it does not suffice as a standalone means of supporting the natural-language

based exploration of graphics. In order to enhance the knowledge base with further semantic information of interest to the user at different conceptual levels, a hierarchy of bottom-level ontologies built atop a semantic foundation (the graphical ontology outlined here) is employed.

### 2.6.3 Bottom-level Ontologies

The formal representation of concepts which are too specific to be included in the graphical ontology is specified by a number of ontologies having their elements conceptually ranked under and supported by the broader concepts of the upper ontology. It may also include elements not necessarily depicted in the graphic [Plh14]. These bottom-level semantic bases are grouped into three main categories in Plhák's thesis: *domain ontologies*, *interest ontologies*, and *user ontologies*.

A *domain ontology* supports the vocabulary and knowledge facts for a concrete domain of knowledge, including non-visual relevant information that could not be included at the top level of the semantic hierarchy. Two specific domain ontologies are introduced as proof of concept for Communicative Images: a *family* model underpinning family relationships, and a *sights* domain covering places of touristic interest [Plh14]. See Figure 2.13 for a simple example of some classes and instances from these domain ontologies. *Interest ontologies* support a dialogue system by enabling the use of structured information about a user's specific interests. They expand a domain ontology by including a dynamic (i.e. user-modifiable) scale system that rates the interest level (*strong*, *weak*, or *no interest*) the user has in some of the ontological elements present in a knowledge base. The dialogue system's output may then be altered to include information of most relevance to the user while hiding uninteresting knowledge items. Finally, a *user ontology* is considered as well. It allows for customization of ontological elements for a certain user so that expressions such as "my wife" in a family ontology can be correctly interpreted by the dialogue system. We have been however not able to find the actual specification of these bottom-level ontologies or any in-depth description of their elements or how they are integrated within the GATE framework.

### 2.6.4 Image Annotation

*Image annotation* refers to the process of mapping of graphical elements, initially void of formal meaning, to ontological elements underpinning their semantics for a given context. Moreover, annotation can define additional information prescribed by the ontology, such as relationships between categories [KPB<sup>+</sup>09]. The original content in SVG images is minimally modified by assigning unique identifiers to its graphical elements. Interesting areas of the image or graphic objects not corresponding to individual SVG elements may also be mapped to semantic features by marking them beforehand through invisible SVG geometries (e.g. points or rectangles) with unique identifiers [KPB<sup>+</sup>09], see Figure 2.13. These unique identifiers are then used in ontological resources when referring to specific graphical elements or areas of the image (c.f. Figure 2.16).

Manually annotating Communicative Images by a human annotator is noted as being too time-consuming to be practical. Therefore, it is suggested for the annotation process to

## 2. STATE OF THE ART

---

ideally be performed by fully automatic methods. However, these methods do not currently provide usable results and present important drawbacks that render them impractical [Plh14]. Instead, an ontology-driven annotation methodology is recommended. Ontology-driven image annotation is a special case of semi-automatic annotation that allows the user to efficiently perform semantic annotation related to the knowledge described by graphical and domain ontologies [Plh14]. This method requires choosing an object in the picture, selecting its semantic category from the ontology, and then assigning values to the prescribed properties. It allows for a significant acceleration of the annotation process. Moreover, the annotation process itself may also be dialogue-based [Plh14].

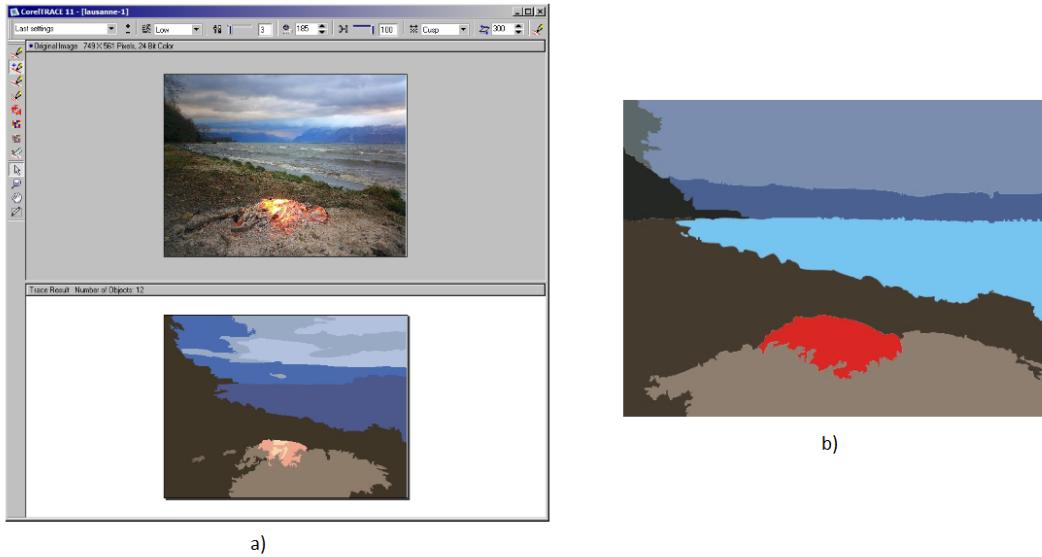


Figure 2.15: Image Segmentation in Communicative Images. a) Automatic image segmentation via tracing, b) Final image segmentation after manual enhancing [KPB<sup>+</sup>09].

The annotation process may be sped up by pre-processing the graphic via image analysis methods that outline objects or regions to be semantically augmented. For instance, Corel Trace's outline trace method is employed in [KPB<sup>+</sup>09] for semi-automatic image segmentation. Image tracing parameters (noise filter, trace complexity, maximum number of colors, node reduction, node type, and minimum object size) can be adjusted during the manual image segmentation pre-processing phase (Figure 2.15) in order to obtain a suitable segmentation of the image into regions to be semantically augmented. There seems however to be no evidence that any actual automatic object detection techniques or ontology-driven methods are employed anywhere within the annotation process in the GATE framework, and images are likely to be manually annotated by authoring the graphical content and handcrafting the supporting knowledge base independently.

A important aspect to remark is that ontology and annotation data are localized outside of the graphical content itself but still within the SVG file inside a separate <metadata>

```

...
<g id="head">
  <ellipse ...head geometry definition... />
  <g id="lefteye">
    <ellipse ...eye contour definition without specific id... />
    <ellipse id="leftpupil" ...pupil definition... />
  </g>
  ... scene graph definition continues here ...
</g>

<metadata id="ANNOTATION_METADATA">
<rdf:RDF>
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://owl.com/ontology.owl"/>
  </owl:Ontology>

  <Head rdf:ID="head"/>
  <Eye rdf:ID="lefteye"/>
  <Pupil rdf:ID="leftpupil"/>
  ... classification continues here ...
</rdf:RDF>
</metadata>
...

```

Figure 2.16: RDF ontology embedded within an SVG file [KO08a].

section therein (see Figure 2.16). In this way, annotated pictures may be handled as well by applications that do not understand the extra semantic data [Ošl09].

### 2.6.5 Dialogue Management

Once an image has been given formal semantics by the embedded ontology its exploration is made feasible through dialogue with the implementation of a *dialogue management* module. This module is a GATE component (named the *DLG* module) independent from the Communicative Images themselves, which are made up of self-contained information describing graphical information. On the other hand, the *DLG* module is responsible for parsing and understanding user questions in natural language and composing reasonable answers with information derived from a communicative image [Plh14]. Moreover, it allows the user to extend the knowledge in the ontology by inputting new information through natural language input queries. The *DLG* module is designed to be completely accessible through a Web interface meeting current standards of Web accessibility.

It is outlined that the dialogue management is fully supported by the Graphical Ontology, previously described. This fact is not completely disclosed, but seems to be exploited by a few specific ontology resources that have to be present in the graphical ontology and which allow for two simple navigational tasks (“communication types” [KO08a]) to be undertaken by the *DLG* module. The first communication type supports a basic controlled natural language named What-Where Language (WWL) [Plh14, KO08a]. This subset of natural language allows for simple questions starting with “what” or “where” to be formulated, and allows to



a)

*User:* Describe the picture.  
*System:* This photo presents the painting called "The Last Supper", which portrays the reaction given by each apostle when Jesus said one of them would betray him.

*User:* Who painted the picture?  
*System:* This painting was created by Leonardo da Vinci.

*User:* What people are in the picture?  
*System:* Sorry, I don't understand your query. Please, type another query.

*User:* Who is in the picture?  
*System:* In the picture, there are Andrew, Philip, Peter, Bartholomew, James Major, Judas, Thomas, Thaddeus, Simon, Matthew, James Minor, John and Jesus.

*User:* Where is Judas?  
*System:* Judas is in the middle.

b)

Figure 2.17: a) A communicative Image b) Dialogue between a user and the image in information retrieval mode [Plh14].

inquire in an instinctive manner about the graphical objects in a scene and their position, such as "what is in the top-left corner?".

The second supported navigational task in Communicative Images is the so-called Recursive Navigation Grid (RNG), based on early work by Kamel and Landay [KL02]. RNG is said to be the orientation backbone of the GATE system [KO08a], enabling the user to obtain information about a specific area of the image with demanded precision and to recursively zoom with desired granularity to subareas of the current one. RNG divides the graphic into nine identical rectangular sectors analogously to the layout of numerical keys 1-9 on the numerical keyboard to which the user is able to refer to during navigation [KO08a]. However, this approach seems to have been only implemented on a related project, a dialogue-based application enabling users to create simple greeting cards [Plh14], and not to be actually present within the GATE framework, with WWL being the only available communication type supported by its *DLG* module.

The *DLG* modules manages user input based on the classic frame pattern approach for natural language processing [BKK<sup>+</sup>77]. A pattern (frame) set for semantically-related questions that may be asked with regards to the ontology is automatically generated. When processing a user's query, the dialogue manager matches it to one of its stored patterns and generates a response and the grammar rules for the user's entry. A dialogue strategy may be employed so the dialogue manager can enquiry for missing information and adapt to the user, including taking into account the communication history [Plh14]. The dialogue module takes into consideration ontological resources, such as instances and properties (e.g. 'is-a' or 'part-of' relations), during the pattern generation step. Patterns are then mapped to grammar rules with respect to its slot types, which identify typical question constructions and should be derived from the Internet or some other text corpora [Plh14]. For instance, Figure 2.18 shows four example patterns generated from a picture containing an instance of the Castle class (a sub-class of Building) of the domain ontology that underpins it. Each of these patterns may correspond to several prompts in natural language.



```

<pattern name="buildingBuilt">
  <date type="date" restriction="moreThanYear" />
  <castle type="building" />
</pattern>

<pattern name="buildingBuiltBy">
  <founder type="person" />
  <castle type="building" />
</pattern>

<pattern name="buildingOwnerByDate">
  <owner type="person" />
  <castle type="building" />
  <date type="date" />
</pattern>

<pattern name="buildingLocation">
  <location type="geoPosition" />
  <castle type="building" />
</pattern>

...

```

Figure 2.18: Generated grammar patterns from a Communicative Image [Plh14].

The dialogue engine parses a user query based on grammars formalized by the Speech Recognition Grammar Specification [Wor04b] and then resolves the parsed query to a given pattern with its slots filled with specific items from the query. Moreover, those words in a user's query that are mapped to certain slots may be expected to belong to a specific category during the initial query's processing phase. For example, in the template "How far is it from SLOT1 to SLOT2?", both slots are expected to be filled in with specific entries from the objects ontology resource representing graphic object instances. The system's output to an enquire about an certain object is determined in two different ways [Plh14]:

- *Information retrieval*: all available information about the object is fetched from the

annotated image, related domain ontologies, and external sources and presented to the user.

- *Information filtering*: an intersection set between the interest ontology and domain ontologies is computed. Each identified resource from the information retrieval phase is then given an appropriate interest weight if it is found within this intersection set, or an interest weight of zero otherwise. The final weight of each resource is then computed by a linear combination of its interest weight and the amount of related supplementary data (number of resources related to the object) in the ontologies. They are then sorted according to their final weights and the first  $k$  resources are provided to the user, with  $k \in N$  representing how ‘talkative’ the system is.

Three communication modes are available to users, who can switch between them as they so desire. On the *information retrieval* mode, users ask questions and get answers. The system is also allowed to ask questions to the users in a turn-taking fashion if the user chooses the *free communication* mode. This mode includes a basic mechanism for dealing with contradictory information from the user. A voting system is employed so that when a considerable amount of information pieces contradict each other, the user is prompted to add further information in order to clarify the misunderstanding. We could however not find any further information on how this voting system was designed or implemented. Lastly, the *image information supplementing* mode allows users to supply the image with missing information pieces. See Figure 2.17 for an example of a real dialogue between a user and a communicative image.

### 2.6.6 Communicative Images for Accessibility

Surprisingly, Communicative Images have not been evaluated with blind or visually impaired users [Plh14] even though making graphics accessible for blind persons was noted as one of their main long-term goals [KO08a]. Some of their potential benefits for increasing accessibility to online graphics are noted, such as aiding blind persons with online shopping by offering on-demand information about the visual characteristics of the offered items. The idea of communicative maps is also introduced [Plh14]. They are Communicative Images that allow users with reduced mobility to ask questions about navigational routes to their destination or about nearby points of interest. Enhanced online social communication and access to educational graphical materials are some further potential use cases underlined.

Actual evaluation of Communicate Images was undertaken by an unspecified number of sighted university students (whose demographic profile is also unknown) on a single image, namely the renowned painting “*Il Cenacolo*” (“The Last Supper”) by Italian Renaissance painter Leonardo da Vinci (see Figure 2.17, a) [Plh14]. The graphic objects in the picture, such as the central figure of Jesus, the Twelve Apostles, the table they gather around, background, etc. and some of their visual attributes, such as color, facial expressions, etc. were manually annotated i.e. linked to ontology resources of a supporting graphical ontology, as previously described. Moreover, details about the painting itself, such as author and completion date, were also included. As the participants were sighted, the picture was hidden from their sight during evaluation.

Most participants reported a very positive subjective impression of the communication process with the image, describing it as “funny”, “easy”, and “natural”. All participants were reported to have been able to build a sufficient mental model of the graphic and imagine the scene. Conversely, not all users agreed on the effectiveness of the approach, with feedback ranging from “very effective” to “not very effective”. This may be due to the simplicity of the used controlled natural language, WWL, which we believe was likely to frustrate users. Some of the most criticized aspects of the system by the participants were actually related to its lack of “support of general questions” beyond the scope of What-Where Language (WWL). It is also important to note that many participants were likely to be already familiar with this very well-known particular piece of art, which might have biased the results of the evaluation.

The evaluation of Communicative Images, albeit lacking in many aspects, shows that users can interact in a satisfactory manner with images using a purely natural language-based approach. Nevertheless, in their current state they present a number of critical drawbacks that prevent their take-up by VIPs, namely:

- Lack of support for graphical domains other than photography. As previously described, the developed graphical and domain ontologies cover visual aspects of real-world graphical objects, such as position, size, color, lightning conditions, etc. On the other hand, the problem of non-visually accessing diagrams fundamentally different to that of accessing photographic content, as different inference mechanisms are at play, and, in addition, the cognitive benefits stemming from visualization need to be somehow compensated by the dialogue system. Diagrams are not considered at any point of Communicative Image’s design, leaving behind a very sizeable subset of graphics which most people come across in their daily lives.
- Cumbersome annotation process. Communicative Images are given formal semantics by manually mapping their graphical objects to resources of a supporting ontology that needs to be authored separately from the image content. This segregated process is tedious, error-prone, and time-consuming, which hinders their adoption by authors of accessible graphics.
- Very limited natural language understanding capabilities. Users were allowed to communicate with the image by means of WWL, a controlled natural language whose grammar does not cover the majority of questions a user may inquire about an image. Moreover, no hints or suggestions are given to the user when the system does not immediately recognize the objects present in the user query. These drawbacks lead to a cumbersome, frustrating, and error-prone dialogue that does not meet most user’s expectations, polluting the reliability of the evaluation results.
- Evaluation by sighted users only. When evaluating computer systems for accessibility it is paramount that the end user groups e.g. blind users are actively involved in the evaluation process [FdCF12]. Communicative Images have not been evaluated by blind or visually impaired users, and thus its results must not be extrapolated to their specific needs and requirements.

In conclusion, pure Natural Language Interfaces (NLIs) to graphics are very under-researched even though they possess very high potential in enabling VIPs access general graphics in an effective, efficient, and inexpensive manner. The case of NLIs to diagrammatic representations is even more deficient, since we could not find a single approach tackling this problem during the Problem Investigation phase described in the present chapter. With this judgment in mind, we proceed to the Treatment Design phase of the Design Science methodology, described in Section 1.5. The following chapters tackle the design problem of how to adapt and expand the framework of Communicative Images in order to obtain meaningful evaluation results that could answer the research questions posed in section 1.4. Lastly, next section will discuss accessibility of Web graphics, since, for reasons described in Section 5.5, the NLIs to diagrams prototype that we have named AUDiaL, which is the main artifact designed in the context of this thesis, is a Web application.

## 2.7 Web Content Accessibility

AT and methods for accessing graphics on the WWW deserve a special mention in this chapter even if these applications could have also been included in other sections of this chapter. Traditional non-visual web browsing relies upon the use of audio to provide a representation of page contents [KYM07], most commonly through the use of screen readers. In principle, the Web has been designed to be an environment for everybody, regardless of age, used device to access it, or whether the user has a disability; a goal that is referred to as "common accessibility" [AW08]. However, images are used extensively on websites and can create major barriers to general Web browsing when they are not accessible [HKRE14]. Lack of accessible alternatives to graphics is one of the main causes of frustration for screen reader users on the Web [LAKM07]. For instance, blind users tend to skip non-accessible images or try to infer their meaning from the surrounding text [BCB<sup>+</sup>07]. They may also make use of several coping behaviors for dealing with inaccessible graphics, such as impulsive clicking on poorly labelled pictures or query reformulation when results given in graphical form are not accessible. Often, users have to surrender their task altogether due to the inaccessibility of graphical content [VH13].

Three main interrelated sides need to be taken into account when assessing Web accessibility, as seen in Figure 2.19 [Hen05]:

- *Content*: the information in a Web page or Web application, including both the natural information such as text and images; and the code or markup that defines the structure, presentation, etc. of the Web document.
- *Users*: the persons accessing the content and the user agents they use to reach and interact with it, such as Web browsers, media players and assistive technologies.
- *Developers*: the persons creating Web content, and the authoring and evaluation tools they employ in the authoring process.

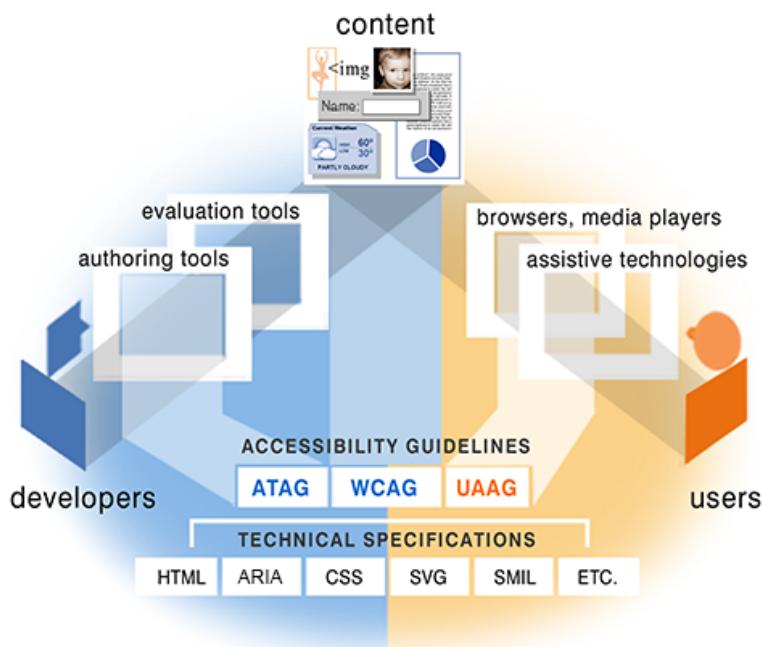


Figure 2.19: Components of Web accessibility<sup>11</sup>

### 2.7.1 Web Accessibility Standards

Successful implementation and proliferation of accessibility features largely depends on the outcome of a number of ongoing standards and best-practices discussions. Regarding Web accessibility standards, the most influential group at work today is the HTML5 Working Group of the World Wide Web Consortium (W3C) [DIA16], and, more specifically, its Web Accessibility Initiative (WAI). While government bodies and usability experts have also developed their own set of Web accessibility guidelines, the WAI guidelines remain the authoritative worldwide reference on the matter [HR13].

Three sets of guidelines on Web accessibility practice have been developed by the WAI, one for each of the previously listed components of Web accessibility. These guidelines are designed to aid a broad population of disabled users, including those with visual impairments, hearing loss, physical disabilities, and older adults. Here we will concentrate on those aspects most relevant to Web users who are visually impaired.

- *Web Content Accessibility Guidelines (WCAG)*: intended to be used by software developers to create accessible Web content. They provide guidance covering a wide range of success criteria and techniques that, used in combination, make websites more accessible. WCAG 2.1 [Kir18] was published in June 2018; its predecessor (WCAG 2.0),

---

<sup>11</sup>Image by Michael Duffy, from: Essential Components of Web Accessibility. S.L. Henry, ed. Copyright W3C© (MIT, ERCIM, Keio, Beihang). [www.w3.org/WAI/intro/components.php](http://www.w3.org/WAI/intro/components.php).

## 2. STATE OF THE ART

---

published in 2008, became an ISO standard in October 2012. Moreover, further recommendations for increasing the accessibility of advanced Web applications are included in the W3C's WAI-ARIA technical specification. In particular, these recommendations refer to dynamic Web content and advanced user interface controls developed using Ajax, JavaScript and related technologies [Coo06].

- *User Agent Accessibility Guidelines (UAAG)*: these guidelines address how Web browsers, media players, and other software and devices that render Web content (i.e. user agents) can be made accessible for users with disabilities. For example, they specify that all functionality must be keyboard operable or how user agents should facilitate access to assistive technology such as screen readers. UAAG 1.0 has been a World Wide Web Consortium (W3C) recommendation since December 2002, while the first public draft of UAAG 2.0 was published on March 2008 and has not reached the recommendation status yet [All08].
- *Authoring Tool Accessibility Guidelines (ATAG)*: these documents explain how software and services that authors (e.g. Web developers and designers) use to produce Web content can be made accessible so people with disabilities can create Web content. They also specify how authoring tools can help authors create more accessible web content i.e. content that conforms to the Web Content Accessibility Guidelines (WCAG) [Ric15].

### 2.7.2 Graphics on the Web

The rest of this section will focus on the parts of the Web Accessibility Initiative (WAI) guidelines concerning graphics, noting their drawbacks and some alternatives found in the literature. It is however important to note that while conforming to the WAI guidelines is a very important step towards assessing the accessibility of a Web site, they can by no means ensure that the accessibility needs for all user groups are properly met. Some authors have pointed out that these guidelines merely ensure the “technical readability” of Web sites, instead of accessibility itself [DPS04].

Graphics embedded in Web documents are part of their content; therefore, the focus in this section will be on those best-practice recommendations laid out by the WCAG 2.1 on ensuring the accessibility of images. Success criteria defined by the WCAG on its guideline 1.1. (“Text Alternatives”) simply specify that all non-text content present on a Web site must have “a text alternative that serves the equivalent purpose” [Kir18]. A few exceptions are noted, though. The ones relevant to static graphics are:

- Images that behave as controls or accept user input. In this case, a name that describes its purpose instead of alternative text suffices.
- Test content that would be invalid if presented in text.
- Non-visual alternatives to graphics used to distinguish humans from computers trying to access a Web site (CAPTCHA) should be included.

- Decorative images and those used only for formatting purposes should be implemented in a way that can be ignored by assistive technology, generally by providing a ‘null’ text alternative (i.e. an empty string as alternative text).

Text alternatives can be generally implemented in HTML by adding them to the `alt` attribute of the `img` element that defines the graphic (Figure 2.20). Screen readers may then fetch the value of the attribute and display it to the user in lieu of the graphic. This means that tools and methods that assess the quality of accessibility on a Web page e.g. [AW08] would consider most graphics containing an alternative textual description to be accessible, at least to the minimum required level. While this may be a sufficient approach for conveying the meaning of a simple graphic or a basic overview of a more complex one, unfortunately it falls short for non-visually representing more informationally dense graphics.



Figure 2.20: The main logotype at `www.jku.at` (left) and its constituent HTML elements, including a text alternative (right)

This limitation has indeed been observed by the W3C [Hak12], which has proposed a two-part alternative to representing “complex images” such as statistical charts and maps [Egg14]. The first part is a short description, specified in the same way we have just seen, whose purpose is just to identify the image. The second part consists in a longer textual description, which may be placed on a different section of the current document or on a different Web page. The `longdesc` HTML attribute can be used to point assistive technology to the page containing the alternative long description. For technologies supporting HTML5, the `figure` element may contain a `figcaption` sub-element wrapping the alternative long description of the image. This alternative simply allows for longer textual descriptions to be displayed to the user, which as we have already discussed are not a sufficient means for representing complex graphics. In a similar vein, the W3C also initiated the Accessible Infographics Community Group [Wor12] with the goal to “make information graphics, like bar charts and maps, accessible as possible to all” by devising and proposing additions to Scalable Vector Graphics (SVG) that improve accessible options for graphical data. However, this community has not been active in the last years, and so far they have not published any reports, guidelines or new technologies.

### 2.7.3 Advanced Approaches

Alternatives to graphics on the Web based on anything other than static text are currently not contemplated by any commonly used guideline; neither does HTML nor other technologies

## 2. STATE OF THE ART

---

for creating Web content support them natively. As we have seen, static textual descriptions do not suffice to offer a proper alternative to graphical content except for the most basic of images, regardless of their length and complexity. A potential temporary solution to this issue has been investigated by Kopeček and Ošlejšek on their primer on communicative images [KO11]. They suggest that, in order for them to be user-friendly and easily accessible, alternative versions of a graphic should be fed into traditional Web browsers via a third-party plug-in that needs to be installed in the user's Web browser. This would allow various types of interaction according to the device and the user's preferences stored in the browser. A double clicking action on an image performed by the user would start a connection to a remote server that would then parse the image and return an accessible alternative, dialogue-based in their case. However, not too many details are provided regarding the architecture and adequate technologies for such a plug-in driven conversion to be feasible, and this approach remains a broad idea yet to be undertaken. Communicative images were discussed at length in Section 2.6. A working similar approach is presented by Elzer et al. [ESC<sup>+</sup>07], in which a browser extension is also used to generate textual summaries of bar charts. Their Internet Explorer extension can be launched via a keyboard shortcut any time the user finds a bar chart while navigating a Web site with the assistance of JAWS screen reader. The tool first runs a brief image processing algorithm to determine whether the graphic is a bar chart. Then, a summary of the chart is computed, appearing immediately afterwards on a new window for the user to read. Further information can be read in Section 6.6, which discusses automatic summarizing techniques for graphical representations.

Tactile and hybrid-based approaches to web navigation have been considered as well. Rotard et al. [RKE05] explored a tactile Web browser that renders both text and graphics contained within HTML Web documents, fully retaining their two-dimensional structural information. It automatically escalates Web content to fit within the small resolution of tactile graphic displays. Text and graphics can be selected individually by the user, who can then explore them haptically. Graphics are converted into a monochrome equivalent and displayed in the tactile display, whereas text can be read either in Braille or via speech synthesis. In some cases, graphics can be explored in an incremental way by hiding/revealing new shapes after the user has finished exploring the current ones.

Yu et al. [YRB02] suggest a multimodal method for accessibility of graphics on the Web in which the graphical information is conveyed through haptic and audio channels. In their approach, line graphs, bar charts, and pie charts are accessible through a force-feedback device. Pre-recorded sound files as well are used to represent graph contents to users. A plug-in, installed on the user's computer, allows the force effect to be activated and to communicate with the source code of a Web page. The features of Web graphs are constructed by integrating appropriate primitive haptic objects supported by the plug-in which then provides the force effects to a force feedback mouse. For example, a rectangular enclosure is used to model the bars in a bar chart. The coordinates of each bar on the graph are measured and subsequently used to define the location and size of the effects. These primitive objects are the constituent elements of the accessible diagrams, which are surrounded by a virtual enclosure which constrains the user's movements in a pre-defined area. Moreover, audio

notes are played e.g. when the mouse cursor moves over a bar to guide the user along the chart. Many other tactile and hybrid approaches to Web accessibility have been undertaken e.g. [KYM07, RPAP00, AYB11, KTM09, YKM<sup>+</sup>06]. However, graphical content is ignored by all of them, and only their alt-text attribute is generally presented to the user in Braille or speech output.

Besides tactile and sonification approaches, dialogue has also been briefly explored as a non-visual means of accessing Web sites. Di Blas et al. present an approach they name WED (Web as Dialogue) based on the assumption that “the interaction between a human being and a Website can be interpreted in terms of a dialogue” [DPS04]. In this special kind of dialogue, the Web site’s conversational turn consists in offers of content or interaction, whereas the user’s turns consist in the selection of an offer, usually by clicking a hyperlink. However they note that this kind of ‘dialogue’ is syntactic rather than semantic, in that the user is not allowed to perform semantic queries e.g. “take me to context X” and has to resort to phoric elements e.g. “take me one step back”. Subsequently they raise the following question: “How should we design a Web application in order to generate successful dialogues with its users?”; a question they aim to answer by observing traditional human-human dialogue strategies and reinterpreting them in Web design terms. They then present a prototype of a Web site supporting an exhibition of prints that took place in a museum that can be navigated via dialogue. Even though this concept only takes into account simple navigational tasks and does not delve into reasoning or semantics of a Web site’s content, it shows how dialogue may be an efficient means of non-visually accessing structured visual information, such as HTML documents or SVG images.

## 2.8 Summary

This chapter has provided an exhaustive literature overview of traditional and state-of-the-art methods and technologies for non-visual access to diagrammatic representations. This process has been motivated by the Design Problem of finding an effective means for blind persons to access diagrams autonomously (section 1.2). Following the Design Science methodology (section 1.13), this chapter corresponds to the Problem Investigation task. Existing approaches have been categorized according to the modality of their interaction with end users, namely, Visually Impaired Persons (VIPs). Five main categories were identified. Each approach was assigned to a category and analyzed. In short:

- *Linear approaches* (Section 2.1) offer a verbal or tabular summary of the diagram, which may then be accessed via text-to-speech (TTS) technologies or through touch, most commonly in the Braille writing system (Section 2.1.2). They are easy and inexpensive to produce and access, and are thus the most common approach to non-visual access to graphics. Linear descriptions of graphics are, however, far from being functionally equivalent to their visual counterparts, and they easily overload the working memory of the reader. Therefore, interactive methods are preferred in order to enable users to form a better mental model of the graphic.

## 2. STATE OF THE ART

---

- *Haptic approaches* (Section 2.2) are adapted representations of the original graphic that are read by means of touch. They are generally printed in raised dots and lines in swell-paper, or displayed by specific devices such as tactile shape arrays. Many of the visual attributes of the original graphic, such as shape, size, or position, can be carried over to its tactile counterpart. However, current limitations in available display size and resolution result in compulsory simplification of the original diagram to the point where much of its information gets lost. Moreover, costly and difficult to use hardware needs to be employed. Due to these and other drawbacks, most blind persons do not even attempt to read a tactile diagram.
- *Sonification* (Section 2.3) refers to the representation of data through non-speech sounds, such as musical notes. Sonification approaches are most commonly employed to non-visually display line charts, as sound properties such as pitch can be mapped to the current value (along the y-axis) of the represented data at the user position on the x-axis. Designing a satisfactory mapping is however difficult to achieve, and concrete numerical values cannot be retrieved. Moreover, many kinds of statistical charts, such as scatterplots, cannot be accurately represented by this means.
- *Speech-supported approaches* (Section 2.5 and Section 2.6) provide information about the graphic in natural language to the user. Well-designed speech interfaces enable effective, efficient and familiar human-computer interactions, since language can naturally express information at all levels of abstraction. Moreover, they require no specific hardware or software, and can be accessed by blind users in a familiar manner e.g. through a traditional Web-based interface (Section 2.7). Most current speech-enabled methods however force users to employ a number of predetermined input commands to navigate the graphic, which require previous user training and do not leverage the full capabilities of Natural Language Interfaces (NLIs). The sole two exceptions are Visual Question Answering (VQA) systems (Section 2.5.3) and Communicative Images (Section 2.6). VQA systems are still in their infancy and have not been designed with accessibility in mind. Communicative Images have only been briefly evaluated with sighted participants exploring photographic content, and diagrams have been left out of their design process.
- *Hybrid approaches* (Section 2.4) combine two or more of the previously listed modalities. They allow for the disadvantages of one approach to be compensated with alternative means from another approach. As most hybrid approaches employ tactile output in combination with sound or speech, they require expensive specific hardware, user training, and support from a sighted person, which greatly hinders their take-up and availability.

It can therefore be concluded that the studied approaches do not meet the requirements of VIPs accessing diagrams, as they either provide a very low output resolution, overload the user's working memory, or specialized software and/or costly hardware have to be employed. Moreover, lack of individual autonomy for the user along with a steep learning curve hinder

their widespread take-up by the visually impaired population. Natural Language Interfaces (NLIs), on the other hand, are intuitive to use and do not require of any specific software or hardware, as they can be accessed online the same way blind users are used to interact with dynamic Web content, thus having the potential of being an adequate alternative to existing approaches to accessing diagrams for blind persons.

Currently, NLIs are being commercially employed in a widespread spectrum of applications. However, despite their increasing popularity, research on this kind of technology is rarely motivated by the goal of accessibility [FS09]. Particularly, dialogue-based approaches to accessing diagrams with a focus on accessibility are currently missing in the Human-Computer Interaction (HCI) literature. Therefore, this thesis aims to study whether natural language is a suitable means for non-visual autonomous access to diagrammatic representations by designing and validating a NLI to semantically-enhanced diagrams. The following chapters discuss the design and validation processes of such resolution.



# CHAPTER

# 3

## Formalizing Visualization Semantics

The first task in the design of an efficient Natural Language Interface (NLI) to diagrams consists in defining a suitable data model underpinning the necessary aspects of information visualization that will be demanded by its users. As the graphic at hand needs not be visible, the most important aspect of said data model is its proper actionability by the dialogue subsystem while retaining as much information as possible from the original graphic. We thus aim to find a suitable formal representation of diagrams that allows for efficient interaction with the NLI, including retrieving information of interest for the user (such as performing wayfinding activities or analytical tasks on the diagram's data) as well as inputting additional bits of information, both user- and system-initiated. This resolution devolved into a new design problem to be tackled, as depicted in Figure 1.13. Namely, the study of *visualization semantics* and how to efficiently model them for their consumption by an accessible NLI. Such undertaking is described in the present chapter.

### 3.1 Motivation and Problem Investigation

Diagrams are most commonly authored using software that produces raster images, such as GIF or JPG files, as output [DFVR10]. However, it soon becomes apparent that these unstructured formats present many critical limitations that obstruct their operability by Assistive Technology (AT) for Visually Impaired Persons (VIPs). Some of these include:

1. The inability to obtain precise information about the data depicted in the diagram, such as the exact value of a given bar in a bar chart.
2. Lack of support for reusing common visualization terms allowing for interoperability between software clients.

3. Adding additional metadata, user-generated or otherwise, is usually difficult and cannot be seamlessly integrated with the graphic itself.
4. Mapping raster images to other formats or representations is a very challenging task, since they are made up of unstructured raw data that requires heavy pre-processing.
5. Unavailability of searching for diagrams meeting specific user-defined criteria within the growing corpora of diagrams on the World Wide Web (WWW).
6. The difficulty to share and merge diagrams, which is often a necessary step in order to perform common analytical tasks supported by more than one diagram.

On the other hand, structuring and providing formal semantics to raster graphics enables their translation into different modalities, such as sound, speech, or haptic alternatives [DFVR10], thereby enabling the automatic or semi-automatic generation of accessible counterparts. This idea was already illustrated, for instance, by Communicative Images (discussed in Section 2.6 of previous chapter). Therefore, we will start this section by analyzing current approaches to the formalization of diagrams which allow their complex processing by software clients.

#### 3.1.1 Common Diagram Encoding Formats

Most graphing software applications that enable users to author diagrams directly stemming from in-app data tables, such as Microsoft Excel and Apple Numbers, make use of their own proprietary Object Model (OM) file formats to encode and store graphics. These formats are independent from the structures used to persist the graphic's raw tabular data, which is generally given by files in Comma-Separated Values (CSV) format. The greatest advantage of these formats is their widespread prevalence, since these tools are very commonly used in business, science, and education to create and divulge diagrams. There also exist open formats, most commonly based on Extensible Markup Language (XML), which have been developed to enable interoperability between software tools working with diagrams. Some popular examples include GXL (Graph eXchange Language), which represents “typed, attributed, directed, ordered graphs” [HSSW06]; GraphML (Graph Markup Language) [BELP13], also underpinning link diagrams; and CML (Chemical Markup Language) [MRR11], the *de facto* language employed to formalize chemical knowledge. Other application-specific XML-based graphic syntax languages include those used by Gao on their thesis [Gao13] and Ferres et al. on iGraph-LITE [FLST13]. Besides these modeling languages, there also exist environments for popular programming languages used in statistical computing that allow programmers to author diagrams of great complexity by following specific diagram-creation grammars, also known as *command languages*. Some advanced environments of this kind include the `ggplot2` library for the R programming language [Wic10] and Python's `matplotlib` library [Hun07].

These formats are generally specific to a diagram type, such as link diagrams (GXL, GraphML) or statistical charts (Excel's OM); or domain-specific, like chemical structural formulas (CML); which limits their applicability to a broad set of diagrammatically-represented domains.

Moreover, they do not provide of native means for embedding high-level semantics to the graphic besides their restrictive support for annotations in the form of metadata, already discussed in section 2.1.7. Relational databases may also be used to store diagram encodings. In this case, concepts are stored in tables of an equivalent relational model. Each table is made up of rows, storing instances of the category (entity type) it represents, which are identified by a unique key each; and columns, generally symbolizing values attributed to its row's instance. Instances may then be conceptually linked to other rows by adding relations between them. Relational databases are the most commonly used method for data storage and retrieval in software systems. However, they also do not possess any native capabilities for representing and storing semantic information about the meaning of their concepts and relationships [Plh14]. Moreover, relational database models are not easily extensible, which is a vital characteristic of data models supporting visualization in which many possible description dimensions exist [DFVR10].

### 3.1.2 Incorporating Semantics

The inclusion of structured formalized representations of the high-level concepts depicted in a graphic has been shown to improve semantic image retrieval and automatic image captioning in photographs [KZG<sup>+</sup>17, Plh14, LZLM07] as well as dialogue-based navigation thereof [Plh14]. Furthermore, they allow for automatically disambiguating between various kinds of knowledge that can be represented by a diagram in different ways, thereby enhancing diagram interpretation [CCS01]. Employing such a formal, unambiguous, and universal representation of terms is also useful as an intermediary between natural language and visualization concepts [Tow04]. Some authors mention that “without deeper understanding of the semantics” of a diagram, the knowledge contained therein is “simply not available or may result in using false information during vital decision-making” and thus “there is always the need to add information besides that which comes formally embedded in graphics” [DFVR10]. A shorter, perhaps more elegant, wording of the same idea states: “visual experience is not merely an epiphenomenon of visual function” [Ede09].

Therefore, the shortcomings of traditional graphic information persistence methods outlined in previous section make evident that, if any sort of complex interaction between the user and the graphic such as dialogue (and, by extension, inference on visual information) is desired, data models supporting the encoding of high-level concepts of the graphic along with its visual primitives are required: “a profound capacity for perceptual contemplation goes together with the capacity for seeking out flexible, open-ended mappings from perceptual stimuli to concepts and to actions” [Ede09]. Another way of expressing this need is by affirming that the semantic gap between low-level image features and high-level human knowledge must be formally bridged. Let us now briefly expand on this idea.

#### The “Semantic Gap”

Research in the field of Content-Based Image Retrieval (CBIR) distinguishes between two different kinds of ‘gaps’ between human interpretations of visual content and those derivable by a computer, known respectively as the *sensory gap* and the *semantic gap* [Liu10]. The

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

sensory gap refers to the uncertainty of graphic objects in terms of the state of the object, where the uncertainty may stem from a particular viewpoint, occlusion of the object, scene clutter, illumination, etc. For example, some bars in a 3D bar chart may be partially occluded by other bars placed to their front, which might result in a poor interpretation of their meaning (for example, the numerical value expressed by the bar) by a human or artificial reader.

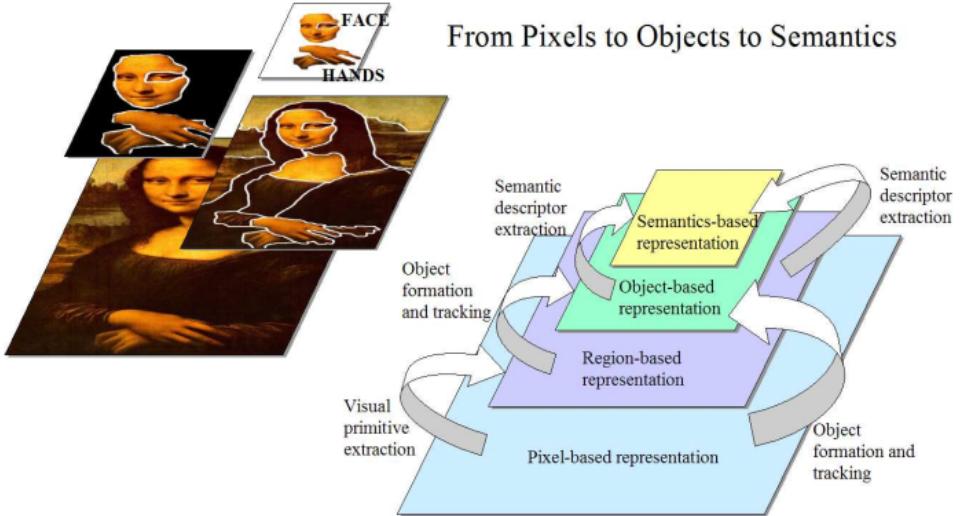


Figure 3.1: Visual Information as a hierarchy of conceptual levels [Tow04].

The semantic gap, on the other hand, deals with the mismatch between the information depicted in the graphic and the interpretation that the same data has for a person in a given context. This is, it expresses the semantic divergence between the simplicity of visual features (e.g. color, shape) and the richness of human semantics [Liu10]. Coming back to the example of a 3D bar chart, the semantic gap is what divides the visually perceptible aspects of a bar such as its height, color, and position within its associated metric graphic space, and its human interpretation (e.g. that 100 units of a given item were sold on a given time period). The same notion pertains to high-level information that can be inferred from relationships between graphic objects in the diagram. For instance, an arrangement of bars of successively decreasing heights may be interpreted as a steady decline in sales for a given item.

#### Hierarchical Semantics

The descriptive dissonance expressed by the *semantic gap* can be conveyed at different levels of abstraction according to specific conceptual levels, as depicted in Figure 3.1. The concept of abstraction (Figure 3.2) is a fundamental aspect of computer science and can be briefly defined as a mechanism that denotes “the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer” [BME<sup>+</sup>07]. On the bottom-most level of

this hierarchy lays the raster pixel representation of the graphic, made up of visual primitives void of any encoded high-level human significance. The vast majority of images found on the WWW are described at this initial level of abstraction only, as it suffices for displaying the graphic on a Web browser.

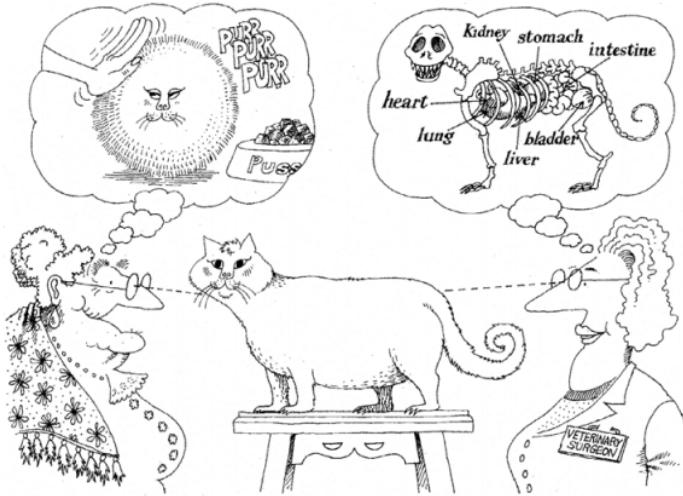


Figure 3.2: Abstraction focuses on the essential characteristics of some object, relative to the perspective of the viewer [BME<sup>+</sup>07].

Each subsequent level of the hierarchy is built atop the concepts described in the previous one, following a bottom-up approach to defining image semantics, and corresponds to an immediately higher level of abstraction oriented towards capturing its semantic interpretation i.e. it conveys a more abstract ‘meaning’ of what is depicted in the image or some particular section thereof [Tow04]. The second level of this semantic hierarchy generally consists of basic feature-based regions of the graphic. For instance, an object’s shape may be identified because of the high difference in contrast between the pixels the object is made of and the pixels that belong to its background. Finally, on the topmost levels of the hierarchy meaning is ascribed to specific objects, parts thereof, or whole scenes.

Deriving domain-specific higher abstractions from a combination of lower-level classifiers while including emergent properties absent from the lower-level abstractions is also the basis of many state-of-the-art approaches to robust object recognition, which are based on the complementary notion of “union of weak classifiers”. This idea has a strong tradition in the field of Gestalt psychology which describes how “image structure is perceived based on principles such as proximity, similarity, continuity, and coherence” [Tow04]. More specifically, the main principles of the Gestalt theory, as outlined by Wagemans et al. [WGK<sup>+</sup>12] (cited in [CF15]), are the following (in very brief form):

- *Holism*: Even if principles such as shape and color are perceived separately by the human visual system, perceptual experiences are intrinsically holistic.

- *Emergency*: emergent properties belong to the whole and not to the individual parts (for example, the density of a forest does not belong to each individual tree).
- *Configuration superiority*: the whole is perceived before the parts. For instance, the forest is perceived before its constituent trees can be discerned.
- *Global precedence*: processing happens from the global structures to the analysis of local properties.
- *Primacy of the whole*: the properties of the whole are born from inter-party relations such as symmetry, regularity, closing, etc. and cannot be directly derived from the properties of the constituent elements.

The notion of an aggregate-composite approach to graphic meaning with higher-level semantic concepts taking precedence over lower-level ones is directly connected with the widely-shared notion of the “recursive nature of graphics” in which graphic object syntactic decomposition can be broken down into recursively finer-grained sub-objects until its elementary graphic constituents are reached. The recursive nature of graphics will be detailed to greater extent in Section 3.5.2, where we discuss how it is supported by the different levels of the suggested conceptual hierarchy.

### 3.1.3 Ontologies for Visualization

From a computational standpoint, some authors translate these ideas by stating that artificial systems dealing with high-level semantic visualization concepts are required to incorporate “versatile intermediate representations” that “(a) make explicit as wide as possible a variety of scene characteristics, and (b) can be linked in a flexible manner to a conceptual system that is capable of growing with need and experience” [Ede09]. Ontologies, as we will discuss with greater detail in the next section, are the preferred method for storing and retrieving this kind of multifaceted semantic information at different levels of abstraction [Plh14], as opposed to the different data modelling methods discussed in Section 3.1.1. Ontologies are moreover suitable to many problems in computer vision that require prior knowledge to be modelled and used [Tow04]. Most importantly in the context of this thesis, ontologies encoded in Resource Description Framework (RDF) have been shown to be an appropriate means for storing semantic metadata about diagrams (at the very least in domains characterized by link diagrams) as a basis for their non-visual exploration by visually impaired persons [Bro08]. This may however not be directly extrapolatable to other diagrams types since, being intrinsically a graph (in its mathematical sense)-based specification, RDF data models may be better suited to representing link diagrams than other diagram types [HD05].

In any case, given the extensive collection of disparate domains that ontologies have been employed to codify, including real-world images and visualization domains, we believe that ontologies are a suitable approach meeting the requirements to formalizing the semantics of diagrams at their various levels of abstraction for their storage, retrieval, and interpretation by a NLI. Some authors go as far as stating that “a collaboratively developed ontology which

formalizes common concepts, relations, and facts from the broad corpus of visualization literature [...] will be a solid foundation for upcoming (semi-)automatic visualization systems” [PV13]. Moreover, ontologies have been satisfactorily used as the means to store data models and their metadata in dialogue-based systems, disability-specific adaptive user interfaces such as [KT06], and other applications for accessibility. Ontology reasoning i.e. automatically deriving new facts from the current ones, also provides the benefit of greatly speeding up the semantic annotation of graphics as well as ensuring its logical consistency, something that will be discussed in Chapter 4. Lastly, ontologies natively enable the standardization of domain terminology in different languages, therefore easily supporting system internationalization [KO11].

Section 3.4 will introduce the methodology employed in this thesis for formalizing visualization semantics through ontologies. But first, Section 3.2 delves into greater detail into what ontologies are, available syntax formats and constructs, and most common application scenarios, in order to lay down the required conceptual framework used throughout the rest of the present chapter.

## 3.2 Ontologies and “Linked Data”

The term *ontology* originally stems from philosophy, where it is used to describe “the existence of beings in the world” [SBF98]. Because many fields in computer science, such as Computer Vision (CV), Natural Language Processing (NLP), and Artificial Intelligence (AI) as a whole, deal with reasoning about models of the world, researchers in these fields have adopted the term ontology to denote “what can be computationally represented of the world in a program” [SBF98]. There exist however clear divergences between what ontologies aim to express in their philosophical and scientific variants. Whereas philosophical ontology is primarily concerned with “an account of being in the abstract” by seeking to establish “universal and largely qualitative accounts of reality”, ontologies prescribed in the natural sciences “demand a more practical quantitative approach that aims to provide explanations and predictions of observable phenomena”, being constrained by a number of “ontological commitments” for each specific branch of science [Tow04]. Ontologies in their scientific meaning have been used for centuries in the natural sciences to formally describe (typically hierarchical) classification schemes together with a set of matching criteria and an underlying theory [Tow04]. Examples of ontologies in the natural sciences include the *Systema Naturae* written by Carolus Linnaeus in the 18<sup>th</sup> Century, the periodic table of elements, and the standard particle models of quantum physics [Tow04]. Ontologies have proven to be of great use in the natural sciences as a means for sharing common vocabularies within scientific communities.

One of the most commonly highlighted use-contexts of ontologies indeed is knowledge sharing between persons and between computer systems [CCS01]. Ontologies enable knowledge sharing between people by facilitating a common agreed formal vocabulary to describe a domain. Knowledge sharing between systems is facilitated through the provision of knowledge interchange languages and by shared commitments to the ontology [CCS01]. Gruber and Olsen [GO14] use a simple example to demonstrate how ontologies enable knowledge sharing,

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

as cited by Cheng et al. [CCS01]. They define an ontology for mathematical modeling in engineering (*EngMath*) that defines the conceptual scheme of mathematics used in engineering. By using such a shared, formal conceptualization engineers may express ambiguity-free knowledge stemming from resources available within the ontology [GO14]. For example, ambiguous equations such as  $V = IR$  often rely on contextual user knowledge for their proper understanding, knowledge which may simply stem from long-established convention. The EngMath ontology provides engineers and computer systems with domain-specific terms, such as ‘voltage’ or ‘resistance’, that fully determine the conceptual model the equation draws on, thus resolving any ambiguity [CCS01]. Besides knowledge sharing, other arguments for using ontologies to describe consumed knowledge in natural language-based applications include enabling reuse of domain knowledge between applications, and making domain assumptions explicit allowing knowledge engineers to change these assumptions when the knowledge about the domain changes. Ontologies are also employed in dialogue-enabled systems because they allow separating the domain knowledge from the operational knowledge employed by a Natural Language Interface (NLI) [Plh14].

Most commonly, the definition of ontology in the computer science literature goes as follows: “an ontology is a formal, explicit specification of a shared conceptualization” [SBF98]. In this definition, ‘conceptualization’ means an abstract model of some phenomenon in the world, given by all its identified relevant concepts. The term ‘explicit’ signals that the type of concepts used and the constraints on their use have to be explicitly laid out, whereas ‘formal’ indicates that the ontology needs to be actionable by a machine, which excludes ambiguous representations such as natural language. Other definitions that can be found in the relevant literature include “a collection of concepts within a domain together with the relationships among these concepts” [Plh14], and “the theory of objects in terms of the criteria which allow one to distinguish between different types of objects and the relations, dependencies, and properties through which they may be described” [Tow04]. Ontologies can be regarded as being made up of a vocabulary of terms along with a specification of their meaning (i.e. their definitions and an indication of how they are inter-related). These resources collectively impose a structure on the domain and constrain possible interpretations of it [Usc98]. An ontology can also be described in formal representation as follows (definition by Maedche and Staab [MS01], cited in [Liu10]):

An *ontology* is formally expressed by a 5-tuple  $O = \{C, R, H, f, A\}$  in which:

- $C$  represents classes or concepts extracted from the real world.
- $R$  represents the relationships between two concepts.  $R \cap C = \emptyset$  i.e. classes and relationships are disjoint sets.
- $H : H \subseteq C \times C$  is the class hierarchy or taxonomy, where  $H(c_1, c_2)$  means that  $c_1$  is a sub-concept or subclass of  $c_2$ .
- $f : R \Rightarrow C \times C$  is a function which links classes non-taxonomically.
- $A$  is a set of ontology axioms expressed in a logical language.

Ontologies are however better understood in the greater context of linked data within the Semantic Web, where multiple heterogeneous vocabularies are “shared and reused across application, enterprise, and community boundaries” [Wor17]. Next section discusses ontologies within this web of data.

### 3.2.1 Linked Data and the Semantic Web

The unprecedented success of the World Wide Web (WWW) as the preferred means of global communication is largely motivated by its distributed nature and the ability for anyone to publicly publish documents that can link to other Web documents. This ability is greatly enhanced by virtue of a common approach to modelling data based on shared vocabularies [ALG]. Even though most Web sites have some degree of structure, they are created in HTML, a language which is oriented towards structuring text documents instead of data [HB11]. On the other hand, interoperability within the Web is fostered when data publishers reuse each other’s vocabularies and schemas; a *semantic agreement* between data publishers is thus paramount to stimulate reuse and rapid development of suitable, error-free data models on the Web [ALG]. Re-using available vocabularies provides many critical advantages to schema design such as [ALG]: (1) aiding interoperability, (2) indicating the credibility of the schema, and (3) preventing designer efforts to be wasted in ensuring that the schema is well defined and properly hosted.

To address this lack of standardization on the Web, the use of multiple different ‘microformats’ along with Web Application Programming Interfaces (APIs) has been suggested [HB11]. These microformats may be employed to tightly specify knowledge referring to specific types of entities, such as organizations, events, reviews and ratings of products and services, etc. However, these formats only allow for a very constrained representation of small sets of entities and, moreover, they generally do not permit relationships between entities to be expressed, and are thus not suitable for sharing arbitrary data on the Web. Perhaps most importantly, these constrained disparate microformats or Web APIs function like isolated “data islands” within the Web, as they require clients to be in possession of a great deal of *a priori* knowledge about their local scope [HB11]. For example, a numeric ID for a specific resource is most likely useless for clients accessing it without in-depth knowledge about the system hosting it. Therefore, if knowledge is to be efficiently distributed and used at a wider scale, a standard mechanism acknowledging the existence and meaning of vocabularies and their connections is required. This mechanism is commonly known as **Linked Data**, and is made up of a set of best practices for publishing and interlinking data within the WWW. In turn, these best practices have been named the *Linked Data principles*, and are generally laid out as follows [HB11]:

- Use strings of characters in Uniform Resource Identifier (URI) format to unambiguously name things of interest (i.e. resources)
- URIs must follow the `http://` URI scheme so they may be accessed by any client that understands the Hypertext Transfer Protocol (HTTP) protocol

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

- Useful information about the resource should be given when its URI is looked up by an HTTP client, for example, a Web browser
- Links to other URIs need to be included to allow for discovery of related resources

It is promptly observed how these principles closely mimic the architecture of the WWW. Web documents and applications can be hosted in many different types servers and consumed by many different types of client applications as well. Much in the same way that the Web uses URLs, HTTP, HTML and hyperlinks within HTML documents to transcend these different technical architectures enabling users to navigate it, Linked Data uses the Linked Data principles previously described to allow data sharing and reuse at a global scale on what has been called the ***Semantic Web*** [HB11]. In short, the Semantic Web was defined by the creator of the Web, Tim-Berners Lee, as “an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [BLJL01].

The Linked Data principles are nowadays being adopted by a large number of individuals and organizations to publish their data in the Semantic Web, resulting in a global data space known as the *Web of Data*. This Web of Data forms a giant global graph containing billions of RDF triples stemming from multiple sources which cover a multitude of different domains [HB11]. Entities within this global data graph are interconnected by HTTP-accessible RDF links, thus enabling dynamic discovery of new data sources by Semantic Web applications, whilst allowing data publishers to represent data in unconstrained ways. As of March 2019, more than 1200 vocabularies had been published in the Linked Data format<sup>1</sup>. Some of these vocabularies are not constrained to a single domain, but span multiple topics. This cross-domain coverage is vital for creating such a global graph of data avoiding data fragmentation into *data islands* [HB11]. The best example of such cross-domain vocabulary is *DBpedia*, a “community effort to extract structured information from Wikipedia and to make this information accessible on the Web” [BLK<sup>+</sup>09]. DBpedia leverages the huge amounts of information available on Wikimedia projects, such as Wikipedia, by extracting it and making it available on the Web following the Linked Data principles. Currently, the English version of the DBpedia knowledge base describes more than 4.5 million things, out of which 4.22 million are classified in a consistent ontology<sup>2</sup>.

In order to be publishable, Linked Data needs to follow a certain specification. Issued data on the Web following the Linked Data principles is represented using the Resource Description Framework (RDF), which is detailed in next section.

#### 3.2.2 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a Linked Data specification developed by the W3C (c.f. Section 2.7.1). The broad goal of RDF was initially expressed as “to define a

---

<sup>1</sup> According to available data at <https://lod-cloud.net>

<sup>2</sup><https://wiki.dbpedia.org/about>

mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (*a priori*) the semantics of any application domain” [Las99]. In short, RDF is a general, flexible data model used to conceptually describe and model things in the world (e.g. people, physical objects and locations, abstract concepts) and virtual resources, as well as how things relate to each other (i.e. Linked Data). Understanding this ability to semantically connect concepts is vital in order to differentiate vocabularies expressed by using RDF as opposed to traditional hyperlinked HTML documents on the Web. Whereas HTML hyperlinks simply indicate that two documents are related in some way, the nature of the relationship needs to be inferred by the (human) reader. On the other hand, RDF allows vocabulary authors to explicitly define the nature of the relationships between resources. Therefore, linked data expressed following the RDF data model allows disparate data sources to be connected into a single global data space [HB11]. In order to understand how resources are defined and related to each other in RDF, let us take a look at the specific vocabulary defined by the RDF data model.

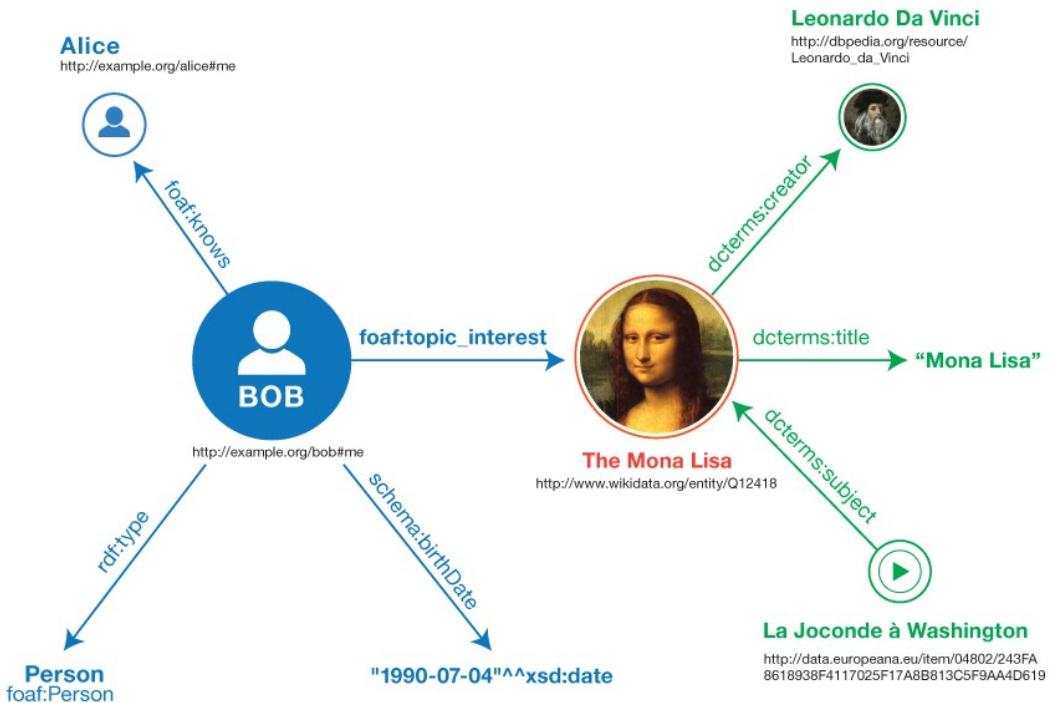


Figure 3.3: A small, informal RDF graph [Sch14]. Every resource (thing of interest except *Literals* and *blank nodes*) is uniquely identified by its URI, e.g. <http://example.org/alice#me>. Often, URIs are made up of a *namespace* referring to a specific context, such as <http://example.org/alice> (or a *namespace prefix* such as foaf); and a *fragment identifier* that comes after the hash symbol (#) such as me or Person. RDF nodes are therefore divided into *URIs*, *Literals*, and *blank nodes* (not illustrated).

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

The RDF data model represents information as a number of directed graphs (in the mathematical sense) made up of subject-predicate-object triples, see Figure 3.3 for an informal example of such a graph. It is designed for the integrated representation of information originating from multiple sources, is heterogeneously structured, and is represented using different schemata [HB11]. Moreover, RDF aims to act as the intermediate between other data models used on the Web. Triples in RDF graphs are statements that have the following structure [Sch14]:

```
<subject> <predicate> <object>
```

Each of these statements refers to a relationship, whose nature is expressed by the predicate, between the *subject* and the *object*. Subject, predicate (called *property* in RDF), and object are resources in the underlying data model. This statement is taken in a directed way from subject to object, mirroring the structure of a simple sentence in natural language, thus the denominations subject and object. It is important to note that in these statements the RDF data model does not make any assumptions about what resource URIs stand for, and they must be used in combination with external vocabularies (ontologies or other conventions) that provide us with the semantic information about their resources. An example of a statement (triple) expressed by the model depicted in Figure 3.3 is as follows:

```
<bob:me> <foaf:knows> <alice:me>
```

In this triple, a number of namespace prefixes have been employed to shorten URIs resources: bob is used instead of `http://example.org/bob`, alice is short for `http://example.org/alice`, and foaf stands for `http://xmlns.com/foaf/0.1/`, namespace of the Friend-Of-A-Friend (FOAF) vocabulary describing social networks between people and other real-world associations. Much like a statement in natural language, this triple could be interpreted as “Bob knows Alice”, not only linking two resources (Bob and Alice), but also indicating the kind of relationship that takes place between them. Some other statements that may be derived from the ontological triples depicted by the graph in Figure 3.3 are “Bob is a person”, “Bob is interested in the *Mona Lisa*”, “the *Mona Lisa* was created by Leonardo da Vinci”, “Bob has birth date 4<sup>th</sup> of July 1990”. The latter triple has the particularity that its object is not a resource of the graph, but a basic value (in this case, a date).

Basic values are not identified by URIs, since they represent data values common across many domains, such as numbers, strings of text, date timestamps, Boolean values, and so on. They may be however associated with a datatype which enables them to be interpreted correctly. The datatype abstraction used in RDF is compatible with XML Schema [Pet12], reusing many of its built-in datatypes (such as `xsd:string`, `xsd:boolean`, `xsd:float`, `xsd:date`, and `xsd:language`), where `xsd` is the namespace prefix for XML Schema: `http://www.w3.org/2001/XMLSchema#`. Moreover, RDF expands XML Schema with two additional, non-normative, datatypes: `rdf:HTML` and `rdf:XMLLiteral` [Cyg14]. These kind of resources not described by URIs are named *Literals*. Examples of typed literals, where a literal is accompanied by its datatype, are “42”<sup>^^</sup>`xsd:integer`, “1.9”<sup>^^</sup>`xsd:float`, and “Mona Lisa”<sup>^^</sup>`xsd:string`. Language-tagged string Literals are represented in a slightly different manner by adding an at symbol (@) in between the string Literal and

the language tag, such as in “*La Joconde*”@fr, representing a French-language string. Literals may only occur in the object of an RDF triple. Finally, a resource for which a URLs or Literal is not given is called a *blank node* or *bnode*. Resources represented by blank nodes are named *anonymous resources*. Blank nodes cannot take place in the predicate of a triple. A formal, shorter definition of the different elements making up the standard RDF data model can be consulted in [HD05].

It is important to note that the data models here described are in the form of an abstract syntax (i.e. not complying to any specific syntax) in order to improve their readability. However, different concrete serialization formats exist for storing RDF graphs, such as Turtle, RDF/XML, RDFa, or JSON-LD [Sch14]. These formats allow RDF data models to be stored as plain text in computer files or databases. Two of these formats have been standardized by the W3C, namely, RDF/XML and RDFa. Moreover, several other non-standard serialization may be employed to fulfill specific needs [HB11]. Their implementation details are however not relevant for the high-level description of RDF data models given in this chapter.

As already mentioned, RDF provides a powerful, simple mechanism for describing resources as subject-predicate-object triples. However, it does not provide any domain-specific terms for describing classes of things in the world and how they relate to each other [HB11]. These terms may be expressed by taxonomy and classification description languages that build on and augment the meaning of what the RDF data model can express, the most commonly used being RDF Schema (RDFS) and the Web Ontology Language (OWL). RDFS and OWL provide vocabularies, known as *semantic extensions* of RDF, that allow describing conceptual models in terms of classes and their properties [HB11], allowing for greater semantic expressivity. Moreover, when paired with a suitable reasoning engine, such as Hermit<sup>3</sup>, these extended models allow implicit relationships to be inferred from data [HB11]. For example, ontological inconsistencies may be automatically detected or subsumption relationships between classes identified. The availability of automatic reasoners is another advantage of employing formal data models in the form of ontologies to underpin visualization-related data, as incoherences or missing bits of information in the semantic annotation (c.f. Chapter 4) of a graphic may be automatically detected and corrected by such technologies. This is especially advantageous when manually authoring data models, as the automatic inference of missing triples can save a significant amount of time from the authoring process. These extensions to RDF will be introduced in the following two sections.

### 3.2.3 RDF Schema (RDFS)

The RDF Schema (RDFS) language [Bri14] is a semantic extension of RDF allowing ontology authors to define semantic characteristics of data expressed by the RDF data model [Sch14]. Its vocabulary is defined in a namespace conventionally called `rdfs`, which is identified by the following URI:

`http://www.w3.org/2000/01/rdf-schema#`

---

<sup>3</sup><http://www.hermit-reasoner.com/>

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

Due to historic reasons, the RDF namespace (`rdf` by convention) is separately referred to by RDFS with the following URI:

`http://www.w3.org/1999/02/22-rdf-syntax-ns#`

RDF Schema groups related resources and the properties between these resources by means of a class and property system, similar to the type systems of object-oriented languages such as Java or Python [Bri14]. The main difference between vocabularies described via RDF Schema and data models derived from object-oriented programming languages is that, instead of defining a class in terms of the properties its instances may have, RDFS describes properties in terms of the classes of resource to which they apply by employing the domain and range mechanism of properties [Bri14]. For example, a traditional object-oriented data model may specify that a Dog class has an `owner` attribute of type Person. On the other hand, the RDFS approach may consist on defining a `has_owner` property with a domain of Dog and a range of Person. This offers greater flexibility to the vocabulary designer, as new properties with the same domain and range may be added without having to re-define their class definitions.

The first building block of RDFS vocabularies are *classes*. Informally speaking, classes are resources acting as the means for grouping other resources (the *instances* of the class) together. Associated with each class is its *class extension*, which is the set of instances of the class [Bri14]. Two classes may have the same class extension, but still be different classes. Moreover, a class may belong to its own class extension i.e. a class may be an instance of itself. RDFS classes are themselves instances of the `rdfs:Class` class. Instances are grouped together under their parent classes by using the `rdf:type` property. For example, the following triple:

```
<bob:me> <rdf:type> <foaf:Person>
```

is used in the small knowledge base depicted in Figure 3.3 to assert that the resource with the URI `bob:me` is an instance of the URI `foaf:Person` (i.e. that Bob is a person as underpinned by the vocabulary defined in the FOAF language). The `rdfs:subClassOf` property may furthermore be employed to state that a class is a *subclass* of another class. When a class  $C'$  is a subclass of a class  $C$ , all instances of  $C'$  are also instances of  $C$ . Additionally, if  $C'$  is a subclass of  $C$ , then  $C$  is said to be a *super-class* of  $C'$ . RDF *datatypes*, introduced in section 3.2.2, are classes whose instances are the members of the value space of the datatype [Bri14]. Other important classes included in RDF Schema are (where their namespace prefix, `rdfs:`, has been omitted for clarity) [Bri14]:

- **Resource:** the “class of everything”; all other resources are subclasses of this class.
- **Class:** the class of resources which are RDF classes.
- **Literal:** the class of literal values such as strings and integers.
- **Datatype:** already discussed, the class of datatypes employed by typed literals.

- `langString`: the class of language-tagged string values.
- `Property`: the class of RDF properties, which are introduced next.

The next constituent block of RDFS are the semantically enhanced counterparts of RDF properties or predicates, introduced in the previous section as the element of an RDF triple connecting its subject with its object in some meaningful way. Firstly, RDFS extends the concept of RDF properties by defining the `rdfs:subpropertyOf` property. This property is employed to state that one property is a *subproperty* of another property, much in the same way that subclasses may be hierarchically arranged by employing the previously mentioned `rdfs:subClassOf` property. If a property  $P'$  is a subproperty of the property  $P$ , all triples containing  $P'$  are also related by  $P$ . For instance, if a vocabulary contains the following triple:

```
(uvo:eg01, ddo:expresses_ordinal_information, dom:population)
```

then we can also infer that

```
(uvo:eg01, ddo:expresses_information, dom:population)
```

exists in the ontology, since `ddo:expresses_cardinal_information` is characterized as a subproperty of `ddo:expresses_information` in the Data Domain Ontology (DDO), outlined in Section 3.5.2. Moreover, the RDFS specification defines the concepts of a property's *range* and *domain*. This is introduced via two additional properties: `rdfs:range` and `rdfs:domain`. Range property occurrences are employed to indicate that the values of a different property must be instances of one or more classes. For instance, the following triple:

```
(uvo:has_shape_named, rdfs:range, uvo:Shape_VA)
```

states that the `uvo:has_shape_named` property must have values which are instances of the `vis:Shape_VA` class. Informally expressed, this means that elements having a shape as stated by the `has_shape` property of the Upper Visualization Ontology (UVO) (c.f. Section 3.6.11) need to conform to the prescribed shapes the ontology supplies as instances of the `Shape_VA` class of said ontology. Likewise, a domain property occurrence may be used to indicate that a resource having a given property must be of an instance of one or more classes. For example, a domain property can be applied to assert that all things having a shape must be Graphic Object instances, preventing other ontological resources, such as colors, from having associated shapes.

Finally, the RDF Schema language provides two properties intended for human consumption of formal vocabularies, namely `rdfs:label` and `rdfs:comment`. The former is the standard way of providing a human-readable version of a resource's name. Given that ontological resources may have a shortened, non-descriptive, or in some other way difficult to comprehend name for a human reader, this property is the most common way of adding alternative, more verbose or clearer names for resources as a string Literal that can be output to a user navigating the vocabulary. In case the RDF label for a resource may not suffice for its proper understanding due to its brief nature, the RDF comment property may be employed in a similar fashion to include a longer description or comment about said resource. These two

properties jointly act as the in-line documentation of an ontological resource. Some examples of popular vocabularies represented in RDFS include the already mentioned FOAF vocabulary, the Dublin Core Schema [DCM12a], which describes digital resources such as webpages, physical resources such as books, and objects like artworks; schema.org [Try16], created to mark-up websites such that they can be more efficiently discovered by search engines; and the SKOS core vocabulary, which expresses “the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, ‘folksonomies’, other types of controlled vocabulary, and also concept schemes embedded in glossaries and terminologies” [Mil05].

As we have seen, RDFS only offers a very basic semantic extension of RDF that defines how resources are structured into classes and connected through properties. However, in order to build valid, semantically coherent ontologies, a more powerful vocabulary allowing axiomatic constraints of knowledge triples is more favorable. The Web Ontology Language (OWL) is a data modeling language that enables vocabulary creators to define ontologies with a much higher degree of semantic expressivity, and is discussed in the following section.

#### 3.2.4 Web Ontology Language (OWL)

The Web Ontology Language (most commonly abbreviated as OWL instead of WOL) is a “Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things” [Hit12]. The OWL vocabulary is defined on the so-called ‘OWL universe’, which divides part of the RDF data model into three parts, namely OWL individuals, classes, and properties [Pat04], akin to RDF Schema, described in section 3.2.3. However, OWL has not been designed to only formulate and exchange vocabularies, but also to allow high-level reasoning with its contained knowledge, generally by means of a reasoner, to take place. Put in other words, OWL captures the aspect of human intelligence that allows us to draw consequences from available knowledge. This feature makes it a suitable language to describe complex knowledge about one or more domains of interest. Even though not all aspects of human knowledge can be reflected in OWL, it can be considered “a powerful general-purpose modeling language for certain parts of human knowledge” [Hit12]. The end result of modelling knowledge with OWL is called an *ontology*, and, much like RDFS vocabularies, OWL ontologies may be converted to RDF models through a number of different syntactic mappings in order to serialize them into a suitable RDF serialization format (see section 3.2.2). The OWL language is made up of three basic notions, namely *axioms*, *entities*, and *expressions* [Hit12]:

- *Axioms* are the basic statements (triples) expressed by an ontology
- *Entities* refers to each individual class, instance, or property in the ontology.
- *Expressions* are complex knowledge descriptions derived from a combination of several entities. Expressions are a central feature of OWL. They enable names and entities to be powerfully combined using so-called *constructors*. For example, the concepts of a metric space and a diagram’s axis can be combined to express the class of metric

axes. This derived class could then take place in ontological statements or in other expressions.

The OWL namespace, conventionally associated with the namespace `owl`, is as follows:

`http://www.w3.org/2002/07/owl#`

As previously stated, the basic modelling structures of OWL are classes, properties, and individuals. OWL extends RDFS and thus classes, individuals belonging to them, as well as sub- and super-classes may be defined in an OWL ontology. OWL however introduces further class axioms that state sufficient and/or necessary characteristics of a class. For instance, the assertion `owl:equivalentClass` allows to specify identical class descriptions, whereas the construct `owl:disjointWith` expresses that two class descriptions have no members in common. Moreover, OWL classes may be described by six different class descriptions, the most common of them being a class identifier (URI reference) analogously to RDF. The other available class descriptions describe *anonymous classes* (i.e. not named by an URI) by placing constraints on their class extension (a *class extension* is the set of individuals a given class is associated with):

- An *exhaustive enumeration* of individuals belonging to the class. For example, the class `ns:RGB_Color` could be described by the individuals `ns:Red`, `ns:Green`, and `ns:Blue`.
- A *property restriction*, which describes the class of all the individuals that satisfy the restriction. For example, we could stipulate that all individuals having some value of the `uvo:has_X_coordinate` and `uvo:has_Y_coordinate` properties describe the `uvo:Graphic_Object` class, since everything that can be positioned in a Graphic Space is a Graphic Object (c.f. Section 3.6.6).
- The *intersection* of two or more class descriptions, which characterizes a class containing exactly those individuals that are members of all the stipulated class descriptions.
- The *union* of two or more class descriptions, which defines a class containing all those individuals that are members of at least one of the stipulated class descriptions.
- The *complement* of a class description, which describes a class that contains exactly those individuals that do not belong to another class.

When it comes to properties, perhaps the most characteristic semantic supplement in OWL is that they are divided into two main categories:

- *Object properties*, which link individuals to individuals. For example, the `vdo:is_labelled_by` object property (see Section 3.7.12) connects a labelled Graphic Object instance with another instance that labels it.

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

- *Datatype properties*, which connect individuals to data values. For instance, the `uvo:has_text` datatype property connects a Graphic Object instance depicting text with its actual text value (string).

Properties defined in OWL may also be semantically enhanced with extra axioms. Constructs already available in RDF Schema, such as `rdfs:subPropertyOf`, `rdfs:range`, and `rdfs:domain`, are part of the OWL language as well. In addition, OWL provides new property axioms that define special property characteristics. Some of them define relations between properties. For example, the `owl:equivalentProperty` property can be used to state that two properties have the same property extension (similarly to class extensions, a *property extension* is the set of instances associated with a property i.e. subject-object pairs of its property triples). Inverse properties, declared by the `owl:inverseOf` construct, defines an inverse relation between two properties, such as stating that `vdo:is_label_of` is the inverse property of `vdo:is_labeled_by`, and vice versa. Different OWL property axioms may be employed to place constraints on the global cardinality of properties. For instance, the `owl:FunctionalProperty` is a built-in OWL class whose instances are properties that can have an unique value. An example of functional property is `uvo:has_X_coordinate`, as resources must at most have one associated value as their coordinate on the X axis of the canvas (c.f. Section 3.6.2). Conversely, an *inverse functional* property, which are declared as instances of the `owl:InverseFunctionalProperty` class, may only have one subject.

Lastly, some property axioms assert logical characteristics of properties. These include two built-it OWL classes: `owl:TransitiveProperty` and `owl:SymmetricProperty`. The former is instantiated to define occurrences of transitive properties, whereas the latter is employed to declare symmetric properties. If a property  $P$  is defined as transitive, then, if occurrences  $(x, P, y)$  and  $(y, P, z)$  appear in the ontology, the property occurrence  $(x, P, z)$  can be inferred to also exist. For instance, the `uvo:has_graphic_object` property is a transitive property so that the recursive nature of graphics, discussed in Section 3.5.2, can be characterized in UVO. Symmetric properties have two occurrences in which the subject and object are mirrored i.e. if the triple  $(x, P, y)$  of a symmetric property  $P$  exists in the ontology, then the triple  $(y, P, x)$  must exist too.

Regarding individuals, three constructs are provided by OWL for stating facts about their identity. Due to the open-world assumption that ontologies operate in, we cannot assume that different named resources refer to different objects in the world. Therefore, `owl:sameAs` may be employed to indicate that two URI references refer to the same individual. Conversely, the `owl:differentFrom` property can be used to state that two URI references refer to different individuals. Finally, `owl:AllDifferent` is a class whose instances are declared to refer to different individuals. This is a special OWL idiom that is generally used to prevent a large number of `owl:differentFrom` occurrences from appearing in an ontology possessing many different individuals. Moreover, in addition to RDF datatypes, OWL declares one additional axiom for defining a range of data values, also known as *enumerated datatype*. The `owl:oneOf` property is employed to define enumerated datatypes in OWL ontologies.

The subject of `owl:oneOf` is a blank node of class `owl:DataRange`, and its object is a list of literals.

OWL also offers additional constructs for annotating classes, properties, individuals, and ontologies, mostly stemming from RDF, such as `rdfs:label` and `rdfs:comment`, already discussed in Section 3.2.2. Moreover, some properties such as `owl:versionInfo`, `owl:priorVersion`, and `owl:backwardCompatibleWith` are included to provide support for ontology use by versioning systems. Constructs from the OWL and other namespaces may be employed to describe ontologies, given that an ontology is a resource itself [Bec04]. An ontology can be defined as an instance of the `owl:Ontology` class, with semantic annotations such as a RDFS label and comment, OWL version information, etc. These constructs related to an `owl:Ontology` instance make up the so-called *ontology header*, and is generally found at the beginning of the serialized ontology file. A highly relevant construct in the context of this thesis is the `owl:imports` property, which references external OWL ontology definitions from the importing ontology. In this manner, their meaning can be included in the importing ontology by reasoners and other applications working with a given vocabulary. This lets knowledge engineers modularize the designed meaning, allowing complex domain hierarchies to be easily represented.

## OWL 2

Eight years after OWL was developed, in 2012 the W3C completed development of OWL 2 [W3C12], an extension and revision of OWL. Its new features can be grouped into six main categories, namely: (1) syntactic sugar for common statements, (2) new constructs that increase expressivity, (3) extended support for datatype, (4) simple meta-modeling capabilities, (5) extended annotation capabilities, and (6) minor features and innovations. Let us now very briefly introduce the most relevant of these new features. Figure 3.4 shows a diagrammatic view of the main (RDF-compatible) components of OWL 2. For further information the reader is directed to the respective W3C recommendation [Gol12].

Some of the syntactic sugar added by OWL 2 includes defining a class as the union of other classes, which are pairwise disjoint, by means of the `DisjointUnion` shorthand for a number of OWL axioms. Pairwise disjointness of classes themselves can also be concisely expressed via the `DisjointClasses` shorthand. OWL 2 also extends the original capabilities of OWL to assert values of a property for an individual. OWL had no direct means for directly asserting values that an individual did not have i.e. *negative property assertions*. Thus, two new property assertions, `NegativeDataPropertyAssertion` for negative assertions related to datatype properties, and, similarly for object properties, `NegativeObjectPropertyAssertion`, were added to OWL 2.

Among the new OWL 2 constructs that extend the supported expressivity we find the `ObjectHasSelf` restriction, which allows objects to relate to themselves by a given property i.e. local reflexivity. OWL 2 also expands cardinality restrictions of properties by adding qualified cardinality restrictions via three new constructs: `ObjectMinCardinality`, `ObjectMaxCardinality`, and `ObjectExactCardinality`. In this manner, restrictions can be

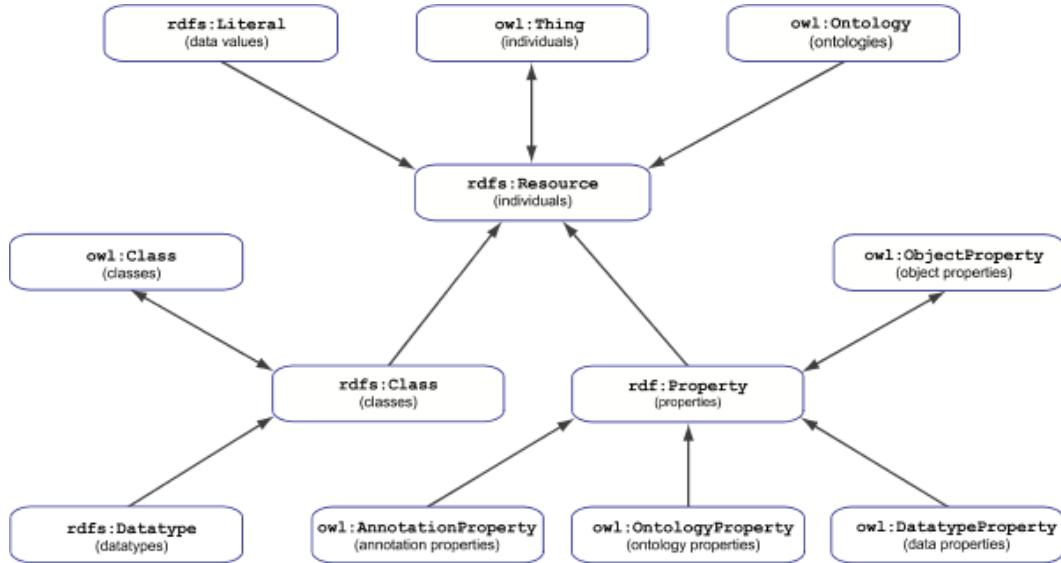


Figure 3.4: Parts hierarchy of the OWL 2 RDF-based semantics [Sch19].

applied not only to the number of instances of a property (for instance, that a Graphic Object may have only two colors), but also to the class or data range of the instances to be counted (that the two colors must be shades of green). These constructs may be applied to object properties, but corresponding constructs exist for datatype properties as well. OWL 2 also adds new characteristics of object properties, namely, *reflexive*, *irreflexive*, and *asymmetric* object properties. OWL 2 expands the concept of disjointness from classes to properties as well by means of the `DisjointObjectProperties` and `DisjointDataProperties` constructs for object and datatype properties, respectively. Properties may also be defined as a composition of other properties, allowing for properties to be propagated. For example, if a graphic element A is a part of another graphic element B, it could be stated that A is located in B by defining the `located_in` property as a subproperty of a `ObjectPropertyChain` axiom inclusively chaining both properties. Finally, keys can natively be defined in OWL 2 by employing the `hasKey` property, which allows keys to be defined for a given class in order to uniquely identify its individuals by values of their properties.

Four extended capabilities have been added to datatype properties in OWL 2. The first one consists on the inclusion of new datatypes (such as numbers, more XML Schema datatypes, strings with or without language tag, Boolean values, binary data, etc.) and *datatype restrictions* (via the `DatatypeRestriction` construct). Datatype restrictions “make it possible to specify restrictions on datatypes by means of constraining facets that constrain the range of values allowed for a given datatype” [Gol12]. For example, minimum or maximum allowed values, or maximum or minimum value lengths can be specified. Next, OWL 2 includes the necessary constructs for n-ary datatypes i.e. representing relationships between the values of one property. Another new capability of OWL 2 is the possibility of explicitly naming a new datatype via the `DatatypeDefinition` construct. Finally, new datatypes

can be generated by combining other datatypes via intersection, union, and compliment constructs.

OWL 2 also allows for simple *metamodeling* of classes and individuals by employing the same name to refer to a class and an individual belonging to its meta-class. However, names cannot be reused for class and datatypes, nor for different properties. When it comes to its extended annotation capabilities, OWL 2 allows for annotations on axioms in addition to ontologies, entities, anonymous resources, and annotations themselves. Annotations are implemented through annotation properties (such as `rdfs:label`, previously discussed) in combination with the `AnnotationAssertion` construct for annotation of classes, properties, and anonymous individuals, and the `Annotation` construct for annotations of axioms and ontologies. Moreover, annotation properties can be assigned domain and range information as well as participate in annotation property hierarchies much in the same way of OWL properties. None of these annotations carry any semantics and will be thus ignored by reasoners.

Finally, among the minor innovations and features of OWL 2 we find the Declaration construct, which signals that an entity is part of the vocabulary of an ontology thus making it easier to catch errors due to axiom misspellings. OWL 2 also adds top and bottom object and datatype properties, in addition to the top (`owl:Thing`) and bottom (`owl:Nothing`) entities for classes already existing in OWL. Next, OWL 2 uses Internationalized Resource Identifiers (IRIs) instead of URLs as the standard mechanism for identifying resources in order to cater for international needs of using identifiers that contain characters outside of the ASCII character set. Anonymous individuals are identified using node IDs instead of being individuals without identifiers.

## OWL Profiles

Meaning can be assigned to OWL 2 ontologies in two alternative ways. The first one, known as *Direct Semantics*, is strongly related to the semantics of description logics and extends the semantics of the SROIQ description logic. The second one, *RDF-based Semantics*, defines the semantics of OWL to be fully compatible with the RDF Semantics specification, expressing an OWL 2 ontology as a semantically extended RDF graph. Ontologies that can be applied direct semantics are usually said to be in the *OWL 2 DL* subset of OWL 2, whereas ontologies that are not in OWL 2 DL are said to belong to *OWL 2 Full*, and can be only interpreted using RDF-Based semantics. These two semantics are generally quite similar, the main difference between them being that under the direct semantics annotations have no formal meaning, whereas under RDF-based semantics there are extra inferences that arise from the RDF view of the universe [Hit12]. In practice, OWL 2 DL can be seen as a syntactically restricted version of OWL 2 Full that makes it easier to write reasoners that operate in ontologies specified by it. Whereas production quality reasoners that cover the entire OWL 2 DL language under the Direct Semantics exist, no such reasoner is available for OWL 2 Full.

In addition to OWL 2 DL and OWL 2 Full, OWL 2 specifies three *profiles*. An OWL 2 profile, also known as *dialect*, is a pruned version of OWL 2 that trades some its expressive power for

more efficient reasoning capabilities. OWL 2 profiles aim to provide approachable subsets of OWL 2 which suffice for a variety of applications. These profiles are namely OWL 2 EL, OWL 2 QL, and OWL 2 RL [Hit12]. Some of these profiles can be seen as syntactic fragments of other profiles. For instance, OWL 2 DL can be seen as a fragment of OWL 2 Full, and OWL 2 QL as a fragment of OWL 2 DL. OWL 2 EL has been designed with the goal of describing large bio-health ontologies with complex structural descriptions and a great number of classes, and is similar to OWL 2 DL, allowing for a very expressive class expression language with no restrictions on how they may be used on axioms. On the other hand, OWL 2 QL is thought to be tightly integrated with relational database (RDB) management systems and can be realized using standard RDBs technologies such as SQL. OWL 2 RL includes many commonly used features of RDFS such as inverse properties and subproperty hierarchies, but restricts class axioms asymmetrically and forbids existential qualification of roles to a class expression. Furthermore, it does not support property chain axioms and equality. Finally, OWL 2 RL is aimed to be employed by applications that require scalable reasoning without sacrificing too much expressive power, accommodating OWL 2 applications that require higher efficiency in exchange for reduced semantic expressivity, as well as RDF(S) applications that make use of some of the OWL 2 features. The syntactic subset of OWL 2 described by OWL 2 RL is straightforward to implement using rule-based technologies, and thus OWL 2 RL is very suited to enriching RDF data. OWL 2 disallows axioms where the existence of an individual enforces the existence of another individual. Moreover, it restricts class axioms asymmetrically.

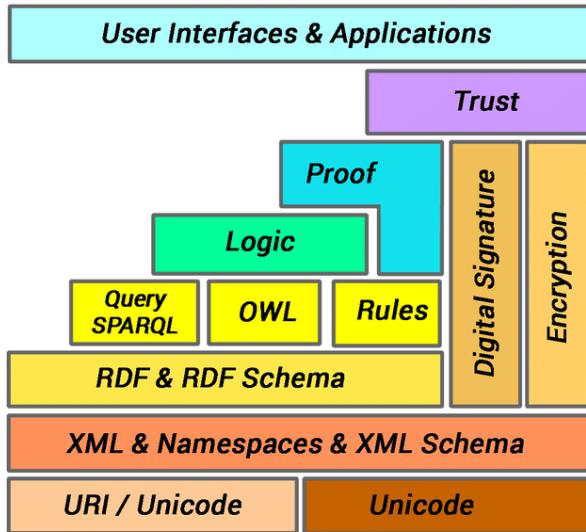


Figure 3.5: Technological stack of the Semantic Web [GB16].

### 3.3 Linked Data Applications in Computer Vision

The motivations for employing ontologies within the plethora of data of the Semantic Web was already introduced in Section 3.2.1. Figure 3.5 displays the hierarchy of technologies

that can be exploited by the Semantic Web and related applications, including the previously described Unicode- (or ASCII-) encoded URLs, URI-based namespaces and XML Schema for describing resources, RDF(S), and OWL/OWL 2 dialects. These formalisms in combination suffice to describe a wide array of conceptualizations that may be employed in disparate sub-fields of human knowledge, including visual information depicted in photography and diagrams.

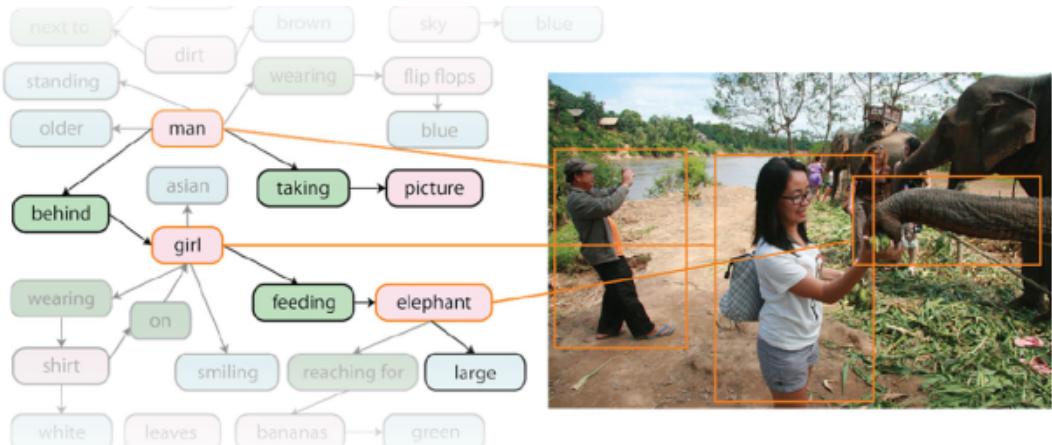


Figure 3.6: A subgraph of the Visual Genome ontology for describing photograph semantics underpinning a real-world scene [KZG<sup>+</sup>17].

In the field of computer vision, the integration of formal vocabularies into intelligent vision systems was early suggested in order to help achieve the ultimate goal of generic image understanding analogous to human vision. However, subsequent research centered their efforts in statistical and mathematical approaches straying off Artificial Intelligence (AI) methods onto pragmatic goals in more constrained scenarios [ATdO<sup>+</sup>18]. Nevertheless, despite impressive recent advancements in machine learning methods for computer vision, these methods still face many limitations, some of which may be resolved with the integration of a suitable vocabulary of well-known *a priori* knowledge. Such assimilation could help mitigate problems such as overfitting of vision models as well as an extreme reliance on the quantity and quality of training data [ATdO<sup>+</sup>18]. Moreover, current methods still perform very poorly when going beyond merely perceptual tasks such as image classification. However, models used in more complex cognitive tasks are still being trained using the same datasets designed for perceptual tasks [KZG<sup>+</sup>17]. In order to successfully tackle cognitive tasks such as question answering, models need to also understand the interactions and relationships between objects in an image. Ontologies come out naturally as a suitable means of formalizing such knowledge. Not only they are an efficient means of incorporating the required contextual knowledge of tasks supported by graphical depictions, but at the same time they allow to derive implicit knowledge by automatic inference [BH10].

Ontologies are generally incorporated in vision systems in three different ways: heavy-weight ontologies (HWO), light-weight ontologies (semantic hierarchies), and formal languages

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

[BH10]. *Heavy-weight ontologies* fully describe concept definitions and relations in the domain of the application, making intensive use of axioms to model knowledge and restrict domain semantics. For instance, a multi-ontology based multimedia annotation model is introduced in [DL06], where an upper, domain-independent, multimedia ontology grounds the vocabulary of a number of lower-level domain ontologies providing multiple domain-specific views of multimedia content. A more recent HWO approach to describing photograph semantics is the *Visual Genome* project of Krisha et al. [KZG<sup>+</sup>17]. The Visual Genome dataset consists on a dense annotation of real-world objects, attributes and relationships depicted in over a hundred thousand photographs. These axioms are canonicalized in region descriptions and question-answer pairs to WordNet (the most popular lexical database of the English language) *synsets* (word synonym sets). Figure 5 shows how Visual Genome resources may be employed to describe a real-world scene in detail.

On the other hand, light-weight ontologies are structured by the use of partial, hierarchically structured vocabularies typically linked via ‘type’ RDF properties [BH10]. This provides a layered abstraction of visualization concepts based on abstraction levels, as introduced in Section 3.1.2. *ImageNet* [DDS<sup>+</sup>09] is probably the most popular dataset of this kind. It offers a large-scale dataset of hierarchically structured images (3.2 million images in total) according to the semantic hierarchy of WordNet. Each set of synonym words in WordNet has an average associated image set comprised of 500 to 1000 images in ImageNet. ImageNet is largely used to train and evaluate applications in object recognition, image classification, and object localization. However, it only covers ‘is-a’ relationships, and further semantics are not embedded within the dataset. These hierarchies focus on structural organization of images, ignoring visual information, a vital part of image semantics [BH10].

Finally, some approaches leverage the inferential capabilities of ontologies as an active framework for computer vision. Image interpretation can be formalized as either *deduction*, where interpretation is “an instantiation of formal knowledge consistent with evidence about the real-world domain” or *abduction*, where “interpretation is an instantiation of formal knowledge which allows to deduce the evidence” [BH10]. Ontologies have also been employed by Content-Based Image Retrieval (CBIR) and exploration systems in order to improve retrieval accuracy and user satisfaction. Some examples of CBIR systems that employ ontologies can be consulted in [LZLM07].

In the field of accessibility to graphics, Communicative Images, discussed in Section 2.6, leverage ontologies to describe the semantics of photographs which are then accessible by blind persons via a Natural Language Interface (NLI). Figure 3.7 shows an example of how the regions of a segmented photograph are connected to resources in a supporting ontology in order to semantically enhance it for accessibility in a Communicative Image. The authoring process of semantically-enhanced graphics will be discussed in Chapter 4.

Describing formal vocabularies nevertheless does not suffice for most use cases of systems allowing users access high-level visual information, as they require not only a characterization of visualization elements, but efficient access to the underlying knowledge bases. Most ontologies quickly grow into having hundreds or thousands of axioms that cannot be easily consumed by humans or even computer systems without some previous filtering. Even more

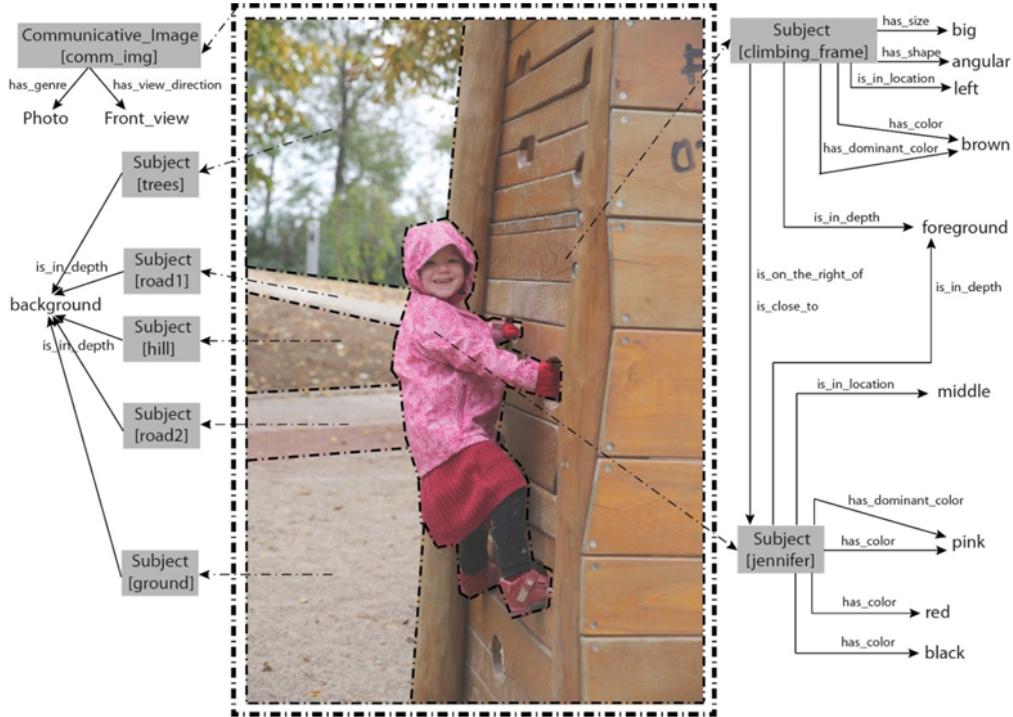


Figure 3.7: Partial semantic annotation of the regions of a Communicative Image [Plh14] via a graphical ontology described in [Ošl09]. Regions in the graphic are associated to resources in a supporting formal vocabulary.

so is the case with data published to the Semantic Web, as, according to the Linked Data principles, it becomes part of a single, global, data space spanning billions of data triples [HB11]. Therefore, an important aspect to take into consideration therefore is how ontologies may be queried in order to fetch only those resources deemed relevant for a particular task. Linked Data applications generally consume global data by following one of the following patterns [HB11]:

- **Crawling Pattern:** mimicking the architecture of classical Web search engines, applications following the crawling pattern crawl the Semantic Web in advance by traversing RDF links. After crawling is completed, the discovered data is processed and integrated within the client application. New sources can be discovered by the crawler at run-time, whereas previously discovered data is cached enabling applications to execute complex queries over large amounts of data with reasonable performance.
- **On-the-fly Dereferencing Pattern:** when this pattern is followed, resource URIs are dereferenced and links are followed the moment that the application requires their data. This approach trades much slower running times for a very high degree of knowledge completeness.

- *Query Federation Pattern:* this pattern is realized by directly sending complex queries specified in formal query languages to a fixed set of data sources. The most commonly employed language of this kind is SPARQL Protocol and RDF Query Language (SPARQL), a recursive acronym, allowing SQL-like queries to be executed against data stored in RDF format. SPARQL queries can however only be used if data sources provide SPARQL endpoints in addition to serving their resources on the Web via dereferenceable URIs. In this manner, applications can fetch remote semantic data without creating a local copy thereof.

Most applications employing ontologies locally as the preferred means for storing domain knowledge make use of SPARQL to query data, as the query federation pattern is preferred when the number of data sources is small [HB11]. However, SPARQL is a formal query language and therefore it is intended to be used by experienced programmers for implementing software applications that consume ontologies. Most end user of these applications, however, need not be familiar with formal query languages or with most aspects of software development. Therefore, more user-friendly approaches to querying ontologies are necessary in order to make client applications usable by a broad amount of users. This one of the main motivations for employing a NLI to ontologies, something that will be discussed at length in Chapter 5.

## 3.4 Formalizing Visualization Semantics

So far we have discussed graphic representation of knowledge (diagrams), the importance of a NLI system enabling diagrams to be non-visually accessible, and its necessity of a suitable technology that bridges the semantic gap between low-level image features and high-level human conceptualizations. We have also resolved that ontologies will be the employed mechanism for encoding this formal mapping. However, we have not yet discussed whether tracing raw visual features into high-level human knowledge is an attainable endeavor. This section aims to answer such question, thus corresponding to the Problem Investigation phase (Section 1.5) of the design problem tackled by this chapter.

### 3.4.1 Controversies in Visualization Formalization

Formalizing visualization semantics is not a novel idea, as several attempts at systematically deriving structured descriptions of the visualization domain can be found in the relevant literature. Some of these approaches proceed at interpreting graphic representations with notions derived from the study of language e.g. [Ric84, Hor98, Eng02, Bar67] and mathematics [Cle94, Kos94, Tuf01, Ber11, TM04], see Figure 3.8 for a small visual example. Authors that derive inspiration from language regard diagrams as “an expression of visual language” that can be analyzed “with regard to its graphic syntax and with regard to its interpretation” [Eng02]. Actually, grounding visual concepts to language is nowadays employed by ontologies used in real-world image understanding tasks, such as Visual Genome [KZG<sup>+</sup>17]. However, the idea of a common language underpinning all aspects of visualization may seem too far-fetched an approach at first. Indeed, Engelhardt points out that “the notion of many

different visual languages seems appropriate, allowing us to distinguish ‘quantitative-bar-chart languages’ from ‘color-coded-geographic-surface languages’” [Eng02]. These specific visual languages (or schemas) possess their own set of compositional rules and their own set of categories of graphic elements with specific syntactic roles [Eng02]. Nevertheless, although data visualization may be made up of a multitude of possible visual languages, these languages have been shown to share many general principles, with each visual language sharing a subset of these general principles [Eng02].

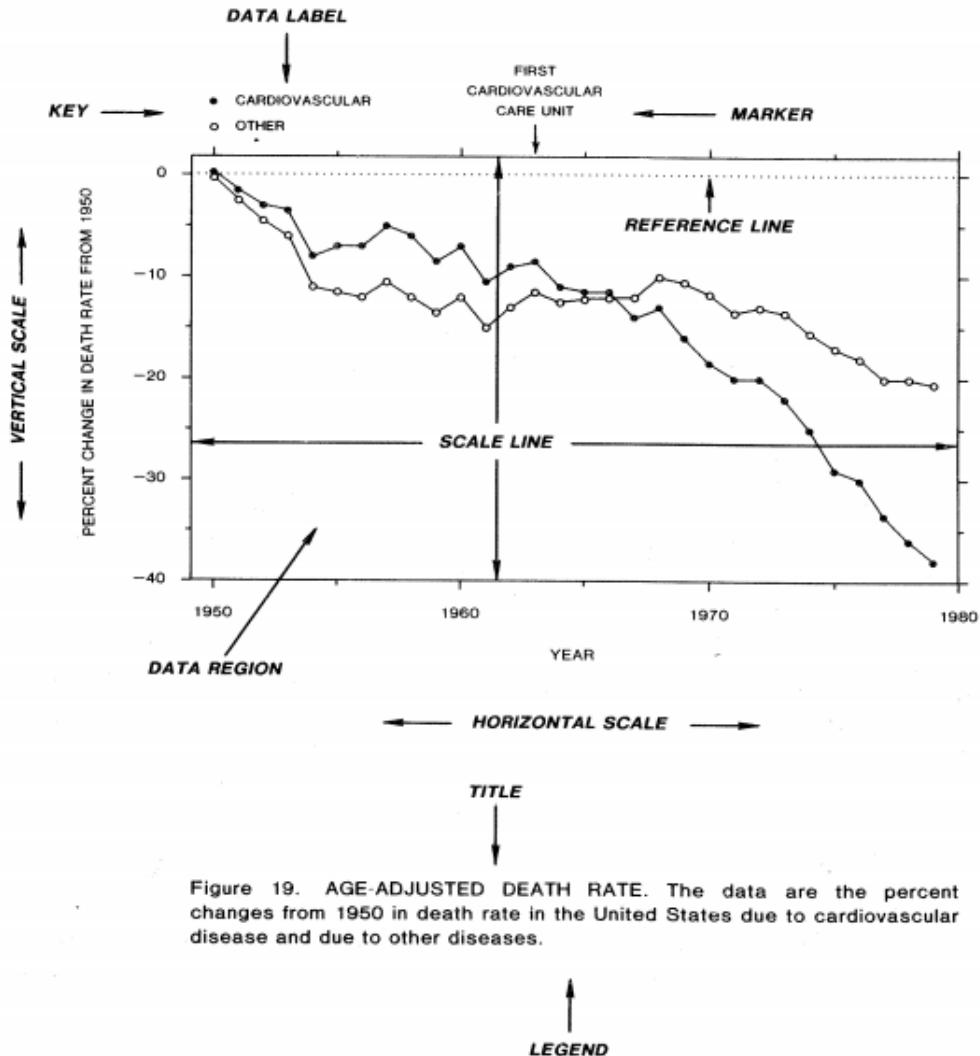


Figure 19. AGE-ADJUSTED DEATH RATE. The data are the percent changes from 1950 in death rate in the United States due to cardiovascular disease and due to other diseases.

Figure 3.8: Partial taxonomy of diagram elements according to Cleveland [Cle94]. Note that this taxonomy of visualization components only considers graphic objects; other aspects of visualization, such as graphic relations, are not covered by it.

This, however, does not mean that any sort of universal consensus exists on the idea that it

is possible to derive systematic accounts of meaning in graphic representations. For instance, in the field of cartography, MacEarchen summarizes some of these critiques in [Mac95]: “borrowing from postmodern thinking, several authors [...] have pointed to the inherent subjectivity in, and rhetorical content of, maps. [...] This perspective suggests that maps are as much a reflection of (or metaphor for) the culture that produces them as they are a representation of a section of the earth or activities upon it. The contention of these authors seems to be that cartography is neither objective nor a science. [...] The appropriate analytical methods, then, should be modeled after literary criticism rather than after experimental methods used in the ‘hard’ sciences”. To these claims, Engelhardt conversely responds that they “are not in contradiction with the notion that meaning in graphics involves phenomena with various systematic tendencies. Regardless of the degree to which such systematic tendencies or systematic principles may be driven by culture and context, they do seem to play important roles in the creation and interpretation of graphics. In summary, I believe that adopting the notion of ‘meaning as a social construct’ does not necessarily entail that one has to deny the role of systematic principles in graphic representations” [Eng02]. Most authors seem to align with the latter notion, many of which have been listed at the beginning of this section.

#### 3.4.2 Visualization Metalanguages

The idea of systematically formalizing diagram elements and their semantics into a set of general principles appears to cohere with current research in visualization semantics, as described in previous section. Moreover, in regards to modern computing-related fields, many problems in vision, such as object recognition, video analysis, gesture recognition, and multimedia retrieval can be viewed as relating symbolic terms to visual information by utilizing general principles deriving formal concepts from the syntactic and semantic structure of natural language [Tow04]. These general principles may also be referred to as the *metalanguage* for discussing diagrams [Ric84], and is the term that will be used in this thesis. The terms making up such a metalanguage are universal in the sense that they extend across cultures and across the whole spectrum of graphic representations [Eng02]. The vocabulary contained in the knowledge base developed for this thesis thus stems from a categorical adoption of the agreements found within these universal metalanguages. We will take their principles at face value in order to map their components into a formal vocabulary, and no further examination on their legitimacy will be undertaken.

Suggested metalanguages of diagrammatic representations found throughout the literature are often structured following a conceptual hierarchical approach, as introduced in section 3.1.2. Early work by Morris in the 1930s ([Mor38], cited in [Ric84]) introduced the notion of graphic objects (which he referred to as ‘signs’ in a broader context not constrained to visualization) to be studied in terms of three different categories of rules according to their level of abstraction: *syntactic*, *semantic*, and *pragmatic* (Figure 3.9). These categories are not mutually exclusive and depend on each other following a hierarchical structuring from less abstract (syntactic rules) to more abstract, higher-level (pragmatic rules): “Syntactical rules determine the sign relations between sign vehicles; semantic rules correlate sign vehicles with other objects;

pragmatic rules state the conditions in the interpreters under which the sign vehicle is a sign" [Ric84]. These three levels of analysis are then considered by Richards in his thesis "Diagrammatics" to build one of the first metalanguages describing diagrams [Ric84]. On the bottom of the hierarchy, the syntactic level (mode of *organization*) deals with the way in which significant elements relate to each other contributing to the relational meaning of the diagram. One level of abstraction above, the semantic category (mode of *depiction*) considers how the graphic image relates to the various elements perceived in the content model. Modes of depiction may range from *figurative* to *non-figurative*. A non-figurative model of depiction "represents the extreme of schematization where only the most generalizable qualities are suggested" whereas a figurative model of depiction if not as schematized, relying on more evident metaphors to convey meaning. Finally, the pragmatic level (mode of *correspondence*) is concerned with "how the diagrammatic elements relate to what is we take the diagram to represent". The mode of correspondence of graphic objects ranges from *literal* (when there is a high level of similarity between the graphic object and what it represents) to *non-literal* (the graphic object and what it represents are only metaphorically related).

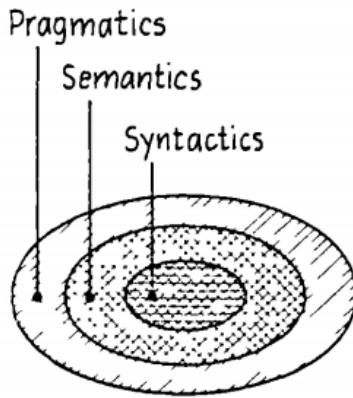


Figure 3.9: Rule categories of signs according to Morris [Mor38, Ric84].

The framework of Richards is criticized by Engelhardt as not being concerned with discussing or unifying existing graphic theories [Eng02]. On the other hand, Engelhardt's work aims to "map out how the various proposed terminologies could be related to each other" in order to come up with a unifying framework of visualization stemming from the consensus that can be obtained from the relevant literature. Therefore, his work, being a unifying, comprehensive approach to general principles of visual language, is expressly applicable to the aims of this thesis, and thus the majority of the resources described by the bottom-most two levels of the suggested axiomatic visualization hierarchy, introduced in Section 3.5.2, are derived from it by formalizing them in the form of ontological resources. The terms that comprise these formalizations will however be divided into different conceptual levels instead of being grouped together under the same namespace, since, as previously discussed, most current research in visualization agrees that visual terms form a hierarchy of conceptual levels that ought to be explicitly characterized. Therefore, Engelhardt's systematic analysis of visualization is to be

formalized, for the most part, in the form of two conceptually ranked structured ontologies, namely, (1) an Upper Visualization Ontology (UVO) grounding concepts related to visual primitives void of any higher level semantics, and (2) a Visualization Domain Ontology (VDO) that formally underpins the elements that make up the composition of a diagram. Domain-, task-, and system- specific vocabularies may then be grounded on these two upper ontologies in order to support full-fledged applications operating on visualization-related knowledge. Each of these layers will be thoroughly described in the following sections.

#### 3.4.3 Existing Ontologies for Visualization

Before delving into the design of the suggested knowledge base, let us first examine existing ontologies in this field. Some authors have already worked towards formalizing knowledge in the field of visualization by means of ontologies. These efforts include VISO [PV13] and the Statistical Graph Ontology (SGO) [DFVR10]. VISO is the most comprehensive attempt at developing a unifying ontology of visualization knowledge that we have found in the literature. It is composed of seven distinct parts, known as the *Graphic*, *Data*, *Activity*, *System*, *User*, *Domain*, and *Facts* modules, respectively. The *Graphic* module formalizes graphic relations and representations, such as graphic attributes. The *Data* module characterizes data variables and structures. Human aspects of visualization are expressed in the *Activity* module. The *System*, *User*, and *Domain* modules allow for describing the visualization context and domain-specific facts. Finally, the *Facts* module formalizes constraints and rankings between resources described in the visualization literature. Of these, only the *Graphic* and *Data* modules have been implemented to a satisfactory degree, whereas the status of the reminder modules has been left to very early design stages. Moreover, VISO was mainly designed with the idea of supporting software tools for visualizing RDF vocabularies, this is, ontology-driven visualization of link diagrams (since RDF vocabularies can be interpreted as link diagrams themselves, see section 3.2.2), and therefore in its current state is not suitable for the description of statistical charts. Nevertheless, some of its resources can be reused for our needs, especially those of the *Graphic* module.

Conversely, SGO [DFVR10] is an ontology supporting a number of statistical chart domains, concentrating mostly on multi-variable line graphs. It provides classes, such as `LineGraph` or `HorizontalBarGraph`, meant to be instantiated in order to formally describe certain types of statistical charts commonly published online. These resources, which primarily underpin visualization objects, are enhanced by external ontologies such as the Basic Relation Ontology (BRO), supporting primitive relations, such as the `hasPart` property allowing composition of objects; and the Statistical Graph iGraph ontology, which describes resources which “do not carry a strong ontological commitment *per-se*, but that are required for specific tasks at the application level”. The Statistical Graph iGraph ontology is thus application-coupled, employed by the iGraph-LITE tool already discussed in section 2.5.4. The underlying data model of a given diagram is formally given by external domain ontologies, such as the *Time-Interval-Primitive* ontology for time and date information. The authors employed this domain ontology to underpin necessary domain knowledge in time charts, which were used for the evaluation of iGraph-LITE. It includes entities covering time intervals (Second, Minute,

etc.) and date (Monday, Year, etc.), as well as properties like `hasPart` (different from the homonymous property defined in BRO) that relate its instances and properties expressing time relations such as `precedes` and `immediatelyPrecedes`.

SGO was developed with the goal of “enabling a multiplicity of applications including exchanging information between statistical agencies in the world, providing accessibility to blind and visually-impaired people, and allowing universal access by portable media such as phones and Personal Assistant Devices” [DFVR10] with special emphasis on providing a scalable and distributed approach to diagram semantics. SGO is the only ontology supporting diagram semantics with a focus on accessibility we have come across in the literature. Unfortunately, we have not been able to find an online repository, copy, or in-depth documentation of SGO, since the endpoints where the authors had originally made it available are no longer operative. Moreover, SGO was developed from a top-down viewpoint analysis to visualization domains constrained to statistical diagrams. Therefore, it does not take into account the atomic graphic elements that composite, higher-level diagram objects are made up of. As it will be discussed in next section, a layered approach to ontology engineering grounded on a set of formal axiomatic primitives is the preferred approach to designing a flexible, reusable, and maintainable knowledge base, and thus effective ontologies need to be designed from the bottom (visual primitives) up (higher-level semantics). The next section outlines the three-layered approach (TLA) ontology design pattern and how we employed a complementary outlook in designing the set of visualization ontologies which are then employed to semantically enhance diagrams allowing them to be operable by Accessible Universal Diagrams through Dialogue (AUDiaL), our NLI prototype.

## 3.5 A Hierarchy of Ontologies for Visualization

The remaining of this chapter describes our suggested approach to modelling visualization knowledge in a formal, axiomatic manner through the use of hierarchically structured ontologies. We will first introduce current hierarchical models in ontology engineering. Afterwards, the suggested semantic model underpinning the visualization domain is thoroughly described.

### 3.5.1 Layered Ontology Design

Hierarchically structuring visualization-related concepts conforms to the arrangement of graphical information into conceptual levels of varying degrees of abstraction that was already discussed on Section 3.1.2 and Section 3.4.2. This structuring not only seems appropriate to formalize visualization domains, but also aids with enabling visualization ontologies to be shared in multiple interactive visual analytic systems and other tools working with information visualization. Moreover, a layer-based approach to ontologies provides the benefit of not falling into under-specification pitfalls for their use by AT [DFVR10]. It is however necessary not to overspecialize the models as to enable their reuse by other potential users besides Visually Impaired Persons (VIPs). Therefore, the layers making up the hierarchy need to be carefully engineered to a proper level of semantic granularity. Ontology designers may overcome these issues by employing a Three Layer Approach (TLA) [DVR07]. A layered approach

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

to ontology design, such as TLA, stimulates maximal reuse of domain ontologies, among many other benefits. It does so by separating taxonomically organized domain terminology from disjointness, complex expressions that define a world view, and application specific requirements that impose a specific data model [DVR07]. Our hierarchy of ontologies for visualization, described in Section 3.5.2, draws on TLA's idea, and thus it will be outlined it in the following paragraphs.

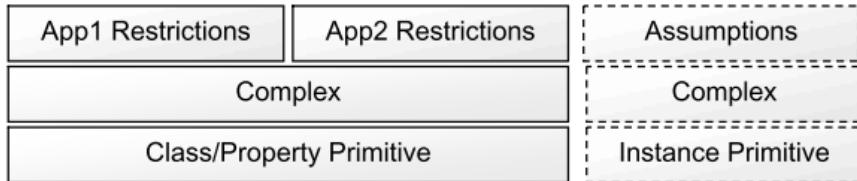


Figure 3.10: Three Layer Pattern of ontology engineering for classes and properties (solid boxes) and instances (dashed boxes), known as TLA [DVR07].

The three-layer ontology design pattern arranges formal resources into three hierarchical conceptual levels (Figure 3.10): the *class/property taxonomic* layer, the *complex* layer, and the *restrictions* layer [DVR07]. The bottom (*class/property* and *instance primitive*) layer is made up of ontological resources (entities and properties) that declare the domain terminology and relations in the form of OWL entities annotated with a human readable label (through the `rdfs:label` property) and a longer but concise human-readable definition (given via `rdfs:comment` property occurrences). Resources on the primitive layer are constrained to having exactly one parent (hence being primitive entities, properties, or instances) as to follow current ontology normalization recommendations, outlined in [Rec03]. Entities are however not required to be disjoint, in order to avoid committing to excessively constrained semantics that might lead to inconsistencies when adding additional domain knowledge to the ontology.

On the next level of the hierarchy, the *complex* layer “refines the primitive layer by imposing restrictions such as necessary or necessary and sufficient conditions beyond the asserted subsumption” [DVR07]. For instance, OWL resources of greater expressivity (see Section 3.2.4) may be employed to augment the domain ontology with new knowledge that can lead to meaningful inferences, such as disjunction, transitivity and reflexivity of properties, etc. Lastly, the “application” layer imposes client-specific constraints to the domain that can be used for document validation and application interoperability. For instance, application-specific classes that adhere to the domain view that the application understands can be specified, and can be automatically validated through a semantic reasoner. This three-layered approach is also applied to class instances and property occurrences. The *instance primitive* layer adds assertions for these resource types (i.e. instantiates a domain) so a flexible model to annotate publicly available data may be employed without need of commitment to more complex logical restrictions [DVR07]. The *instance complex* layer adds axioms to denote necessary relations between instances, using more expressive resources of the OWL language in a similar manner to how it is achieved in the *class/property complex* layer. On the topmost level, the *assumptions* layer contains logical axioms which are needed to enforce a specific data model.

Each layer in TLA therefore increases the available expressivity of the resources contained therein in an analogous manner to the formulation of more complex descriptions from simpler ones in description logics. These separation of the ontological commitment into levels allow users to choose from different specifications of knowledge according to whether they want to commit to either a) a non-disjoint primitive tree formally underpinning domain resources b) a logically enriched tree containing additional domain knowledge, or c) an application-constrained data model. As stated before, this approach not only stimulates knowledge reuse, but also broadens the potential use cases of developed ontologies in different applications, and enhances maintainability and evolution of formal knowledge bases. These features make a layered approach similar to TLA ideal for its use in describing the semantics diagrams for being queried by visualization and navigation software tools, as semantically-enhanced diagrams are potential data models not only in assistive technology, but also in education, journalism, robotics, and a myriad of other spheres. The developed hierarchical ontologies that will be discussed in the rest of this chapter (Figure 3.11) are thus divided into several layers of successively increasing semantic commitment constrained to the world view of diagrams.

### 3.5.2 Five-layer Ontology Pattern for Visualization

The suggested model of formalized visualization semantics is structured hierarchically in five levels as depicted in Figure 3.11: first, an upper ontology, known as the Upper Visualization Ontology (UVO), which is enhanced with visualization semantics by the Visualization Domain Ontology (VDO). Data itself beyond the diagrammatic realm is underpinned by the Data Domain Ontology (DDO) along with additional domain ontologies added to the knowledge base. Tasks supported by visualization are described in the Visualization Tasks Ontology (VTO). On the highest level on the hierarchy, the needs of specific users and systems may be characterized by specialized ontologies. In the following, each of these layers will be characterized in detail.

#### Visualization Ontologies

The foundational layer of the hierarchy is known as the Upper Visualization Ontology (UVO). Deriving a parallel world view to that of descriptive linguistics, visualization semantics are built on top of an inventory of available vocabulary, known as lexicon. The lexicon of visualization is likewise comprised of what could be deemed as ‘nouns’ (Graphic Objects and Graphic Spaces) and ‘adjectives’ (Visual Attributes and Visual Layers) that provide additional visual information about Graphic Objects and Graphic Spaces. On the next level of the conceptual hierarchy lays the Visualization Domain Ontology (VDO). This level underpins properties of the visual primitives defined in the UVO that enrich them with the semantic knowledge necessary for the correct interpretation of the diagram. Carrying on with the likeliness of visual and linguistic structures we could portray VDO as including ‘verbs’ (Graphic Relations) semantically linking graphic elements together and syntactic knowledge about the roles of each resource taking part in a graphic relation.

UVO and VDO thus describe the formal vocabulary on which domain-dependent visualization terms rely for their graphical outline on two-dimensional displays. They therefore are not

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

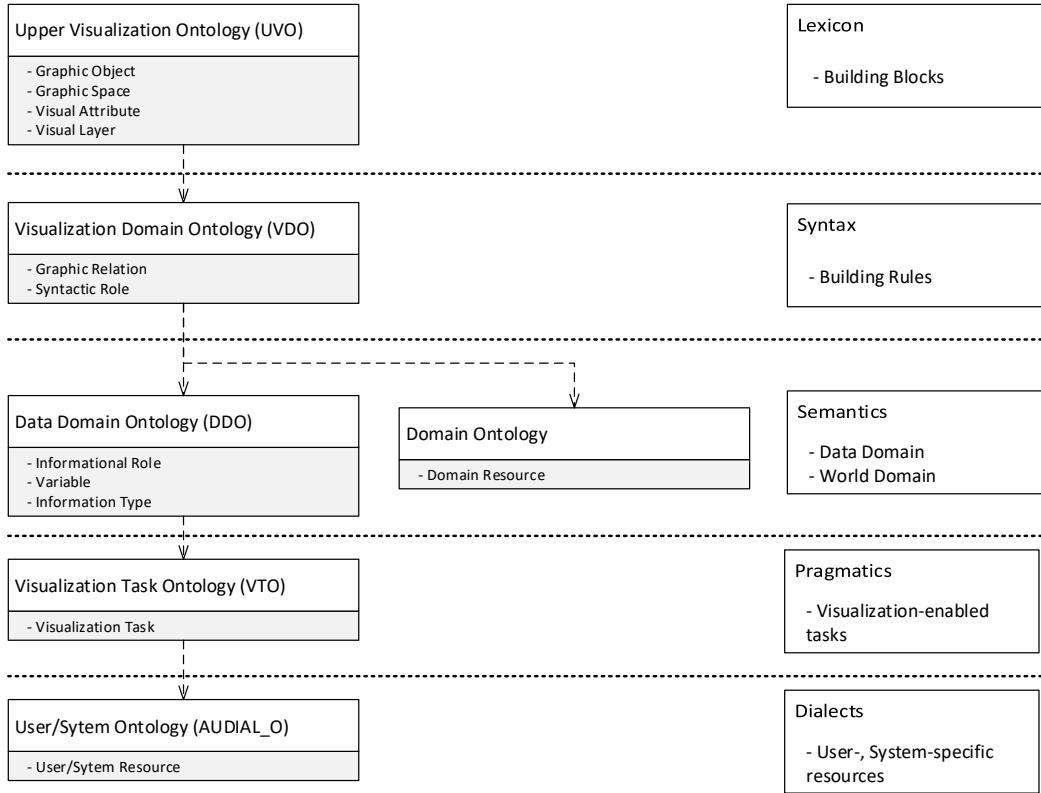


Figure 3.11: Hierarchical five-layer ontology pattern for visualization

concerned about underpinning symbolic representations of the observations that graphics depict. This is to say that the UVO and VDO describe *visualization* itself, not the latent *data* conveyed by the diagram nor higher level concepts grounded on graphic objects and relationships. On the other hand, most approaches to formalization of diagram semantics employ the so-called tabular data model [VP] as the underlying characterization of the diagram. This model consists on row-by-column arrays of numerical observations on a set of variables [Wil05], and is an appropriate and the most straightforward way to model the numerical information to be displayed by a graphics system when the graphically displayed information stems from a previously-known data source. However, given that this thesis is concerned with allowing VIPs to access diagrams from arbitrary sources whose underlying tabular data models may not be available, we need to ground the formal terms of the foundational ontology in the readily perceptible graphical elements of the graphic and properties thereof. Moreover, serializing data from the visualization domain into its raw, tabular form may result in information loss, since the diagram's author may have chosen to display data in a certain visual arrangement in order to convey a specific communicative purpose.

The UVO and VDO therefore take on the task of describing the formal, axiomatic underpinnings

of visually perceptible primitives serving as *a priori* knowledge for object recognition systems (see Section 3.3), and together can be interpreted as the primitive layer (c.f. Section 3.5.1) of visualization-related entities and relations. At this initial level, these visual primitives entail the perceptual manifestations of semantics which are completely void of domain-, task-, or application-specific knowledge. As a basic example, a certain graphic element on a diagram could be described by the UVO as being an instance, `uvo:EGO1`, of the `uvo:Elementary_Graphic_Object` class (`uvo` will be the preferred prefix of resources belonging to the UVO throughout this thesis). This instance could in turn be the subject of `uvo:HVA1`, an occurrence of the `uvo:has_color_named` object property, whose object would be the Literal `uvo:Red`. Further property occurrences may formally assign different visual characteristics to this element, such as having a rectangular shape and a given height and width measured in pixels (see Section 3.6.9 for more information on visual attributes). However, none of these resources contribute domain-specific semantics to the graphic object at hand. For instance, the Graphic Object instance could be a metric bar representing car sales or a surface locator in a map symbolizing a country. This information is however missing from what the UVO and VDO are able to express, and thus resources from a more semantically constrained knowledge base ought to be embedded in the formal description of this graphic. Figure 3.12 highlights the potential roles of UVO and VDO within a general automatic image understanding processing pipeline.

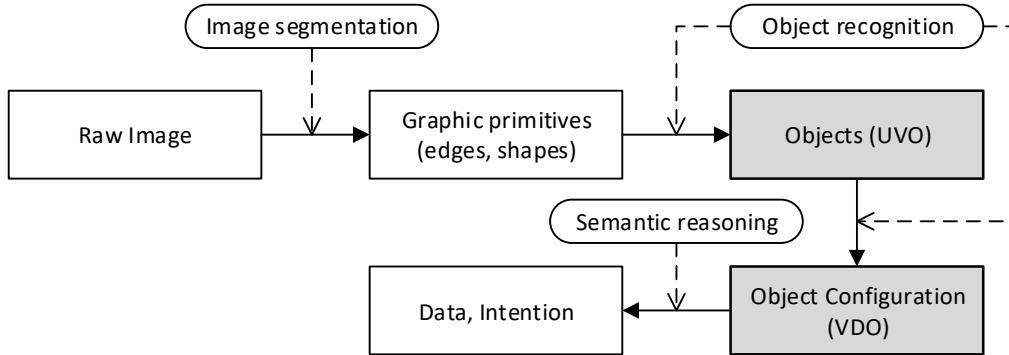


Figure 3.12: Automatic diagram understanding processing pipeline. UVO and VDO may be employed to support object recognition mechanisms by providing *a priori* domain knowledge on visualization (image adapted from [KJ18b]).

The semantic resources described by these two first levels of the conceptual hierarchy are not meant to include the whole range of visually perceptible objects and diagram geometries, as such an undertaking would be hopeless due to the infinite nature of the subject at hand. They are nevertheless meant to act as the sufficient axiomatic underpinnings for their operation by client applications. Town [Tow04] notes how the criteria for determining perpetually significant visual properties depend on the objective and prior knowledge of the observer, and how such criteria can be ranked and further divided into those which are necessary for the

object or action be recognized and those which are simply contingent to the task. Therefore, a fully comprehensive semantic base covering all aspects of visualization not only is unfeasible to engineer, but is also rendered impractical for our goal. Moreover, given the fact that the domain of this thesis is constrained to statistical charts, we have not experimented with the semantic enrichment of maps, link diagrams, and other diagram types. The knowledge bases outlined in this chapter are therefore solely contemplated to meet the demands for the semantic underpinning of two-dimensional, static statistical charts (including time charts).

#### Frege's principle

The UVO and VDO layers are together built on the basis of the so-called principle of compositionality of meaning, also known as *Frege's principle* [Eng02] i.e. a recursive definition of graphic syntax in which a collection of perceivable objects arranged in a given structure in two-dimensional space may function as a single graphic object at a higher level of abstraction (see also Section 3.1.2 and Section 3.4.2). The syntax of diagrams under Frege's principle adheres to the following rules:

1. A *Graphic Representation* (diagram) is a *Graphic Object*
2. A *Graphic Object* may be:
  - a) An *Elementary Graphic Object*
  - b) A *Composite Graphic Object*, made up of:
    - i. A *Graphic Space* that is occupied by it
    - ii. A set of *Graphic Objects*, which are contained within that *Graphic Space*, and
    - iii. A set of *Graphic Relations* in which these *Graphic Objects* are involved

Therefore, a diagram is first and foremost regarded as a Composite Graphic Object enclosed within a Graphic Space e.g. a sheet of paper or a computer screen. Moreover, Composite Graphic Objects contain a number of constituent Graphic Objects taking part in Graphic Relations between each other and the Graphic Space that they occupy. Therefore, the syntactic decomposition of the formal resources making up the diagram applies recursively until its graphic primitives are reached. This syntactic structuring of the vocabulary of diagrams aligns perfectly with the notion of a hierarchy of visual semantics as well as with the principles of Gestalt theory already discussed. Moreover, as we will discuss in Section 3.6, it also allows us to characterize the vocabulary as a light-weight ontology (c.f. Section 3.3) in a natural, straightforward manner by describing Graphic Objects as OWL classes and Graphic Relations as OWL classes and properties.

The syntactical decomposition of graphic objects into a graphic space, sub-objects, and their graphic relations is also known as the *recursive nature of graphics*, and is extensively leveraged by UVO and its bottom ontologies. This principle is adopted with different names

by the visualization ontologies outlined in this document. For example, Communicative Images (Section 2.6) define the logical structure of world fragments by means of ontology-supported aggregate-component relationships between resources [KO08a]. Similarly, SGO, already discussed in Section 3.4.3, infers that “semantically annotated parts are themselves semantically annotated” by means of the property chain inclusion axiom of OWL, asserting that the function composition of its `hasPart` and `hasTag` properties is a subproperty of `hasTag`.

### Data and Domain Ontologies

The topmost levels of the formal hierarchy discussed so far only characterize resources confined to the realm of data visualization. It is evident, however, that the data depicted by its graphical representation are ultimately what end users and applications are most concerned with. Therefore, the *latent numerical data* of the diagram needs to be embedded within its semantic formal counterpart if client applications are to action it. This is what the next level of the ontological hierarchy achieves. The Data Domain Ontology (DDO) underpins the raw tabular structure of the diagram information by means of ontological resources, such as variable names, informational roles of graphic objects, and information types. In addition, domain semantics of the data not necessarily depicted in the graphic may be necessary for its proper comprehension relative to the perspective of the viewer; such external facts may be included in the knowledge base by means of object properties connecting resources from external domain ontologies with the resources described in the visualization ontologies of the hierarchy, as described in Section 4.3.2. This approach parallels that of Communicative Images, in which individuals appearing on a photograph are abstracted by means of suitable domain ontologies providing vocabulary and background knowledge about them [Ošl09].

Owing to the great effort in recent years in developing large-scale domain ontologies, especially in the fields of medical and biological sciences [MB03], it is becoming increasingly likely to find specialized domain ontologies that cover the prior knowledge necessary in order to answer commonsense reasoning related questions [WSL<sup>+</sup>16]. Therefore, the inclusion of domain ontologies within the hierarchy makes answering questions such as “how many people live in the capital of Austria?” possible, even if demographic knowledge about Austria is missing from the diagram and the only relevant piece of information to the query available therein is that there exists a Graphic Object label with associated text “Vienna”. The knowledge that “Vienna” and “the capital of Austria” are semantically equivalent may then be derived from an ontology underpinning Austrian administrative divisions, which allows the query to be reformulated as “how many people live in Vienna”. From this moment on all required information for solving the question is found within the semantically-enhanced diagram, and the query may be resolved.

### Visualization Tasks Ontology

Ranked next in the semantic hierarchy after data and domain ontologies comes the Visualization Tasks Ontology (VTO). This ontology describes the most relevant vocabulary of user tasks that can be performed on visually depicted information in the form of diagrams. The

vocabulary of tasks depends on the data depicted by the diagram more than on how the data is diagrammatically depicted [CL98]. Moreover, a focus on analytic primacy instead of on representational primacy, this is, concentrating on mapping visualization resources to user analytic goals, greatly benefits end-users of information visualization by not relying on the author's own insights and expressive capability within a particular tool thereby increasing the value and utility of information visualization [AES05]. For these reasons, VTO lays on top of DDO in the visualization hierarchy. Some authors bring attention to the necessity of approaching tasks in an adaptive goal oriented manner in order to come closer to capturing image semantics [Tow04]. User questions as part of analytic activity in diagrams have been shown to range from broad high-level goals that may be decomposed into more specific, low-level queries [AES05]. Therefore, we have laid our focus on describing low-level analytical tasks covering the range of user queries when working with diagrammatically displayed information. It is then the duty of client applications (e.g. AUDiaL, described in Chapter 5) or end users to combine these primitive tasks in order to derive answers to questions of a higher conceptual level.

Moreover, given this thesis' focus of enabling VIPs access diagrams autonomously, wayfinding tasks enabling VIPs explore the diagram in an exploratory manner, with no specific task in mind other than its overall examination, have also been considered. The reasons for the inclusion of these specific navigational tasks in the ontology will be discussed in their respective entries of Section 3.9.6 and, later, in Section 6.3. The simple tasks contained in VTO may then be combined by client applications through composition and superimposition mechanisms. Composition-based high-level tasks (such as, generally, the main intention of the diagram's author) take the order of their sub-tasks into account and bear a dominant motive [CL98]. On the other hand, superimposition-based tasks, the underlying primitive tasks are all related to the same diagram but their intentions do not interfere [CL99]. Note that the resources outlined in VTO are independent from the bottom ontologies in that VTO contains no knowledge regarding the mechanisms for carrying out tasks by actioning the resources of UVO, VDO, DDO, or any domain ontology. VTO could be regarded as a taxonomy of low-level tasks that may be performed in diagrams by both sighted and VIPs. However, the execution and handling of the tasks themselves is out of the scope of what ontologies can express, this being the task of software applications consuming and modifying the knowledge base.

#### User and System Ontologies

Lastly, the bottom-most level of the hierarchy consists of user- and system-specific resources for handling the knowledge contained in the top ontologies. The knowledge base underpinning a diagram's semantics may be accessed in radically contrasting fashion depending on the end user's needs. In the context of this thesis, AUDiaL (Chapter 5) is concerned with enabling VIPs access diagrams in an autonomous manner. Therefore, a number of techniques had to be included in order to partially compensate for the cognitive benefits that its users lack, which are outlined in Section 6.7. Interaction mechanisms that require ontological support may thus include their own resources by means of application-specialized ontologies placed at

the bottom of the hierarchy allowing them to effectively consume resources from the bottom ontologies. Examples from the existing literature of the adaptation of user interfaces to specific user and system needs by means of ontologies are discussed in Section 3.10.

### A small example

Let us end this section with an illustrative example of how the different semantic levels of the suggested hierarchy could be involved during the solving process of a wayfinding activity by the user in a Natural Language Interface (NLI) to diagrams. Given a bar chart on Austrian demographics and a wayfinding query, expressed as the following command in natural language by the user: “move from the bar labelled ‘masculine Vienna 2014’ to‘male Vienna 2018’”:

- UV0 characterizes the visual primitives of the diagram, such as the fact that there are two rectangles with certain *visual attributes* (for example, their color, position, and height).
- VDO recognizes each graphic primitive as playing a certain syntactic role. For example, the two relevant rectangles to the query are characterized as being *metric bars* in a *metric space*. Moreover, it underpins that these bars are *labelled* by a number of graphic objects, among other graphic relations. In our example, the two bars are being labelled by graphic objects with associated texts “Vienna”, “2014”, and “2018”.
- DDO describes that there are three *variables* depicted in the diagram; namely, *population*, *years*, and *gender*; and which *informational role* each graphic objects plays therein in order to express these variables.
- An external domain ontology e.g. WordNet, recognizes that the words “*masculine*” and “*male*”, found in the user query, are *synonyms*.
- VTO characterizes the task referred to in this query as a combination of two *filtering analytical tasks* and a *jump navigational task*.
- The system ontology indicates that the user has previously saved information about “Vienna”, such as a personal text annotation, which client applications may choose to display to the user along with the answer (c.f. Section 6.4).

Client applications may then make use of this information or parts thereof in order to execute the high-level task and display the most relevant answer to the user. AUDiaL, described in Chapter 5 is the NLI application prototype that will be employed in this thesis to interface between the user and the ontologies here described. We now proceed to given a detailed account of the most important resources of each ontology in the semantic hierarchy.

## 3.6 Upper Visualization Ontology

This section outlines the main resources contained within the upper ontology for visualization known as the Upper Visualization Ontology (UVO), with the associated namespace `uvo`. The rationale behind the vocabulary included was previously outlined in Section 3.5.2.

### 3.6.1 Graphic Thing (`uvo:Graphic_Thing`)

- **Entity Type:** Class
- **Subclass of:** `owl:Thing`
- **Subclasses:** `uvo:Correspondence`, `uvo:Graphic_Object`, `uvo:Graphic_Relation`, `uvo:Graphic_Space`, `uvo:Image`, `vdo:Informational_Role`, `vdo:Syntactic_Role`, `uvo:Visual_Attribute`

Much in the same way OWL 2 defines a super-class, `owl:Thing`, to refer to all individuals (c.f. section 3.2.4), UVO defines a super-class of visualization-related resources, namely `uvo:Graphic_Thing`. This class has been included in UVO to take place as the root of the taxonomic tree for the visualization vocabulary characterized by the UVO and VDO ontologies, and is thus the super-class of all graphic objects, graphic spaces, graphic relations, graphic roles, and visual attributes described in the remainder of this chapter.

### 3.6.2 Graphic Object (`uvo:Graphic_Object`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** `uvo:Elementary_GO`, `uvo:Composite_GO`

A *Graphic Object* is a perceivable unit that takes part in a graphic representation. These are what Richards calls *significant elements*: any element in a diagram which “at least is capable of having some meaning” [Ric84]. Following Frege’s principle (Section 3.5.2), meaning in graphic representations depends upon the graphic objects it contains as well as the graphic relations that graphic objects are involved in. Images themselves (e.g. an entire scatterplot diagram) are graphic objects, as are its visual primitives, such as each of the individual data points that make up the scatterplot. Furthermore, any meaningful grouping of graphic objects can make up a graphic object of a higher conceptual level. Graphic objects are thus hierarchically structured following the recursive nature of graphics. Therefore, some formal distinction between atomic and divisible graphic objects needs to take place in UVO. This syntactic distinction is obtained by means of two subclasses of the Graphic Object class, namely the *Elementary Graphic Object* and *Composite Graphic Object* classes. This entails that the Graphic Object class should not be directly instantiated i.e. individuals must be

instances of either the Elementary or Composite Graphic Object class. More information about these subclasses is given in the following sections.

Being visually perceptible, graphic objects possess a number of visual variables such as position, size, shape, color, and so on, which may be vital for the proper interpretation of their semantics. These visual variables or aesthetic attributes of graphic objects are described by the `uvo:Visual_Attribute` class and its subclasses. Moreover, being the only resources of UVO that carry visual attributes entails that the class of graphic objects is equivalent to the anonymous class of instances being involved in at least one X coordinate and one Y coordinate property occurrences. This fact is described by means of OWL existential restrictions (see Section 3.2.4). Semantic Annotator for Inkscape (SAI), a semantic image annotator introduced in Chapter 4, automatically enriches graphic objects with structural annotations about their position, simplifying the authoring of graphic objects. More generally, any ontological instance taking part in triples containing visual attributes could also be regarded as a graphic object. However, given that many of these occurrences may be missing from a given image's semantic annotation, we have specified graphic objects just as being equivalent to the class of instances carrying information about their position in order to simplify ontology design.

A point could be made on whether graphic objects appearing in a diagram could be interpreted as class instances of an underlying domain ontology of the specific domains the diagram depicts. For instance, let us consider a bar in bar chart illustrating that the population of Vienna in 2017 was of 1.8 million inhabitants. This bar could be understood as e.g. the Vienna instance of the class `City` of a domain ontology describing Austrian demographics. This instance may then take part in `has_population` and `has_year` property occurrences, in this manner not only describing the individual context of the information originally illustrated by the bar, but also additional contextual knowledge not available in the diagram. However, this approach not only would be an ontological stretch, as pointed out by some authors [DFVR10], but would require a much greater deal of pre-processing of the original diagram before it could be queried by a client application. Therefore, within the hierarchy of ontologies suggested here each graphic object in a diagram must be an instance of any of the subclasses of `uvo:Graphic_Object`. Graphic objects may then be semantically enriched by two complementary means: (a) by associating them to supporting resources of a domain ontology, or (b) by means of the diagrammatic context contained within the image, itself underpinned by resources of the UVO and VDO ontologies here outlined. Both approaches are supported by the natural language interface described in Chapter 5, but the latter may be more easily embedded in a diagram due to its constrained realm and simpler characterization given the image primitives.

### 3.6.3 Elementary Graphic Object (`uvo:Elementary_GO`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Object`
- **Subclasses:** `uvo:Composite_Symbol`

*Elementary graphic objects* are atomic graphic objects in regard to their syntactic interpretation. In other words, an elementary graphic object is such that it may not be further decomposed into meaningful sub-graphic objects. For instance, each bar in a simple bar chart would be regarded as an elementary graphic object. Even if a bar could be seen as being made up of four individual segments that make up the rectangle characterizing its shape, these segments in isolation lack any interpretation, and thus is the bar (e.g. its SVG rectangle element) and not the segments that can be considered an elementary graphic object. On the other hand, if the bar was stacked (i.e. if it contained a number of constituent bars that can be individually interpreted) then it could not be considered as an elementary graphic object, but instead as a composite graphic object. Elementary graphic objects are referred to as “significant elements” in Richard’s thesis, where they are employed to describe “the primary units of analysis” [Ric84].

In SVG diagrams, elementary graphic objects are generally associated with each SVG renderable element, usually constrained to the basic shapes therein, such as those determined by a circle, line, mesh, polygon, text, or image SVG tag. However, this needs not always be the case, and several SVG renderable primitives may be combined to make up a symbolic elementary graphic (see the Composite Symbol class below). The characterization of visual primitives as elementary graphic objects thus depends on the function of the graphic representation in its communicational context [Eng02]. This feature of elementary graphic objects is what Goldsmith named the *separateness* of the unity of a single image [Gol78]: “the degree of separateness is obviously going to vary with the level of discussion appropriate to the intention of the picture: for example, in a landscape the single image [elementary graphic object] might be a ‘cow’, while in a portrait it might be an ‘eye’ or even a ‘pupil’”.

In a dialogue-based system to graphics, graphic objects could be interpreted as either elementary or composite according to the communicative granularity expected by the user. For example, when obtaining a high-level overview of a scatterplot, its cloud of points may be regarded as an elementary graphic object, as the user is only interested about high-level conceptual properties of the information depicted by the scatterplot, such as the correlation between its variables. On the other hand, when sequentially navigating through it, the datapoint cloud needs to be regarded as a composite graphic object, with each individual point being considered an independent elementary graphic object. However, since the framework here outlined relies upon the vector interpretation of diagrammatic visual primitives, AUDiaL (Chapter 5) always assumes each SVG basic shape to be an elementary graphic object instance.

#### 3.6.4 Composite Symbol (`uvo:Composite_Symbol`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Elementary_GO`
- **Subclasses:** None

According to Engelhardt “a *composite symbol* is a graphic object that is composed of a small number of Elementary Graphic Objects (often two) which are arranged in a conventionally fixed arrangement. [...] Most composite symbols are members of a ‘family’, which is characterized by a shared visual vocabulary and a shared compositional grammar” [Eng02]. Some examples of graphic objects adhering to this definition are traffic signs or visual icons in Graphical User Interfaces (GUIs). In other words, composite symbols are Composite Graphic Objects that are regarded as elementary graphic objects because they follow a set of rules shared among composite symbols of the same kind. For instance, in traffic signs a blue circular shape generally means ‘obligation’. This is the first elementary graphic object of the composite symbol. What the driver is obliged to do is depicted as a second elementary graphic object, such as an arrow pointing to the direction the driver must follow. However, each of these elementary graphic objects would be void of meaning without the other one, and thus cannot be considered independently.

In many applications it is however difficult to distinguish between composite graphic objects taking part in object-to-object graphic relations and composite symbols. As a matter of fact, most authors do not include terms similar to the composite symbol in their visualization taxonomies. Therefore, authors of semantically-enhanced diagrams are advised to interpret ensembles of graphic primitives as composite symbols only when its semantic validity is conclusive, erring on the side of caution when in doubt.

### 3.6.5 Composite Graphic Object (`uvo:Composite_GO`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Object`
- **Subclasses:** None

A well-known principle in Gestalt psychology (c.f. Section 3.1.2) states that objects “are not exhaustively described solely on the basis of their constituent parts” and thus “any attempt at recognizing objects based on pre-defined [...] feature sets is inherently limited in its applicability” [Tow04]. It is therefore a necessary requirement in an ontology for visualization for composite objects present in diagrams to be associated to their own ontological instances, as they may be given semantics not applicable to their constituent elementary graphic objects. This is attainable in UVO by means of the *Composite Graphic Object* class. Any Graphic Object which is not a Composite Symbol and is made up of an aggregation of more than one Elementary Graphic Object must be an instance of this class.

Aggregation of elementary graphic objects (and constituent composite graphic objects of a lower conceptual level) into composite graphic objects is realized in UVO by means of two transitive properties: `uvo:is_part_of_graphic_object` and its inverse `uvo:has_graphic_object`. A composite graphic object is said to be made up of a number of constituent graphic objects as given by all resources appearing in occurrences of the `uvo:has_graphic_object` object property where the composite graphic object is the

triple's subject. Moreover, given that these are transitive properties, the constituent elements of a composite graphic object which itself is a constituent graphic object instance of a higher-level composite graphic object are also constituent graphic objects of the higher-level composite graphic object (see Figure 12).

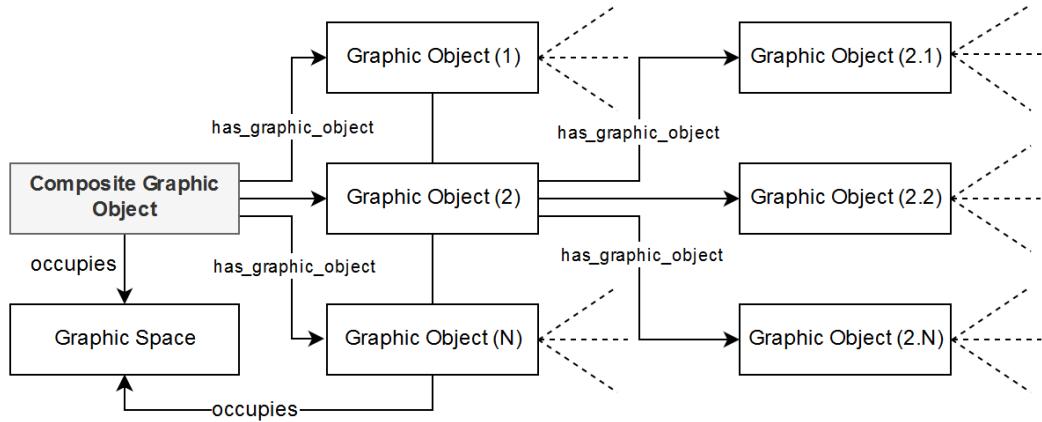


Figure 3.13: Recursive nature of graphics in UVO (figure after [Eng02]).

Some approaches to formal visualization semantics consider some visual properties of graphic objects to also be transitive in the sense that elementary graphic objects share some visual properties of the composite objects they belong to. For instance, the Graphical Ontology used by Communicative Images (Section 2.6.2) defines visual properties such as ambient light and distance as transitive [Ošl09]. However, properties carrying information related to the visual attributes of graphic objects are not considered transitive in UVO, since diagrams may not follow the same physical laws that are observed in photographs. When a number of properties occurring in a composite graphic object instance must also apply to its constituent elements, these new triples may be easily included to the knowledge base by employing Semantic Annotator for Inkscape (SAI), described in Chapter 4.

### 3.6.6 Graphic Space (`uvo:Graphic_Space`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** `uvo:Arbitrary_GS`, `uvo:Meaningful_GS`, `uvo:Metric_Axis_GS`

A *graphic space* is “the substrate of all spatial relations within graphic representations” [Eng02] i.e. the mental construction of one-, two- or three-dimensional spaces that graphic objects occupy. The proper characterization and interpretation of graphic spaces is vital for the analysis of diagrammatically displayed information; some authors state that “the most

important characteristic of a picture is [...] the method of arranging information in space" [Kur96]. Therefore, all graphic object instances in a diagram must theoretically be associated with at least one graphic space instance. A graphic space in which no spatial (i.e. object-to-space) relations take place is known as an Arbitrary Graphic Space (`uvo:Arbitrary_GS`). On the other hand, a graphic space which is occupied by graphic objects taking part in at least one spatial relation is defined as a Meaningful Graphic Space (`uvo:Meaningful_GS`). These two kinds of graphic spaces are implemented in UVO as subclasses of the Graphic Space super-class. Since all graphic spaces are either meaningful or arbitrary, no object may be directly instantiated from the Graphic Space class. In arbitrary graphic spaces, no information is expressed by the positioning of the graphic objects it contains and thus the relative and absolute position of graphic objects has to be ignored by client applications. This kind of graphic space is rather uncommon in diagrams. On the other hand, meaningful graphic spaces carry essential information for the interpretation of a diagram, and will be expanded upon in the following sections.

Graphic spaces could be understood as taking a certain amount of physical space much in the same way graphic objects do, with the top-most graphic space of the diagram occupying its entirety. However, this interpretation would not be semantically coherent with the proposed framework, as graphic spaces themselves are not visible (in words of Wilkinson [Wil05], no *aesthetic attributes* can be linked to them), and therefore they do not possess associated coordinates, shapes, or other visual attributes. For example, the metric graphic space expressed by the combination of two axes in a scatterplot may be interpreted as consisting of the 2D plane extending beyond their limits. Therefore, meaningful graphic spaces in UVO are uniquely identified not by their visual properties such as location or size, but by which graphic object and metric axis instances occupy and define it. For example, the collection of all objects in a world map (e.g. countries) excluding those belonging to different visual layers (e.g. a map legend) would make up the constituent elements defining the integral metric space of the map.

Graphic space instances may be completely disregarded from or not even incorporated to knowledge bases that underpin simple diagrams where they may be inferred from other semantic aspects. For instance, most diagrams contain a main graphic space (where the data objects and axis lay) and perhaps a secondary graphic space containing a legend describing further aspects of the data. In such cases, client applications can directly infer data point values and labels from the existing object-to-object relationships and coordinate properties, since the possibility of inferring wrong information from an irrelevant graphic space instance does not exist, and thus adding graphic space instances would be redundant.

### 3.6.7 Metric Graphic Space (`uvo:Metric_Space_GS`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Meaningful_GS`
- **Subclasses:** `uvo:Composite_Metric_Space_GS`, `uvo:Elementary_Metric_Space_GS`

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

*Metric graphic spaces* are those graphic spaces in which the absolute positioning of graphic objects contained therein is subject to strict interpretation. A formal description of metric space can be found in [Wil05], p.183. A vital consideration regarding metric spaces in diagrams is that a meaningful distance measure between any two points of the metric space can be defined. Therefore, as opposed to other meaningful graphic spaces in which objects' positioning may be altered somehow without changing their underlying semantics, the absolute positioning of graphic objects in a metric graphic space carries information subject to direct interpretation i.e. any change in the absolute positioning of a graphic object would result on it carrying different semantics. For example, the space occupied by a bar in a bar chart is a metric space, since even a small increase or decrease in the bar's size involves changing the numerical information it carries.

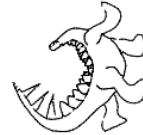
			
Translate	Rotate	Reflect	Dilate
<b>Isometry (distance invariant)</b>			<b>Similarity (shape invariant)</b>
			
Stretch	Shear	Project	Conform
<b>Affine (parallelism invariant)</b>		<b>Projective (straightness invariant)</b>	<b>Conformal (angle invariant)</b>

Table 3.1: Transformations of the plane according to Wilkinson [Wil05]. The class each transformation belongs to and its type of invariance are indicated in bold on the row below.

Metric spaces can be either *Elementary Metric Spaces* or *Composite Metric Spaces*. Elementary metric spaces (`uvo:Elementary_Metric_Space_GS`) are those which cannot be decomposed into a smaller number of graphic spaces without losing interpretability. For example, an *integral metric space*, like the 2-dimensional Euclidean space of geographical maps, is an elementary metric space. On the other hand, a composite metric space (`uvo:Composite_Metric_Space_GS`) is made up of two or more independent graphic spaces, such as the combination of two graphic spaces constructed by the two metric axes in a scatterplot.

Lastly, any metric space may be *distorted*. These are metric spaces in which the order and approximate direction between graphic objects is preserved, but the ratios of spatial directions are not [Eng02]. A metric space may be distorted for several reasons, such as simplifying the presentation of its elements, emphasizing variation or covariation, to make detail visible, or to match its form to theory or reality [Wil05]. Table 3.1 displays a number of categorized distortions (i.e. transformations) that may be applied to a two-dimensional integral metric space. For a detailed description of these transformations, the reader is directed to [Wil05], Chapter 9. Distorted metric spaces are therefore those affected by *affine*, *projective*, or *conformal graphic transformations*.

All graphic objects contained in a distorted metric space, including reference objects such as axes, are equally transfigured by its transformation. Therefore, distortions themselves are not considered in UVO, since all relevant semantics of the graphic (data point values, trends, correlations, statistics, etc.) may be inferred without the need for undoing the transformation. In some cases, the transformation may have been included with the sole purpose of highlighting some aspect of the data which would otherwise be not easily inferred by means of sight. Moreover, the specific transformation parameters are most likely unknown when parsing diagrams from unknown sources *a posteriori*. *Distorted metric spaces* may however be added to the axiomatic representation of a diagram by simply instantiating `uvo:Distorted_Integral_Metric_Space_GS` for distorted integral metric space instances and `uvo:Distorted_Elementary_Metric_Space_GS` for every other distorted metric space instance.

Metric graphic spaces are the only sub-class of meaningful graphic space in UVO, since graphic relations involving object positioning taking part in other kinds of meaningful graphic space comprise relative positioning between graphic objects which allows a certain degree of freedom in their absolute positioning within the graphic space they occupy. Therefore, these graphic relations are better suited to be formalized by means of object-to-object graphic relation instances. It is however important to underline that the coordinate information of graphic objects in other meaningful graphic spaces must not be ignored by client applications, since the relative positions between objects may still carry vital information e.g. for cluster detection.

### 3.6.8 Visual Layer (`uvo:Visual_Layer`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** None

*Visual layers* in graphic representations stem from the mental model of a three-dimensional space derived from its two-dimensional representation. They correspond to a certain depth level in the equivalent mental space of a graphic space, where depth is interpreted as a discrete sequence of perceptually different levels or 'layers'. Visual layers occupy a graphic

space allowing graphic objects inhabiting it to take part in superimposition graphic relations (see below) as well as other kinds of relations in which they are said to be ‘in front of’ or ‘behind’ one another, such as the relationship between a grid and the data points laid ‘on top of’ it in a scatterplot.

For the most part, visual layers need not strictly be included in a knowledge base in order to make diagram semantics actionable by computer applications, since the relevant semantics for user task operations are already expressed by the graphic relations that graphic objects and graphic spaces take part in. Nevertheless, visual layers in UVO can be added to graphic space instances by means of the `uvo:has_visual_layer` object property. In addition, graphic object instances may be associated to one or more visual layers by means of the `uvo:belongs_to_visual_layer` object property. Moreover, the relative spatial ordering between visual layers can be expressed in UVO by means of the `uvo:is_above_visual_layer` and `uvo:is_below_visual_layer` object properties.

#### 3.6.9 Visual Attribute (`uvo:Visual_Attribute`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** `uvo:Area-Fill_VA`, `uvo:Spatial_VA`

*Visual attributes*, also known as *aesthetic attributes* or *retinal variables* in the visualization literature, comprise a subset of the visually perceivable characteristics of the world that can be attributed to individual graphic objects, such as size, color, or shape. Because of their immediate visual salience, visual attributes are arguably the most studied characteristic of diagrammatic representations. Visual attributes are of vital importance in diagrams since they have the ability of encoding quantitative and categorical information and thus they characterize the display method that underpins the semantics of a diagram. The encoded information is decoded by means of elementary graphical perception tasks by sighted readers. For example, variation in position can carry *selection*, *associative*, *ordered*, or *quantitative* variables of data, whereas variation in color may carry *associative* and *selection* variables [Ber11]. Along with high-level cognitive tasks, this combination of perception and cognition is what make diagrams advantageous over other data displays [Cle94]. For an in-depth study on the characteristics and properties of visual attributes with regard to diagram semantics the reader is directed to Bertin’s “Semiology of Graphics” [Ber11], Part 1, Chapter 2, section C (“The retinal variables”).

Visual attributes in diagrams are chosen “in a trade-off between the psychological dimensions they elicit and the types of routines that can be implemented in a rendering system” [Wil05]. Therefore, not all naturally occurring visual attributes are fit for their use in diagrammatic representations of data. The visual attributes in UVO have been included with these constraints in mind stemming from those appearing in the relevant literature. According to Engelhardt [Eng02], visual attributes are most conveniently divided into two groups, namely spatial visual attributes and area-fill visual attributes:

- *Area-fill Visual Attributes* are visual attributes that do not affect the spatial properties of their graphic object when they change. Area-fill visual attributes included in UVO are *color*, *texture*, and *optics* (see Table 3.2).
- *Spatial Visual Attributes* are those visual attributes of a graphic object that, when modified, alter some spatial characteristics of the object. These include the *orientation*, *position*, *shape*, and *size* of graphic objects (see Table 3.3).

When no specific property for a given visual attribute exists, graphic object instances are embedded information about their visual attributes in a knowledge base with the inclusion of `uvo:has_visual_attribute` object property triples. It is important to underline that the constituent graphic objects of a composite graphic object instance may have completely different visual attributes than those of their whole and therefore this property is not transitive.

### 3.6.10 Area-fill Visual Attribute (`uvo:Area-Fill_VA`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Visual_Attribute`
- **Subclasses:** `uvo:Optics_VA`, `uvo:Color_Named_VA`, `uvo:Texture_VA`

*Area-fill visual attributes* in UVO (Table 3.2) are characterized according to Wilkinson's taxonomy [Wil05]. They are divided into three sub-classes: *color*, *texture*, and *optics*. These subclasses are briefly outlined next.

*Color* is arguably the most common area-fill attribute employed by diagram authors, which generally make use of it for either decorative purposes or to represent nominal information. Color is however not suitable for codifying numerical or ordinal information [Ber11]. Computer screens display colors within the complete visible spectrum by mixing three primary colors (channels) with different brightness intensities: red, green, and blue. Accordingly, color in so-called RGB formatted images is stored by setting the intensity values for each pixel's channels to a brightness value of between 0 and 255. A specific shade is thus uniquely designated by its RGB triplet, most commonly as an equivalent hexadecimal value ranging from #000000 (black) to #FFFFFF (white). Graphic objects in UVO are associated color by means of `uvo:has_color` datatype property occurrences, whose triple objects are strings formatted in such manner. Alternatively, colors expressed in the HSV color model (skipped here for brevity) may also be expressed by means of the `uvo:has_brightness`, `uvo:has_hue`, and `uvo:has_saturation` datatype properties. Color gradients are underpinned by discretizing the graphic object possessing the gradient into a number of elementary graphic objects in a proportional division graphic relation with their parent, each of which is then associated with a distinct occurrence of `uvo:has_color`.

For the purposes of natural language-enabled access to diagrams, however, color expressed in a formal color model does not meet the high-level vocabulary of the user, which is likely to

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

express different color shade bins by common names such as 'blue' or 'pink'. Therefore, named colors may also be associated to graphic objects by means of the `uvo:has_color_named` object property, whose domain entails individuals of the `uvo:Color_Named_VA` class such as `uvo:Black` or `uvo:Yellow`. Lastly, the Semantic Color Model (SCM) [KO08b] is an intermediate alternative between both approaches to color interpretation previously discussed. In SCM, colors may automatically be verbalized in a few words by expressing them as the mixture of their two most salient primary colors e.g. a 'red-green shade'. This interpretation is however not suitable for its inclusion in UVO, as it must be inferred by client applications from RGB color model values.

Color	Texture - Granularity	Texture - Pattern	Texture - Orientation	Optics - Transparency	Optics - Blur

Table 3.2: Area-fill visual attributes characterized in UVO.

*Texture* is defined as the "number of separable marks contained in a unitary area" [Ber11]. Texture visual attributes are *pattern*, *granularity*, and *orientation* [Wil05]. Pattern is used in diagrams to express nominal variables, much in the same way of color. Note however that this use is discouraged by Bertin [Ber11], who recommends using texture in diagrams for encoding ordinal information only. Patterns are non-exhaustively characterized in UVO by their common given names, implemented as individuals of the `uvo:Texture_Pattern_Named_VA` class such as `uvo:Texture_Grid_Pattern_VA`. Note that the specific characteristics of the pattern are not relevant for the interpretation of the graphic, since labelling graphic relations may rely on the common sharing of the same pattern between graphic objects only, and are therefore not included in the knowledge base. Pattern granularity may however be employed in diagrams to convey ordinal variables [Ber11], and as such it has been included in UVO. It partially characterizes distinct pattern granularities (e.g. *coarse-* and *fine-grained*) by means of the available individuals belonging to the `uvo:Texture_Granularity_Named_VA` class. The last aspect of texture is orientation. As it was in the case of color, orientation may be expressed in UVO at two opposite levels of abstraction. On the one hand, it may be precisely expressed in degrees  $\in [0, 180]$  from the horizontal position relative to the viewer by means of the `uvo:has_orientation` datatype property. On the other hand, it may be expressed at a higher conceptual level by means of `uvo:Texture_Orientation_Named_VA` class individuals linked to graphic objects with `uvo:has_visual_attribute` object property

occurrences.

The last category of area-fill visual attributes is *optics*, which includes the sub-categories of *transparency* and *blur*. These attributes are less commonly used than the other area-fill attributes discussed so far, and most authors do not even consider them in their taxonomies. Moreover, transparency and color may actually be synonymous in many contexts, especially when no superimposition between graphic objects takes place. These two categories thus are of minor relevance in UVO. Transparency includes three disjoint individuals: *opaque*, *translucent*, and *transparent*, of the `uvo:Transparency_Named_VA` class, whereas blur is considered through the `uvo:Blur_Named_VA` class, whose individuals represent non-exhaustive categorical blur intensities ranging from *no blur* to *intense blur*. As usual, graphic object instances are assigned optics by means of the `uvo:has_visual_attribute` object property.

### 3.6.11 Spatial Visual Attribute (`uvo:Spatial_VA`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Visual_Attribute`
- **Subclasses:** `uvo:Orientation_Named_VA`, `uvo:Position_Named_VA`, `uvo:Size_Named_VA`, `uvo:Shape_VA`

According to Engelhardt [Eng02], *spatial visual attributes* include the sub-categories of *position*, *size*, *shape*, and *orientation*. These attributes (Table 3.3) are characterized in UVO by instantiating sub-classes of the `uvo:Spatial_VA` class, which we will describe in the following. Given that the underlying graphic primitives making up the diagrams considered in this thesis are stored in vector format, spatial visual attributes thereof may be fetched or derived from the corresponding SVG attributes of their constituent elements placed on the SVG canvas space. Chapter 4, which details SAI i.e. the prescribed authoring tool for semantically-enhanced diagrams, describes the mapping of SVG attributes to ontological resources, whereas this section focuses on describing the formal resources themselves.

The absolute *position* of graphic objects on the two-dimensional plane is underpinned by the `uvo:hasXCoordinate` and `uvo:hasYCoordinate` datatype property occurrences they take part of. These properties have their point of origin (zero value) on the top left corner of the visible space following the coordinate system of the SVG image format. Their value is measured in pixels from the point of origin, with positive values being to the right for the X coordinate and to the bottom for the Y coordinate. Note that no such thing as a ‘user unit’ may be expressed in UVO. Scalable positions must therefore be mapped to absolute positions before being stored in the knowledge base. The position of an Elementary Graphic Object is given by the coordinates of the point laying in the ‘center’ of the object, whose calculation may differ depending on the shape. The center for simple shapes such as rectangles is easily located, whereas for shapes such as paths or complex polygons some *ad-hoc* compromise needs to be employed, since they do not have an actual middle point. Composite graphic

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

objects need to consider the positioning of their constituent elementary graphic objects in order to have their center points computed and included in the knowledge base.

			
Position	Orientation	Size	Shape

Table 3.3: Spatial visual attributes characterized in UVO.

In any case, the absolute positioning measured in pixels is usually not of direct relevance for the user. Instead, the absolute positioning of a graphic object within the diagram may be expressed in more user-friendly terms by means of the `uvo:has_position_named` object property. Occurrences of this property link graphic object instances with individuals of the `uvo:Position_Named_VA`. These individuals characterize the discrete division of space following the model of Falomir et al. described in [CF15]. Moreover, the relative positioning between two or more graphic objects may be inferred from this information and communicated to the user by client applications when relevant. One of the most relevant semantics derived from relative positioning between diagrammatic elements is that of working (i.e. navigational) order [Myn95], described in Section 6.3.1, which is required prior knowledge for most navigational tasks performed on diagrams.

The *size* of a graphic object is defined as a perceptual stimulus stemming from variation in the dimensions of its area [Ber11]. Size is very commonly employed as the carrier visual attribute for ordinal and numerical variables in diagrams, thus needing both quantitative and qualitative axiomatic interpretations. Size in UVO may be measured in squared pixels by means of the `uvo:has_area` datatype property. However, expressing size in quantitative terms is not useful for all kinds graphic objects such as paths and text, among other SVG shapes. Moreover, size communicated in this form is commonly not very practical for end users. Therefore, individuals underpinning non-exhaustive discrete size categories such as ‘very small’ or ‘medium’ are predicated by instances of the `uvo:Size_Named_VA` class. Sometimes, only a certain dimension of a graphic object acts as the carrier of information, such as bars in a bar chart, whose width is often freely chosen by its author following aesthetic criteria. In such cases, the `uvo:has_length` and `uvo:has_width` datatype properties ought to be employed by client applications instead of (or along with) `uvo:has_area` in order to compute the associated numerical value of the graphic object. Note that three-dimensional graphic objects are at the moment not considered in UVO. Objects of changing size (such as a line of varying thickness characterizing changes in value) are to be divided into a number of elementary graphic objects of fixed size in a containment relationship with their parent.

*Orientation* of graphic objects may be characterized in the exact same way of orientation of texture area-fill visual attributes we already discussed in the previous section. Therefore, the `uvo:has_orientation` and `uvo:has_orientation_named` properties of UVO are overloaded and may underpin the orientation of graphic object instances themselves as well as their textures. Lastly, the *shape* of a graphic object may also be of relevance and has been included in UVO. The specific attributes controlling the shape of a graphic object are irrelevant for the purpose of interpreting diagrams, as shape is used in most cases to convey nominal information [Ber11] (what Wilkinson [Wil05] calls *categorical aesthetics*) and thus clients only need to be able to distinguish between different shapes. Therefore, shape is implemented in UVO in a straightforward manner with the description of a number of disjoint categorical shapes such as `uvo:4_point_star_shape` or `uvo:Square_shape`, which in turn are instances of the `uvo:Shape_VA` class.

## 3.7 Visualization Domain Ontology

In this section we outline the resources belonging to the Visualization Domain Ontology (VDO), the second layer of the suggested hierarchy of visualization semantics. As already discussed in Section 3.5.2, these resources govern the structure of graphic representations by providing a syntax akin to that of natural language which depends on the ‘building blocks’ provided by the Upper Visualization Ontology (UVO), outlined in the previous section.

### 3.7.1 Graphic Relation (`vdo:Graphic_Relation`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** `vdo:Object_to_Object_GR`, `vdo:Object_to_Space_GR`

Along with graphic objects and graphic spaces, the *graphic relations* that take place among them are a vital part of what imbues meaning to graphic representations. Two or more graphic resources are said to be involved in a graphic relation if a meaningful interpretation can be ascertained from their relative context i.e., from the way they relate to each other by comparing some of their visual characteristics, such as their relative positions, sizes, etc. A composite graphic object is syntactically structured by the set of graphic relations in which its constituent graphic objects are related [Eng02]. Examples of graphic relations are the relationship between a container graphic object and the graphic objects contained therein or the affiliation between a labelling graphic object and the graphic objects it labels. Graphic relations may be divided into two different kinds: *object to object* (O2O) graphic relations, and *object to space* (O2S) graphic relations, depending on the nature of the graphic elements that take part in it. Furthermore, object to object graphic relations may be divided into *attribute-based* relations and *spatial* relations between objects. These subclasses will be discussed in their respective resource entries.

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

At first glance it may seem that graphic relations ought to be characterized as properties in an ontology, as they underpin a relationship taking place between subject and object graphic objects. However, we have decided that the idea of graphic relations being ontological instances in which two or more graphic objects take part most closely resembles the graphic syntax described by Frege's principle (Section 3.5.2). Moreover, in graphic relations where many graphic objects partake, this approach simplifies vocabulary design. Let us consider a cluster made up of a thousand data points. Under the former resolution, a total of 999,000 triples would have to be included to the knowledge base, each linking two data points in the cluster together. On the other hand, describing the clustering relationship as an ontological instance would lead to including 2,000 triples in the ontology (one triple for each data point connecting it to the clustering instance, plus their respective inverse properties) i.e., a reduction of almost 99,8% to the size of the vocabulary. Graphic relations of a diagram are therefore realized by connecting graphic relation instances to graphic object instances by means of the property `uvo:hasSVGEElement`. Each graphic object instance whose constituent SVG elements are also constituents of a graphic relation instance is then considered to be taking part in said relation. Furthermore, each graphic object instance taking part in a graphic relation is said to play a *graphic role* in it. For instance, in a linking relationship at least one of its constituent graphic objects must play the *connector* role. Graphic roles will be discussed in Section 3.7.14.

Some authors consider visual attributes to be a subclass of the graphic relation class [PV13]. However, we consider that they refer to two fundamentally different concepts with regard to their interpretation. Firstly, visual attributes are properties of individual graphic object instances, whereas graphic relations must be applied to at least two graphic objects in a dependent manner. If one of its graphic objects gets removed from the relation, its semantics may completely change or become illogical. Moreover, visual attributes can be inferred by means of sight without any understanding of the diagrammatic roles of the graphic object they pertain to. This, on the other hand, is not the case of high-level graphic relations such as labelling. For example, in order to derive that two graphic objects are involved in a labelling object-to-object relationship it is paramount to first understand the concept of the *label* syntactic role that graphic objects may play in a diagram. Therefore, Graphic Relation and Visual Attribute are disjoint classes in VDO and UVO.

#### 3.7.2 Object-to-Object Graphic Relation (`vdo:Object_to_Object_GR`)

- **Entity Type:** Class
- **Subclass of:** `vdo:Graphic_Relation`
- **Subclasses:** `vdo:Attribute-based_O2O_GR`, `vdo:Spatial_O2O_GR`, `vdo:Labelling_GR`

Graphic relations involving two or more graphic object instances are named *object-to-object* (O2O) graphic relations. In this kind of graphic relation, graphic objects are *syntactically*

*anchored* to each other [Eng02]. This means that, if any of the graphic taking part in the relationship is removed, the nature of the relationship changes (or is rendered incongruous). Moreover, this anchoring is loosely defined in the sense that the objects' spatial arrangement may be altered without influencing the nature of their graphic relation. For instance, two objects connected by a connector could be moved to a greater distance from each other, but, as long as the connector object keeps connecting them, the linking graphic relation is not altered. Similarly, graphic objects taking part in the same clustering relation may be spatially rearranged as long as the cluster they belong to remains discernible from other clusters. Object-to-object graphic relations may be further divided into *attribute-based* and *spatial* object to object relations depending on whether the relation hinges on visually perceptible attributes of the objects besides position (e.g., a shared color) or whether it depends on the relative position of the objects with respect to each other (e.g., points in a cluster). Therefore, the `vdo:Object_to_Object_GR` class has two disjoint subclasses in VDO. Namely, `vdo:Attribute-based_O2O_GR` underpins attribute-based O2O graphic relations whereas `vdo:Spatial_O2O_GR` characterizes spatial O2O graphic relations. None of these three classes may be directly instantiated. Instead, graphic relation instances may belong to any of their child classes, which specify the specific type of O2O graphic relation taking place between its graphic objects. The child classes of `vdo:Attribute-based_O2O_GR` and `vdo:Spatial_O2O_GR` are described in the following sections.

An object may be simultaneously involved in a number of different object-to-object graphic relations, as it is generally the case in most diagrams. For example, a color-encoded slice in a pie chart with a legend may be involved in a proportional division relationship (itself a member of a containment relationship) within the pie graphic object, as well as being involved in a color-based relationship with its respective label in the legend. Each of these simultaneous graphic relations in VDO is underpinned as an instance of the object-to-object graphic relation class sharing the same object instances in their `vdo:involves_graphic_object` property occurrences. Graphic object instances taking part in O2O graphic relations may also be identified with the inclusion of SVG shapes to the graphic relation instance by means of the `uv0:hasSVGEElement` datatype property. In this manner, those graphic objects whose SVG elements are contained within the SVG elements of the graphic relation can be inferred to be taking part in it. Since these properties are automatically included by SAI (Chapter 4), this approach simplifies the ontology authoring process.

We will now discuss the available spatial object-to-object graphic relations of VDO, see Table 3.4. Namely, these are *clustering*, *containment* (and *proportional division*), *line-up* (and *proportional repetition*), *labelling*, *linking*, *superimposition*, and *separation by a separator*. Afterwards, attribute-based object-to-object graphic relations will be listed.

### 3.7.3 Clustering Graphic Relation (`vdo:Clustering_GR`)

- **Entity Type:** Class
- **Subclass of:** `vdo:Spatial_O2O_GR`
- **Subclasses:** `vdo:Separation_by_Separator_GR`

Graphic objects taking part in a *clustering* graphic relation are visually perceived as being in close proximity to each other forming a single, higher-level, entity (a cluster), see Table 3.4 for a simple example. Gestalt principles suggest an innateness to the visual categorization of clusters [Wil05]. Clustering is said by some authors to be the organization phenomenon most associated with visual perception, as the human brain has an innate capability of synergistically combining elements and clues of the visual field in order to form strong evidence of grouping [CF15]. Clustering relationship are therefore widely employed in statistical diagrams to group together related datapoints sharing a number of attributes. Perhaps the most salient example of clustering graphic relations being used in diagrams is the scatterplot, where high-level information about a dataset, such as the correlation between variables or outlier detection, can be quickly inferred by means of sight.

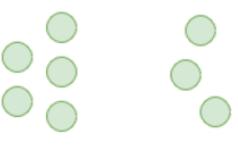
		
Clustering	Separation by Separator	Linking
		
Line-up	Ordered Line-up	Superimposition
		 <span style="border: 1px solid black; padding: 2px;">Label</span> <span style="border: 1px solid black; padding: 2px;">Label</span>
Containment	Containment (Proportional Division)	Labelling

Table 3.4: Spatial object-to-object graphic relations characterized in VDO (note that labelling is a special, higher-level graphic relation that may also be characterized by an attribute-based graphic relation).

Given their visual salience and commonplace usage in diagrams, the clustering graphic relation is included, under different denominations, in all approaches to visualization formalization we have studied. For instance, clusters are named *multipoints* or *point clouds* by Wilkinson

[Wil05]. He notes how whether clusters should be considered elementary or composite graphic objects is a challenging matter: “In graphics software, there is a choice that object-oriented designers have argued over for years. On the one hand, we may call a point cloud a single object. On the other hand, we may call each point a single object”. He then goes on to conclude that clusters ought to be regarded as “collections of points in order to encapsulate functions that are most appropriately handled at the point level”. UVO handles this issue in a similar but conceptually different manner: as we already discussed, each individual point taking part in a clustering graphic relation is on itself an instance of the elementary graphic object class taking part in an instance of the clustering graphic relation class. In this manner, software-enabled user tasks requiring finer granularity at the datapoint level may disregard the graphic relation instance focusing on the characteristics of the individual elementary graphic objects, whereas tasks that retrieve information applicable to the cluster as a whole may fetch attributes from the clustering graphic relation class instance.

Clustering graphic relation instances may be conveniently embedded to an SVG graphic by means of SAI (Section 4.5) by selecting all graphic object instances taking part in it and adding an associated instance of the Clustering Graphic Relation class. Moreover, clusters may be automatically computed in a straightforward, efficient manner via the nearest neighbors’ algorithm. This method measures the (Euclidean) distance of each graphic object in the image to every other graphic object on the SVG canvas. If the computed distance is less than one quarter of the maximum diagonal image, it is considered that those two elements are part of the same clustering graphic relation instance. An example implementation of the nearest neighbors’ algorithm can be consulted in [CF15].

### 3.7.4 Separation Graphic Relation (vdo:Separation\_By\_Separator\_GR)

- **Entity Type:** Class
- **Subclass of:** vdo:Clustering\_GR
- **Subclasses:** None

Separation (more specifically, *separation by a separator*) graphic relations are a kind of object-to-object graphic relation that perform the same syntactic function as clustering graphic relations, previously discussed. However, this kind of relationship is generally employed when the constituent graphic objects of different clusters are too close together, preventing the clusters to be visually told apart from each other, or when the author of the diagram wants to highlight the division of the dataset into sub-groups for some other reason (an example is shown in Table 3.4). The syntactical grouping of graphic objects into distinct clusters is thus made explicit by another graphic object playing the syntactic role of a *separator* (vdo:Separator\_SR). The separator is most commonly drawn as a line segment dividing the graphic space occupied by the graphic objects in two parts.

Given the syntactic similarity between the clustering and separation graphic relations, the latter is described by UVO as a subclass of the Clustering Graphic Relation. However, it is

necessary for non-separator graphic objects taking part in separation graphic relation instances to also be involved in a clustering graphic relation instance. Otherwise, it would not be possible to tell the members of each group apart, since they are all grouped under the same Separation by Separator Graphic Relation instance. By adding a number of Clustering Graphic Relation instances, the common characteristics of the separated graphic objects can be unambiguously operated by client applications. The graphic objects playing the syntactic role of separator must obviously be excluded from any Clustering Graphic Relation, being only associated to instances of the Separator Graphic Relation class.

#### 3.7.5 Linking Graphic Relation (**vdo:Linking\_GR**)

- **Entity Type:** Class
- **Subclass of:** vdo:Spatial\_O2O\_GR
- **Subclasses:** None

A *linking* graphic relation (see Table 3.4) is made up of three graphic objects, two of which are syntactically related in the diagram; and a third one (the *connector*) bringing forward the relationship between the other two graphic objects by physically connecting them, most commonly in the shape of a straight or curved line segment. Moreover, the connector graphic object may be *directed*, indicating that the relationship takes place from one graphic object (the *subject*) to another (the *object*), or *undirected*. This kind of graphic relation is extensively used in link diagrams, discussed in Chapter 1, in which they express a certain relationship among their components. For instance, family trees generally depict family relationships by connecting individuals via labelled line connectors. Line connectors are also traditionally employed in diagrams to connect labels with the elements they label.

In UVO, a linking graphic relation instance always involves a *connector* graphic object and two non-connector graphic objects, which are said to play *node* roles. The connector graphic object must be involved in a `vdo:has_syntactic_role` object property occurrence as its *subject* in which the *object* is an instance of the *Connector* syntactic role class (`vdo:Connector_SR`). Graphic object syntactic roles are described in next section. Other authors, such as Wilkinson [Wil05], consider the connector (which he names *edge*) to be the sufficient graphic object for a linking graphic relation to take place. In this thesis however it is required for connectors to take part in linking graphic relations along with two other graphic objects since, if an edge of sorts without two connecting graphic objects appears in a diagram, we consider it to be a better candidate for some other kind of graphic relation, such as a separation by separator or an object to space relation.

#### 3.7.6 Lineup Graphic Relation (**vdo:Lineup\_GR**)

- **Entity Type:** Class
- **Subclass of:** vdo:Spatial\_O2O\_GR

- **Subclasses:** vdo:Ordered\_Lineup\_GR, vdo:Proportional\_Repetition\_-GR

*Lineups*, also known as *alignments*, are one of the most commonly employed object-to-object graphic relations in diagrams. Graphic objects taking part in a lineup graphic relation are arranged in a visually perceptible line, each of them having two immediate neighbors in the resulting 'chain' of objects, except for those placed at either end. One of the most common occurrences of this kind of relation in a diagram are bars lineups in a bar chart. They could be seen as 'elongated' cluster instances, which may be arranged in all sorts of shapes, not just constrained to forming straight lines. Moreover, lineups can be *ordered* or *unordered*, depending on whether altering the position on an object within the lineup alters the intended meaning of the representation [Eng02]. Lineup graphic relation instances are considered unordered by default in VDO. Ordered lineup graphic relation instances may be added by instantiating the vdo:Ordered\_Lineup\_GR Lineup subclass.

Frequently, lineup graphic relation instances are arranged along a metric axis in the diagram. For example, bars in a bar chart may be positioned along one of its axes depicting ordinal information, such as time. In this case, the fact that the graphic objects taking part in the lineup graphic relation are also occupying a metric graphic space must be taken into consideration when operating with them. For example, numerical labels for each bar may be inferred by interpolating the position of the bar graphic objects with the positions of the labels along the metric axis it occupies.

Special care must be taken so that containment graphic relations are not misinterpreted as lineup graphic relations. Any time a composite graphic object is divided into sub-graphic objects, the most likely graphic relation taking place is one of containment. Use of lineup instances must not be employed to underpin the relationship between a composite graphic object and its constituent elements. For instance, a stacked bar in a bar chart plays the role of the container for its constituent sub-bars, and they should not be understood as a lineup of graphic objects. See Figure 3.14 for a visual example.

### 3.7.7 Proportional Repetition Graphic Relation (vdo:Proportional\_Repetition\_GR)

- **Entity Type:** Class
- **Subclass of:** vdo:Lineup\_GR
- **Subclasses:** None

A *proportional repetition* is a special case of lineup graphic relation. It consists on an alignment of identical copies of the same graphic object, as opposed to traditional lineups which can be made up of any number of dissimilar graphic objects. The number of occurrences of the object of the lineup usually carries meaning in the diagram much in the same way the *size* visual attribute can be employed to compare magnitudes between graphic objects of different

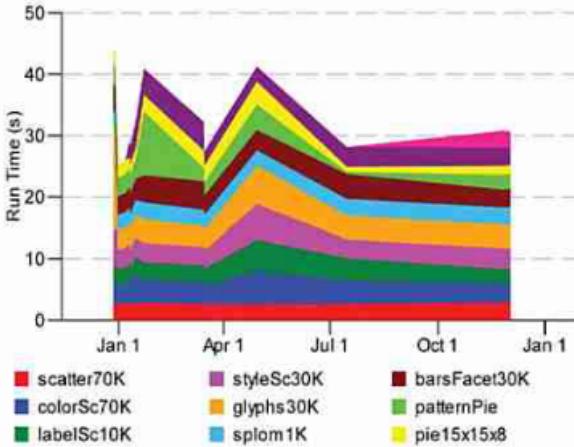


Figure 3.14: A stacked area chart (image taken from [Wil05]). Each of the colored areas (*area locators*) are interpreted in UV0 as sub-graphic objects of a composite graphic object underpinning the aggregation of all area locators. Each area locator thus is in a *containment* relationship with the composite graphic object, and must **not** be interpreted as taking part in *lineup* graphic relation instances.

sizes. Therefore, they usually carry a similar syntactic role to that of metric bars in a bar chart, sometimes involving an implicit metric axis in the direction of the lineup [Eng02].

The graphic objects taking part in a proportional repetition graphic relation usually carry little to no meaning by themselves, and client applications need generally only consider how many of them are participating in it, as most relevant information (labels, etc.) is generally associated with the graphic relation instance itself. Nevertheless, sometimes the object being repeated in the lineup has an associated labeled graphic object in a supporting legend graphic object that needs considering.

### 3.7.8 Containment Graphic Relation (`vdo:Containment_GR`)

- **Entity Type:** Class
- **Subclass of:** `vdo:Spatial_O2O_GR`
- **Subclasses:** `vdo:Proportional_Division_GR`

*Containment* graphic relations are one of the most common graphic relations appearing in diagrams. Containment relation instances have two associated graphic objects (see Table 3.4): one named *container* and another, physically enclosed in its entirety within the container, called *content* or simply *node*. Containment graphic relations are essential to the recursive nature of graphics (Section 3.5.2) and therefore very often instances of this class are rendered redundant, as the containment relationship may be implicitly inferred from the two inverse properties

`uvo:has_graphic_object` and `uvo:is_part_of_graphic_object` underpinning aggregation of graphic objects into composite graphic objects. Unless the containment relationship needs to be made explicit because of domain-specific accounts (for example, to explicitly annotate membership relationships in grouping diagrams or to characterize a proportional repetition graphic relation) it is advised to leave containment graphic relations out of the diagrammatic vocabulary, resorting instead to using aggregate-composite relationships of composite graphic objects with their constituent elements.

A container may contain any number of graphic objects of any kind, but each of these containment relationships must be expressed as a new instance of the containment graphic relation. Moreover, graphic relations are transitive i.e. the container of a graphic object is also the container of graphic relation instances with the graphic object's contents in a recursive fashion. However, given that graphic relationships are underpinned by ontological instances instead of properties, this knowledge cannot be directly embedded in the ontology. Instead, the transitivity of `uvo:has_graphic_object` can be leveraged (since contained objects are also constituent graphic objects of the container graphic object). Constituent graphic objects of a composite graphic object carrying the `vdo:Content_SR` syntactic role may then be inferred to be also members of containment graphic relationships with their grandparent graphic object. Additionally, as already discussed in the line-up graphic relation entity entry (Section 3.7.6), special care must be taken not to misinterpret containment graphic relations for line-up graphic relations (see Figure 3.14).

### 3.7.9 Proportional Division Graphic Relation (`vdo:Proportional_Division_GR`)

- **Entity Type:** Class
- **Subclass of:** `vdo:Containment_GR`
- **Subclasses:** None

*Proportional division* graphic relation instances divide the container graphic object taking part in a containment graphic relation into a number of contained graphic objects taking up the entirety of its area (see Table 3.4). In this manner, each constituent graphic object of the container represents a percentage of the total value expressed by it. A traditional implementation of this sort of graphic relation are pie charts, in which the pie (container graphic object) is divided into a number of slices (content graphic objects).

It is paramount to characterize this kind of graphic relation any time they take part in a diagram, since their interpretation differs from that of aggregate-composite relationships between composite graphic objects and their constituent graphic objects. Client applications should infer the associated value of contained sub-objects partaking in proportional division graphic relation instances from the value of their container graphic object and their relative size difference (unless labels or some other available informational graphic objects offer this information instead).

### 3.7.10 Superimposition Graphic Relation (vdo:Superimposition\_GR)

- **Entity Type:** Class
- **Subclass of:** vdo:Spatial\_O2O\_GR
- **Subclasses:** None

A *superimposition* graphic relation instance (see Table 3.4) has two associated graphic objects: one in the foreground and another one in the background partially occluded by the former graphic object. Superimpositions are often employed in order to alter the meaning of the background object with additional information provided by the object in the foreground. For example, traffic signs may employ a red bar diagonally crossing their entirety in order to negate the meaning that can be inferred from the graphic objects they contain. This kind of graphic relation is sometimes ambiguous and could be also interpreted as one of containment. As a rule of thumb, an overlap of graphic objects that involves partial occlusion or where the foreground objects sticks beyond the borders of the background object shall be regarded as a superimposition of objects [Eng02].

### 3.7.11 Attribute-based Object-to-Object Graphic Relation (vdo:Attribute-based\_O2O\_GR)

- **Entity Type:** Class
- **Subclass of:** vdo:Object\_to\_Object\_GR
- **Subclasses:** vdo:Color-based\_O2O\_GR, vdo:Texture-based\_O2O\_GR,  
vdo:Blur-based\_O2O\_GR, vdo:Transparency-based\_O2O\_GR,  
vdo:Orientation-based\_O2O\_GR, vdo:Size-based\_O2O\_GR, vdo:Shape-based\_O2O\_GR

*Attribute-based* object-to-object graphic relation instances relate graphic objects involving variations in visual attributes such as size and color. For instance, it is common that points in a scatterplot are related to some graphic object in an accompanying legend sharing their same shape where extra information is given about them. In this case, each of these points in the scatterplot would take part in a *shape-based* object-to-object graphic relation with the informational object in the legend. Therefore, for each visual attribute available in UVO (see Section 3.6.9), a sub-class of the vdo:Attribute-based\_O2O\_GR class exists, all of which are listed in the 'Subclasses' list item above. The only exception is position, since graphic relations based on the relative positioning of graphic objects are included under the spatial object-to-object graphic relation class, already discussed.

### 3.7.12 Labelling Graphic Relation (`vdo:is_label_for`, `vdo:is_labeled_by`)

- **Entity Type:** Object Property

*Labels* are graphic objects visually associated to a number of nodes (other graphic objects) in order to augment them with additional semantics, generally by means of text or pictograms. A *labelling* graphic relation therefore is that taking place between a label graphic object and the node or nodes it labels. Labelling is a special, of a conceptual higher-level, graphic relation which must be derived from a basic object-to-object graphic relation between the label and the node it labels. Most commonly, this foundational O2O graphic relation is either one of clustering, where the label is placed in close proximity to the labelled nodes; or one of linking, where the label and the nodes it labels are connected by connectors. Even if labelling is to be derived from other object-to-object graphic relations, most authors give dedicated attention to it in their models and taxonomies for visualization. For instance, Wilkinson [Wil05] considers labelling a special aesthetic function (i.e. a Visual Attribute), whereas Engelhardt [Eng02] acknowledges labelling as a special case of spatial clustering. Moreover, having an explicitly characterized property for describing labelling graphic relations can greatly simplify the annotation process of semantically-enhanced diagrams via *syntactic overload*, as described next.

Given that the labelling graphic relation is not a basic graphic relation *per se*, it has not been included in VDO as a subclass of `vdo:Graphic_Relation`. Instead, labelling graphic relations are characterized through the `vdo:is_label_for` object property and its inverse `vdo:is_labeled_by`, whose occurrences axiomatically connect label and node graphic objects together. These properties may be overloaded to cover labelling relationships that may be inferred from any combination of graphic relations in order to simplify handling of the knowledge base by client applications. For example, let us consider a diagram labelled by a nested legend in a background-inset display (as shown in Figure 3.15). A graphic object within the legend is employed to label elements of the diagram by means of a shared color (an attribute-based O2O graphic relation). Such complex inference of labelling relations between graphic object could then be simplified by human annotator or client applications by instantiating occurrences of the `vdo:is_labeled_by` property between the elements of the diagram and their associated label in the legend.

### 3.7.13 Object-to-Space Graphic Relation (`vdo:Object_to_Space_GR`)

- **Entity Type:** Class
- **Subclass of:** `vdo:Graphic_Relation`
- **Subclasses:** None

Instances of the *object-to-space* (O2S) graphic relation class predicate meaningful relationships between graphic object instances and metric graphic space instances. For example, a bar in a

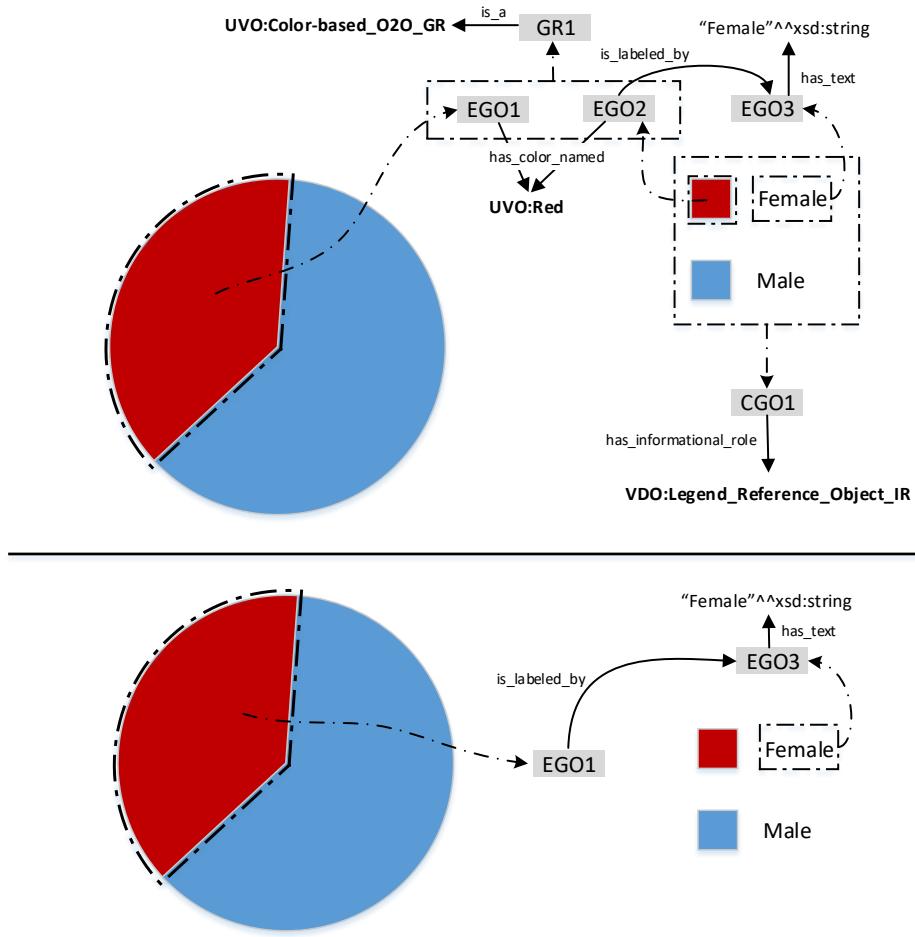


Figure 3.15: The syntax of a complex labelling attribute-based object-to-object graphic relation (above) may be simplified by means of a `vdo:is_labeled_by` property occurrence (below). `vdo:is_labeled_by` is a syntactically overloaded object property that may underpin graphic relations of disparate nature.

bar chart may be associated with its metric axis in a knowledge base by instantiating an O2S graphic relation and then associating both graphic objects (the bar and the axis) with the graphic relation instance. In VDO, much in the same way of O2O graphic relations, O2S graphic relations may be instantiated in two different ways: (1) by means object properties (namely, `uv0:involves_graphic_space`, which connects a O2S relation with the graphic space instance it involves; and `uv0:involves_graphic_object`, associating the O2S relation with each graphic object instance it affects); and (2) by means of shared constituent SVG elements between the O2S graphic relation instance and its members (metric graphic space and graphic object instances).

O2S graphic relations may be completely disregarded in simple semantically-enhanced diagrams. For instance, a knowledge base underpinning a scatterplot consisting of a single composite graphic space does not need to instance any O2S graphic relation, since client applications can safely assume that each data point and axis in the diagram occupies the same graphic space (which may not even be instantiated in the ontology, see Section 3.6.6).

### 3.7.14 Syntactic Role (`vdo:Syntactic_Role`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** `vdo:Object_to_Object_Anchoring_SR`, `vdo:Object_to_Space_Anchoring_SR`

For a graphic relation to be comprehensively characterized, each graphic object taking part in it must play some *syntactic role* therein. For instance, in a separation by separator graphic relation, the graphic object playing the role of separator has to be unambiguously identifiable. Otherwise, the graphic relation could not be interpreted. Syntactic roles are therefore a necessary addition to any knowledge base underpinning diagrammatic representations. Graphic syntactic roles are said to be ‘anchored’ to a certain syntactic structure much in the same way that syntactic roles in natural language are anchored to certain words or phrases (e.g. adverbs being attached to verbs) [Eng02]. For example, a graphic object playing the connector graphic role in a linking graphic relation must be anchored to two nodes (i.e. two graphic objects playing the ‘node’ role in the graphic relation). Syntactic roles are accordingly categorized in two disjoint groups: *Object-to-Object (O2O) anchoring* syntactic roles and *Object-to-Space (O2S) anchoring* syntactic roles, which are described in the following sections.

Syntactic roles in VDO are implemented through literals belonging to the subclasses of the `vdo:Syntactic_Role` class, such as `vdo:Axis_SR` characterizing the syntactic role of an axis graphic object. Graphic objects partaking in graphic relations may then be assigned syntactic roles by means of `vdo:has_syntactic_role` object property triples. Some syntactic roles may be omitted from a knowledge base and safely inferred by client applications when their role can be deduced from the roles of other graphic objects taking part in the same graphic relation. Most commonly, `Node_SR` syntactic roles need not be included in knowledge bases, and is included in VDO for the sake of completeness.

### 3.7.15 Object-to-Object Anchoring Syntactic Role (`vdo:Object_to_Object_Anchoring`)

- **Entity Type:** Class
- **Subclass of:** `vdo:Syntactic_Role`
- **Subclasses:** several, see class description

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

Graphic objects taking part in object-to-object graphic relations are said to play *object-to-object (O2O) anchoring* syntactic roles. This means that objects in such a relationship may to some extent be positioned differently without altering the meaning of their relationship i.e. their syntactic roles mostly depend on each other and not on the graphic space they occupy. VDO characterizes six different O2O anchoring syntactic roles, namely:

- **Label** (vdo:Label\_SR): labels are one of the most common syntactic roles found within diagrams. Each labelling graphic relation needs at least one *label*, which may be labelling one or more graphic objects (each of these in turn playing a *node* syntactic role).
- **Separator** (vdo:Separator\_SR): In separation by separator graphic relations, the graphic object acting as the boundary between objects plays a *separator* syntactic role, whereas the rest play *node* roles. It is important to underline that separators generally do not live on metric (especially integral) graphic spaces. If the absolute positioning of all points on a separator carry interpretable semantics, the graphic object would be a line locator instead (an object-to-space anchoring syntactic role).
- **Connector** (vdo:Connector\_SR): In linking graphic relations, the graphic object playing the *connector* role is in charge of physically connecting two graphic objects, each playing a *node* syntactic role in the relationship. Note that the connector syntactic role is implemented as a class with two members: *directed* connectors (vdo:Directed\_Connector\_SR) follow a semantically relevant direction, whereas *undirected* connectors (vdo:Undirected\_Connector\_SR) do not. In directed connections, nodes being connected may either play the *subject* (vdo:Subject\_SR) or *object* (vdo:Object\_SR) role in the relation.
- **Container** (vdo:Container\_SR) and **Content** (vdo:Content\_SR): A containment O2O graphic relation is made up of one *container* and one or more graphic objects physically enclosed within the container. The contained graphic objects may indistinctly play a *content* or *node* syntactic role in the relation.
- **Modifier** (vdo:Modifier\_SR): some of the constituent elements of a composite symbol graphic object may play this role to signal an alteration to the meaning of the symbol's main element. For instance, a diagonal red line drawn across a graphic object (e.g. a cigarette icon) often indicates that its meaning is to be negated. This role is not strictly necessary, since client applications may ignore the constituent elements of the composite symbol if its meaning is already clear.
- **Node** (vdo:Node\_SR): Every graphic object instance taking part in an object-to-object graphic relation whose role is not one of the roles listed above must play the *node* syntactic role therein. Therefore, this role may be skipped from knowledge bases, as it is immediately inferable.

### 3.7.16 Object-to-Space Anchoring Syntactic Role (vdo:Object\_to\_Space\_Anchoring)

- **Entity Type:** Class
- **Subclass of:** vdo:Syntactic\_Role
- **Subclasses:** several, see class description

Graphic objects play a certain *object-to-space (O2S) anchoring* syntactic role in each graphic relation they belong to depending on the way they are anchored to the graphic space of the relation. Obviously, only graphic object contained within meaningful graphic spaces play O2S anchored graphic roles. The basic O2S anchoring syntactic roles are *point locator*, *line locator*, *surface locator*, and *volume locator*. Derived from these basic roles, we may also distinguish between *axis*, *grid line*, and *metric bar* roles. Let us now take a closer look at them.

- **Point Locator** (vdo:Point\_Locator\_SR): the meaning of graphic objects playing the *point locator* role is attached to a specific point of the graphic space where it is located. For example, each datapoint graphic object in a scatterplot plays a point locator syntactic role in its graphic relation with the metric graphic space of the diagram.
- **Surface Locator** (vdo:Surface\_Locator\_SR): graphic objects in which every point of their area is subject to interpretation with relation to a graphic space play a *surface locator* syntactic role within the corresponding object-to-space graphic relation. Most commonly, this role is found in objects relating to integral metric spaces, especially in maps.
- **Line Locator** (vdo:Line\_Locator\_SR): a graphic object playing a *line locator* syntactic role carries meaning along a line in space. As opposed to surface locators, the width (and thus, area) of the line is void of interpretable meaning. The most notable example of line locators in statistical diagrams are those of line charts, where each point along a line can be interpreted in relation to the metric space it occupies.
- **Volume Locator** (vdo:Volume\_Locator\_SR): the three-dimensional equivalent of the surface locator syntactic role i.e. every point in the graphic object's volume is subject to interpretation in the graphic space it occupies. This syntactic role is rarely found within statistical charts. Note that most three-dimensional objects appearing in diagrams are not volume locators. For example, in three-dimensional bar charts, the bars' size is only meaningful in one dimension (their length), with every other dimension merely serving a decorative purpose.
- **Metric Bar** (vdo:Metric\_Bar\_SR): a *metric bar* is a special case of line locator that, due to its prominence in diagrams, is characterized separately in VDO. The main characteristic of metric bars is that their interpretation is subject to the presence of

a parallel metric axis, which may be implied from a metric space. Therefore, metric bars take part in object-to-space object relations containing a metric space instance, an (optional) metric axis instance, and metric bar instances. The length of the bar may then be interpreted in relation to the length of the metric axis and its labels. `vdo:Metric_Bar_SR` is typified as a class in VDO with two individuals: `vdo:Metric_Bar_SR` and `vdo:Stacked_Bar_SR`. *Stacked bars* are metric bars made up of a combination of individual metric bars in a proportional division containment graphic relation.

- **Metric Axis** (`vdo:Axis_SR`): a *metric axis* is a line locator, commonly labelled by *tick marks* (`vdo:Tick_Mark_SR`) and *scale breaks* (`vdo:Scale_Break_SR`), that creates a metric space and serves a spatial reference informational role for the graphic objects living in it. Common metric axes include *horizontal*, *vertical*, *circular*, and *radial* metric axes. A metric axis may be employed to represent the passing of time, in which case it is called *time axis*. Note that often metric axes need to be inferred from metric spaces when an explicitly drawn axis is not present in the diagram in order to embed the necessary semantics into the knowledge base.
- **Grid Line** (`vdo:Grid_Line_SR`): *grid lines*, also known as *reference lines* [Cle94], are line locators that help sighted readers compare positions in a metric space by marking them. Since they do not carry any information by themselves, graphic objects playing this role may be safely ignored by client applications in most scenarios.

## 3.8 Data Domain Ontology

This section illustrates the principal resources of Data Domain Ontology (DDO), which semantically enhance the *visualization* knowledge provided by UVO (Section 3.6) and VDO (Section 3.7) by providing a number of resources related to the intrinsic *data* the diagram portrays and its interpretation. For more details, see section 3.5.2.

### 3.8.1 Informational Role (`ddo:Informational_Role`)

- **Entity Type:** Class
- **Subclass of:** `uvo:Graphic_Thing`
- **Subclasses:** `vdo:Reference_Object_IR`

A graphic object, in addition to any of the syntactic roles characterized in VDO, also needs to play an *informational role* in the graphic relations it takes part in. Whereas the syntactic role refers to the role of the graphic object within the graphic relation's structure, its information role is concerned about how it may be interpreted by a human reader. For example, an explanatory superimposed *legend* in a diagram may be a composite graphic object taking part in a superimposition graphic relation with the rest of the diagram. However, this information by itself does not suffice to represent it as the diagram's legend. Therefore,

the `ddo:Legend_Reference_Object_IR` informational role needs to be related to the legend graphic object for this knowledge to be included in its semantic foundation.

Each graphic object may only play one single informational role within the whole diagram, as otherwise its interpretation would be incongruent. Constituent graphic objects of composite graphic objects may however play a different informational role to that of their parent. Each graphic object instance may be assigned an informational roles by means of the `ddo:has_informational_role` object property which connects them. Engelhardt [Eng02] suggests that informational roles are divided into *Information Objects*, *Reference Objects*, and *Decoration Objects*, and are as such characterized by DDO.

*Information objects* (`ddo:Information_Object_IR`) carry the inferable data depicted by the diagram. As such, any variation to their meaningful visual attributes would result in the data being changed. For example, modifying the position of a data point in a scatterplot would also alter its underlying numerical value. This is in sharp contrast with *decoration objects* (`ddo:Decoration_Object_IR`), which, as their name implies, are merely decorative, and their removal from the graphic representation would not result in data loss or modification. *Reference informational objects* require a longer explanation, and are introduced next.

### 3.8.2 Reference Informational Role (`ddo:Reference_Object_IR`)

- **Entity Type:** Class
- **Subclass of:** `ddo:Informational_Role`
- **Subclasses:** None

*Reference objects* do not carry any knowledge about the diagram's embedded data itself, but may not be considered as serving a merely decorative purpose either. Rather, they allow for the proper interpretation of the data portrayed by the information objects of the diagram. Therefore, removing or altering reference objects would not result on the diagram's data being modified, but it may make it harder or impossible to infer. For example, the meaningful color of graphic objects in a map may not be explainable without an accompanying explanatory legend. Likewise, a metric graphic space would not be easily interpretable without an accompanying axis graphic object labelling it. We distinguish between two disjoint reference informational roles, characterized in DDO as individuals of `ddo:Reference_Object_IR`. Namely:

- **Legend Objects** (`ddo:Legend_Reference_Object_IR`): legends are composite graphic objects, generally superimposed or shown next to the main part of the diagram, which are in charge of explaining the interpretation of the meaningful visual attributes of the graphic objects present in the diagram (with the only exception of position [Wil05]). Legend graphic objects are usually made up of a number of elementary graphic objects labelled by textual graphic objects.

- **Spatial Reference Objects** (`ddo:Spatial_Reference_Object_IR`): this kind of graphic object serves as a reference to interpret a meaningful space. The most common syntactic role for spatial reference objects is the axis, which generally appears in diagrams in order to signify the starting point and magnitude of the variable expressed by one dimension of a metric space. Spatial reference objects may either be *continuous* or *categorical*.

### 3.8.3 Variable (`ddo:Variable`)

- **Entity Type:** Class
- **Subclass of:** `ddo:Chart_Domain`
- **Subclasses:** `ddo:Dependent_Variable`, `ddo:Independent_Variable`

Ultimately, statistical diagrams are nothing but a visual way of displaying raw data values (c.f. Section 1.1.2). The *Variable* class of DDO is the super-class for the data types of the latent data depicted in the diagram. If data were to be expressed in the row and column-based tabular model (such as those employed by most data display computer applications), each column of the model would be mapped to a literal resource of `ddo:Variable`. For example, a simple time chart depicting population growth with each passing year would have two instances of `ddo:Variable` in its corresponding knowledge base: these could be named `ddo:Year` and `ddo:Population`.

Note that no knowledge regarding the nature of these variables may be embedded in DDO, as DDO is exclusively concerned with characterizing the latent data of the diagram as-is. Most commonly, domain knowledge about the diagram has to be inferred by readers from its surrounding text, previous own knowledge about the subject, or some other external means. Likewise, variables in DDO may be associated with high-level knowledge by means of external ontologies representing the given domain of the variable (Section 3.5.2). For example, resources from OWL-Time [CL17], an ontology (recommended namespace prefix `time`) which describes temporal concepts and their properties, could be associated to variables related to time. In the previous example, `ddo:Year` could be given higher-level time semantics by associating it with `time:year`, an OWL-Time datatype property which defines “years position in a calendar-clock system” [CL17]. This semantic association may be done by means of a `ddo:related_to` object property occurrence, with object `ddo:Year` and subject `time:year`. In some cases, a class or property expressed by an instance of `ddo:Variable` in the diagram is identical to that of a supporting domain ontology. In such cases, authors may choose to relate them by means of the `owl:equivalentClass` and `owl:equivalentProperty` object properties of OWL.

Variables are encoded in diagrams by means of graphic relations, either object-to-space (e.g., through a metric graphic space whose dimension is proportional to the measure of a variable) or object-to-object (e.g., via color intensities in proportion to the measure of a variable).

Graphic relations may be associated with certain variables through `ddo:expresses_-information` object property triples. In object-to-space graphic relations characterized by a graphic object playing the `vdo:Axis_SR` syntactic role, the axis object may be linked to the variable it represents. Every other object-to-space graphic relation as well as object-to-object graphic relations are associated with the variable they depict by connecting the graphic relation and variable instances together. Finally, variables may directly implement the `ddo:Variable` class or any of its subclasses, which will be introduced next.

### 3.8.4 Dependent Variable (`ddo:Dependent_Variable`)

- **Entity Type:** Class
- **Subclass of:** `ddo:Variable`
- **Subclasses:** None

Statistical diagrams are commonly employed in science in order to discover how two quantitative variables are related. In such a two-variable study, the goal is to find how one of them, called the *dependent variable*, depends on the other, the *independent variable* [Cle94], also known as *regressor*. Examples of diagrams commonly employed to graph two quantitative variables include the scatterplot, line chart, and time chart. DDO allows variables to be complemented with information about their dependency by means of the `ddo:Dependent_Variable` and `ddo:Independent_Variable` subclasses of `ddo:Variable`.

Note that `ddo:Variable` may be directly instantiated when no dependency between variables takes place or when this fact is unknown. For instance, categorical and ordinal variables must be instances of `ddo:Variable`, as well as quantitative variables in diagrams with only one thereof. On the other hand, a diagram may contain more than one composite graphic space or higher-dimensional graphic spaces. In this case, the dependency relations between the dependent and independent variables must be explicitly characterized by means of `ddo:depends_on_variable` object property occurrences.

### 3.8.5 Information Type (`ddo:Information_Type`)

- **Entity Type:** Class
- **Subclass of:** `ddo:Data`
- **Subclasses:** None

Diagrams may carry different information types by means of well-suited graphic relations. For example, the position of a data point in a scatterplot carries quantitative information (an exact numerical value). This knowledge can be established from its position because the data point takes part in an object-to-space graphic relation with the graphic space created by a metric axis. Likewise, the color of graphic objects in a demographic chart can, for

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

example, determine to which category of an existing ordinal variable (e.g., sex, where blue stands for ‘Male’ and red for ‘Female’) the object is to be associated with. In the former case, these informational graphic objects may be assigned exact meaningful numerical values, whereas in the latter case, they are associated with categorical variables which cannot be numerically interpreted nor sorted. Therefore, variables may belong to different *information types* according to whether their values can be absolutely quantified and/or sorted.

According to Bertin [Ber11], there exist three different information types (what he calls *levels of organization*), which have been implemented as individuals of `ddo:Information_Type`. Namely:

- **Nominal** (*qualitative* level; `ddo:Nominal_Information`): nominal variables include “all the innumerable concepts of simple differentiation such as professions, products, languages, etc.” [Ber11]. As such, nominal variables are not ordered in a universal manner and may be reordered arbitrarily without altering the knowledge contained in the diagram i.e. each value is of equal importance.
- **Ordinal** (*ordered* level; `ddo:Ordinal_Information`): the values of an ordinal variable can be ordered in a universally meaningful manner. Their ordering is always defined in relation to a temporal order (such as age), an order of sensory discrimination (such as named sizes), an order of intellectual or moral discrimination (for example, good-mediocre-bad), or certain social structures (such as military rankings) [Ber11].
- **Cardinal** (*quantitative, interval-ratio* level; `ddo:Cardinal_Information`): the object of quantitative variables is to specify the variation in distance among its values [Ber11]. Therefore, a quantitative variable’s values can be mapped to real numbers and carry their same properties, including meaningful sorting as well as closed addition, subtraction, and multiplication.

Variables may be characterized as having a given information type by means of `ddo:has_information_type` object property occurrences linking `ddo:Variable` and `ddo:Information_Type` instances. Not all graphic relations are suitable for representing any information type. For example, shape-based graphic relations are useful for telling different categories that graphic objects belong to apart (i.e. shape is an appropriate visual attribute for the representation of nominal information). Conversely, shape ought not be used to communicate quantitative information, as this results in an inefficient and confusing understanding of the diagram by the reader [Ber11].

Consequently, information about which graphic relations are appropriate to characterize which information types may be characterized through the `ddo:appropriate_for` object property of DDO. Ideally, this property would link subclasses of `vdo:Graphic_Relation` with individuals of `ddo:Information_Type`. However, this approach would be outside of OWL DL and OWL Lite (c.f. Section 3.2.4), which only allow individuals as object property subjects or objects. Therefore, in order to make DDO compliant with OWL Lite, an approach, described in [Noy17], is used in which a parallel hierarchy of instances as property subjects

is created. The individuals of the subclasses of `vdo:Graphic_Relation_Subject` represent corresponding subclasses of `vdo:Graphic_Relation` that may be employed as subjects or objects in property triples. Moreover, these individuals are semantically linked to the classes they represent by means of the `rdfs:seeAlso` annotation property. For example, in order to characterize that shape-based graphic relations are appropriate for describing nominal information, the following triple is added to DDO: (`vdo:Shape_O2O_GR_Subject`, `ddo:appropriate_for`, `ddo:Nominal_Information`). This configuration is displayed in Figure 3.16.

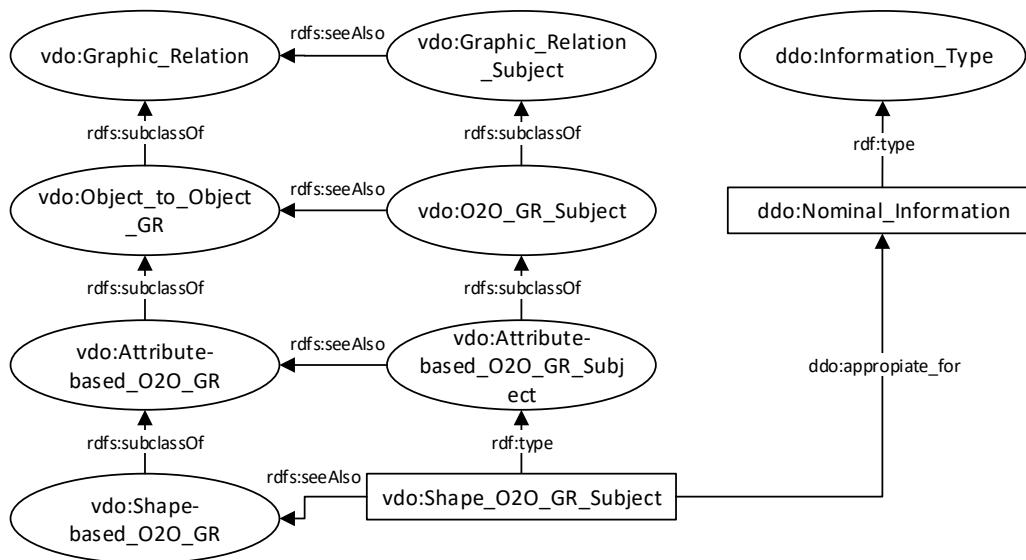


Figure 3.16: Suitability of a graphic relation to represent an information type as characterized by DDO.

### 3.9 Visualization Tasks Ontology

The Visualization Tasks Ontology (VTO) characterizes entities related to analytical and wayfinding/navigational tasks that human readers and software clients may perform in datasets visually depicted in diagrammatic representations. Analytical low-level tasks have been gathered from current pertinent taxonomies, most notably those of Amar et al. [AES05] for statistical charts, a link diagram-specific enlargement thereof by Lee et al. [LPP<sup>+</sup>06], and the classic task by data type taxonomy of Shneiderman [Shn96]. As already discussed in Section 3.5.2, this set of low-level tasks comprises the kind of queries most users make when accessing information visualization systems [AES05] which may be combined in order to conceive intentions at a higher conceptual level [CL98]. Some authors attempt to capture the decomposition of high-level tasks into sub-tasks by formally characterizing their relationships

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

[CED06]. However, VTO does not resolve to thoroughly characterize a complete vocabulary of visualization tasks, since tasks supported by vision are open-ended in nature [Ede09]. Lastly, note that tasks spanning more than one diagram in terms of their constituent elements (i.e. *inter-graphical* queries [DFVR10]) have not been considered in VTO.

The root of the taxonomic tree for the visualization task vocabulary characterized by VTO is expressed by the `vto:Visualization_Task` class of which every other task-related class inherits. Each task is characterized by VTO as an individual of some subclass of `vto:Visualization_Task`. For example, a filtering low-level task is typified as an individual, `vto:Filter_Task`, belonging to the `vto:Reading_Task` class which categorizes it in a cluster of semantically related tasks. VTO can thus be viewed as a formal taxonomy of visualization tasks more than an ontology in the strict sense (or, rather, a lightweight ontology c.f. Section 3.3), since no relationships between resources are defined beyond `is-a` RDF properties.

Throughout this section the terms ‘data case’ and ‘attribute’ will be used as prescribed by Amar et al. [AES05]. In the context of this thesis, *data cases* are individual variable instances, which are depicted in diagrams by graphic objects carrying the information object role (see Section 3.8.1). *Attributes* refer to any information of a data case that may be inferred from graphic relations such as labelling or position in a graphic space. Some examples of attributes of a certain graphic object may include a visual attribute, a textual label, or its value with respect to a metric axis.

#### 3.9.1 Reading Task (`vto:Reading_Task`)

- **Entity Type:** Class
- **Subclass of:** `vto:Visualization_Task`
- **Subclasses:** None

*Reading tasks* comprise simple analytical tasks whose goal is to retrieve certain data cases or attributes thereof according to some conditions. Reading tasks do not compare or correlate values or attributes between data cases. The following two low-level reading tasks are included in VTO:

- **Filter** (`vto:Filter_Task`): find data cases satisfying a number of given conditions. For example: “which cities have a population greater than one million?” or “which cars are black?”. Filtering is also sometimes called *zooming* in the HCI literature [Shn96].
- **Retrieve Value** (`vto:Retrieve_Value_Task`): given a number of data cases, return attributes thereof, e.g. “How many people lived in Berlin in 2004?” or “what car colors are available?”.
- **Retrieve Adjacent** (`vto:Retrieve_Adjacent_Task`): given a number of data cases, return those data cases which are immediately adjacent to them as depicted

in the diagram (either by means of connectors or separators). This task is especially commonplace in link diagrams, where knowing the neighbor nodes of the current one greatly helps in non-visual wayfinding tasks such as exploration.

Note that filtering may be regarded as the most elementary of all tasks, as it is generally performed as a precursor step to solving other tasks. For example, in the retrieve value query “how many inhabitants did London have in 2018?” it is required to first filter those data values labelled by ‘London’ and ‘2018’; only then may their attributes be fetched. AUDiaL, discussed in chapter 5, heavily leverages this during task solving by instantiating `QueryFilter` objects that refine the user’s query before resolving it. Moreover, filters may make heavy use of domain-dependent knowledge. For example, queries such as “what was the population of London two years ago?” or “what is the population of my city?” require a contingent high-level interpretation of their filters that cannot be inferred from the semantics of the diagram itself. The use of domain, user, and interest ontologies may enable reasoning by client applications about diagrammatically represented elements expressed in the user’s own terms. For a discussion on extending dialogue-enabled graphics by means of user and interest ontologies the reader is directed to [Plh14], pp. 46-51.

### 3.9.2 Comparison Task (`vto:Comparison_Task`)

- **Entity Type:** Class
- **Subclass of:** `vto:Visualization_Task`
- **Subclasses:** None

*Comparison tasks* contrast attribute values of several data cases according to the user’s needs. The comparison may take place between (filtered) data cases themselves or between data cases and an external value of interest. However, values are not aggregated, correlated, or summarized in any way. Five comparison tasks are prescribed by VTO, namely:

- **Compare** (`vto:Compare_Task`): given a number of data cases, compare attributes thereof. Example queries may include “compare the populations of London and Paris” or “does the Ford Focus consume more than 5.5 liters per 100 km?”.
- **Find Maximum** (`vto:Find_Maximum_Task`) and the related **Find Minimum** (`vto:Find_Minimum_Task`): this is the “find extremum” task defined by Amar et al. [AES05] split into two disjoint tasks for convenience. These tasks find (filtered) data cases having an extreme (maximum or minimum) value of some of their attributes. For example: “which city was the most populated in 2014?” or “which car model is the oldest?”.
- **Compute Intersection** (`vto:Compute_Intersection_Task`): some diagrammatic representations allow for the execution of set operations in the data they depict.

### 3. FORMALIZING VISUALIZATION SEMANTICS

---

Set operations in VTO include *intersection*, *union*, and *complement*. An example intersection query could be “which countries have a population greater than 10 million but smaller than 50 million?”.

- **Determine Range** (`vto:Determine_Range_Task`): given a set of filtered data cases and an attribute thereof, return the available interval of values of said attribute e.g. “what car colors are available?” or “what is the range of the population in German states?”.
- **Sort** (`vto:Sort_Task`): given a set of filtered data cases, place them in the order prescribed by one of their shared attributes. Example queries may include “sort cars according to their consumption” or “rank cities by population in 2014”.

#### 3.9.3 Derived Value Task (`vto:Derived_Value_Task`)

- **Entity Type:** Class
- **Subclass of:** `vto:Visualization_Task`
- **Subclasses:** None

*Derived value tasks* compute a numerical aggregation of data cases of interest providing a statistical summary thereof. They underpin any of the summary statistics that may be requested by the user, including measures of statistical location, dispersion, and shape. Expressed in its *pro-forma* abstract a derived value task consists on applying an aggregation function  $F$  to the corresponding attributes of a set of filtered data cases. Its most relevant individuals are:

- **Compute Average** (`vto:Compute_Average_Task`): compute the arithmetic mean of a set of attributes e.g. “what is the average population of Italy from 1990 to 2010?”. Similarly, tasks to compute other statistical measures such as mode, median, standard deviation, variation, and kurtosis are characterized in VTO, but will be skipped here for brevity.
- **Compute Sum** (`vto:compute_Sum_Task`): add up a set of attributes from a (filtered) set of data cases. For example: “what is the total population of Europe?”.
- **Count** (`vto:Count_Task`): given a set of (filtered) data cases, return the number of distinct values of a given attribute. Example query: “how many countries are there in Europe?”.

#### 3.9.4 Distribution Task (`vto:Distribution_Task`)

- **Entity Type:** Class
- **Subclass of:** `vto:Visualization_Task`

- **Subclasses:** None

*Distribution tasks* are concerned with the study of the empirical measure of certain attributes of the data cases. As opposed to derived value tasks, the answer to these tasks cannot be expressed by means of a single numerical aggregation, requiring longer verbal descriptions. The following individual distribution tasks are considered by VTO:

- **Characterize Distribution** (`vto:Characterize_Distribution_Task`): given a set of (filtered) data cases, describe the distribution of one of their attributes. This task allows users to obtain a formal high-level overview of the underlying data distribution. For example: “what is the distribution of the male population in Qatar?”. For a more general overview, the *Summary task* (described below) ought to be employed instead.
- **Find Anomalies** (`vto:Find_Anomalies_Task`): given a set of data cases, find outliers (values of a certain attribute differing significantly from the rest) therein. This is another task that gives users effective insight into the underlying data distribution. Finding outliers allows for the discovery of errors or especially significant data cases. An example query could be “are there cars with an exceptional fuel consumption?”.
- **Retrieve Summary** (`vto:Summary_Task`): this task, of a conceptual higher level to the ones discussed so far, returns a comprehensive natural language description of the whole diagram, generally by combining a number of low-level analytical tasks into an integrated report. Example queries include “Give me an overview of the diagram” or “summarize this data”. Summary approaches are domain-dependent, but generally include individual views of each variable along with a combined view of the dataset as a whole [Shn96]. A discussion of summarization strategies for diagrammatic representations is discussed in Chapter 6.

### 3.9.5 Correlation Task (`vto:Correlation_Task`)

- **Entity Type:** Class
- **Subclass of:** `vto:Visualization_Task`
- **Subclasses:** None

*Correlation tasks* deal with finding existing dependencies between data cases. *Dependency* is the statistical association between random variables characterized by certain attributes of the data cases. Making the relationship between two variables explicitly apparent is very commonly one of the reasons why data is chosen to be diagrammatically represented. Two main correlation tasks are defined by VTO:

- **Correlate** (`vto:Correlate_Task`): given a set of (filtered) data cases and two attributes thereof, try to find a predictive dependency between them. Two random

variables can be said to be dependent if they fail to satisfy a test of probabilistic independence. An example query is “what is the relationship between car weight and fuel efficiency?”.

- **Cluster** (`vto:Cluster_Task`): given a set of (filtered) data cases, partition them into similar groups according to certain attributes thereof. The clustering process may be guided by correlations among attributes of a high-dimensional space or by a similarity measure between attributes. Humans have a natural tendency towards clustering datasets into similar groups, as this may result in a number of important connotations depending on the domain at hand [AES05]. Some example queries may include “group cars according to fuel consumption” or “cluster the states of Germany by their population”.

#### 3.9.6 Navigation Task (`vto:Navigation_Task`)

- **Entity Type:** Class
- **Subclass of:** `vto:Visualization_Task`
- **Subclasses:** None

*Navigation tasks*, also known as *wayfinding tasks*, allow users to (non-visually) perform wayfinding activities, such as exploring a dataset in order to get acquainted with it in a general manner before scrutinizing it through the execution of more goal-oriented tasks (c.f. Section 6.3). As it will be discussed in chapter 6, one of the most challenging aspects of designing non-visual counterparts of diagrams is arguably the question of how to enable efficient navigation mechanisms. Navigation tasks are designed with accessibility in mind and are of lesser use for sighted persons and client applications. In this section we will merely give a quick overview of the low-level navigational tasks characterized by VTO without delving into application-specific details. For an in-depth reasoning of these tasks and the motivation for their inclusion in VTO as well as their implementation in the dialogue-enabled prototype, the reader is redirected to Section 6.1 and Section 6.3.

The term ‘node’ will be used throughout the descriptions of navigational tasks to refer to a graphic object that may be visited by the user. These include only information objects that carry meaningful information to the user in the context of diagram navigation. As such, decoration objects as well as reference objects such as metric axes or legends do not bear navigational order and may not be reached by means of these tasks.

- **Where am I?** (`vto:Where_Task`): users need to be aware of their current position at all times in order to prevent disorientation and frustration during navigation. The “where am I?” command thus needs to be implemented by client applications by providing useful landmark information about the current node such as its appropriate labels and attributes, position within the diagrammatic hierarchy (if any), closest neighbor nodes, user annotations, etc.

- **Go to First/Last** (`vto:Go_to_First_Task`, `vto:Go_to_Last_Task`): these tasks perform a quick navigational jump from the current node to the first (or last) node in the diagram according to a certain navigational order.
- **Move to Previous/Next** (`vto:Move_Left_Task`, `vto:Move_Right_Task`): these tasks allow users to sequentially navigate a diagram in a node-by-node fashion. Given a navigational order, moving to the next involves jumping to the node having the navigational index immediately following the current one, whereas moving to the previous node means jumping to the node with the immediately preceding navigational order. Note that the navigational order of a diagram does not have to follow its spatial arrangement (e.g. moving in a top-bottom and left-right fashion), even if this is the case in most applications.
- **Move Up/Down** (`vto:Move_Up_Task`, `vto:Move_Down_Task`): many diagrams depict information in a hierarchical manner. Issuing move up and down commands allow users to go one level higher and lower, respectively, in the hierarchy relative to the current node. For example, if the user is currently in a stacked bar of a bar chart, moving down allows the user to go to the first metric bar contained within the stacked bar.
- **Go to Highest/Lowest** (`vto:Go_to_Highest_Task`, `vto:Go_to_Lowest_Task`): jump to the node having the highest/lowest value of a certain attribute. By default, the chosen attribute should be the value of graphic objects with respect to the metric axes of the diagram (e.g. moving to the bar depicting the highest/lowest population in a bar chart describing population by country), but these tasks could also refer to the nodes having the extreme value of a given visual attribute (e.g. the query "go to the biggest country" in a geographical map).
- **Jump to** (`vto:Jump_Task`): move to the first one of a set of filtered nodes. For example, in a bar chart illustrating the population of Europe by country and year, the query "go to Germany" would result in moving from the current bar to the first bar depicting the population of Germany. This task is a generalization of the more specific jumping tasks discussed in this section.
- **Reset Navigation** (`vto:Reset_Task`): compute the navigational order of the diagram's nodes and subsequently move to the first one. This task may be requested in order to initialize the navigational order before carrying other wayfinding tasks for the first time or if new nodes have been added to the diagram.
- **Retrieve Visit History** (`vto:Retrieve_History_Task`): during exploration of a diagram, users generally visit a greater number of nodes than they are able to keep in their working memory. Moreover, users need to be able to identify if they are currently in a node they have previously visited. Therefore, it is important that users have the possibility of backtracking in their exploration path by issuing a "retrieve history" command in some way. The client system may then display certain information about

all previously visited nodes in order or some salient subset thereof. This helps users with navigation and with preventing disorientation during wayfinding activities.

Each time a user moves from one node to another, additional information about the jump may be supplied. For example, AUDiaL (Chapter 5) returns the textual labels associated with the newly moved to node as well as a comparison between the previous node's value and the current one. Moreover, if the jump has spanned more than two nodes, trend information is communicated to the user as well (see Section 6.6.2 for more information). Nevertheless, the tasks here defined are only concerned with the action of moving from one node to another, as any additional information is domain-dependent and, therefore, communicating it lays within the responsibilities of client applications consuming a knowledge base. Moreover, client applications may make use of supplementary ontological resources for their own navigational needs, such as the use of home nodes in AUDiaL described in Section 6.5. These constructs may be added to the topmost level of the ontological hierarchy; namely, the user/system level, which we will briefly discuss next.

## 3.10 System and User Ontologies

The bottom-most level of the hierarchical arrangement of ontologies (c.f. Figure 3.11) allows for customization of the knowledge base according to the specific needs of a user or a client application. Not every person is interested in the same aspects that may be communicated by a diagram, nor do all persons have the same needs with regard to their individual abilities. Given that resources at this level are system-dependent in nature, this section will outline how user/system ontologies have been employed in this thesis and other HCI approaches we have found in the literature but will not include any particular resources. For a description of the system/user resources employed by our Web NLI to diagrams prototype (AUDiaL) the reader is directed to Chapter 5.

Communicative Images, discussed in Section 2.6, employ two different ontology types at the user level; namely, *interest* and *user* ontologies. Interest ontologies enable dialogue systems to utilize semantic resources about a specific user's elements of interest in order to adjust their dialogue strategy accordingly. Such a vocabulary is generally built by either observing the interactions between the user and the system and automatically adjusting its user-specific weighted resources or by means of directly enquiring users about their preferences [Plh14]. The user ontology maintains knowledge directly related to user themselves, such as their personal information. This way, ambiguous expressions such as "my daughter" can be converted to more concrete expressions like "daughter of John Smith" [Plh14]. In a similar fashion to the interest ontology of Communicative Images, the SIGHT system of Demir et al. is said to include a simple user model that reflects user preferences about a diagram [DOS<sup>+</sup>10]. For example, if the system detects that many queries about a certain entity tend to be asked by a user, it should increase the *a priori* importance of the propositions related to said entity. It appears however that such customization system was not implemented, as we have not been able to find further information about it.

The work of Karim and Tjoa [KT06] is of special relevance here. Their work proposes a user ontology (named *Impairments-User* ontology) that describes the existing impairments of disabled users as well as how GUI components ought to be adapted to certain disabilities through ontology-supported mechanisms. For example, a level of visual acuity of one quarter (20/80) may be characterized in the Impairment-User ontology through an individual, OneQuarterVA, of the VisualAcuity class, itself a subclass of the Blindness ← VisionImpairment ← Impairment class hierarchy. At the same time, a given element of a user interface may have an associated AvailableText individual taking part in a textSize datatype property occurrence. By specifying a restriction on its recommendedText property according to the OneQuarterVA individual, the text size of a given element of the GUI may be automatically adapted for users having a 20/80 visual acuity. Their approach shows an interesting consolidation between user and domain ontologies in order to adapt a user interface to the user's special needs.

Formal mappings between the specific needs of a user and available interface characteristics such as those previously outlined may also be employed in the context of a Natural Language Interface (NLI) to diagrams described in this thesis. For instance, the prosody configuration of text-to-speech output could be adjusted according to the user's preferences, or the most commonly enquired about graphic objects in the ontology could be given preference over the rest when they are output in dialog boxes. Currently, formal resources at the user/system level have been included in order to support the following mechanisms. These mechanisms as well as the system/user ontological resources underpinning them will be described at greater length in their respective sections of Chapter 5 and Chapter 6.

- Personal user annotations of graphic objects
- Creation of user-defined navigational landmarks e.g. home nodes
- Specific verbalizations of tasks characterized by VTO
- System support for wayfinding tasks e.g. navigation index of graphic objects
- System support for dialogue handling between system and user e.g. ranking of ontological resources displayed in dialogue suggestions
- Highlighting tasks of special relevance e.g. the main analytical task of an input query

This brief overview of the bottom-most level of the semantic hierarchy for visualization rounds up our discussion on the formal semantics that will be employed by our client applications for authoring and accessing semantically-enhanced diagrams. Next chapter will provide a description of how to bridge the gap between the formal knowledge bases here described and the graphical primitives that constitute diagrams.

### 3.11 Summary

This chapter has tackled the design problem of formalizing visualization semantics as a prerequisite for designing an effective accessible NLI to diagrams. Such problem arose from the fact that diagrams commonly stored as raster images, proprietary object models, and other unstructured formats do not enable the realization of complex inference and analytical tasks supported by visualization (Section 3.1). In addition, whereas methods that automatically deal with simple perceptual tasks such as image classification have experienced a thrilling increase in accuracy, user tasks at a higher conceptual level, such as Visual Question Answering (VQA) in diagrams, are still not achievable by these technologies at an acceptable level. The conceptual difference between low-level image features that may be automatically extracted from the image and high-level semantics derived from the image is known as the “semantic gap” (Section 3.1.2).

The integration of a suitable vocabulary of well-known *a priori* knowledge into the primitive graphical components of a diagram enables the realization of cognitive tasks, such as question answering, on diagrammatically displayed information. Such a formal vocabulary for describing visualization semantics needs to conform to the different conceptual levels underpinning the descriptive dissonance expressed by the semantic gap as illustrated by Gestalt psychology, and, more specifically in the study of diagrams, Frege’s principle (Section 3.5.2). Ontologies, described in Section 3.2, are currently being suggested in the fields of Content-Based Image Retrieval (CBIR) and Visual Question Answering (VQA) in photographs as the means of providing the necessary *a priori* semantic knowledge to perform high-level inference on real-world image data (Section 3.3).

The problem investigation phase, outlined in Section 3.4, concluded a hierarchical set of ontologies as the most suitable means for formalizing visualization due to the inherent benefits ontologies provide when describing knowledge in this field. Namely, the recursive nature of graphics outlined by Frege’s principle is built-in in ontology engineering; and graphic objects (Section 3.6.2), graphic spaces (Section 3.6.6), and their properties and relations may be expressed in a uniform manner by means of ontological classes, properties, constraints, and other axioms. Moreover, semantic reasoners allow to simplify the annotation process of semantically-enhanced graphics by automatically inferring new knowledge from the existing one. New knowledge may also be imported to the vocabulary by means of the myriad of available domain ontologies published to the Semantic Web (Section 3.5.2). Lastly, the knowledge contained in ontologies may be efficiently retrieved by NLIs, which is the ultimate goal of this thesis.

Consequently, the treatment design phase consisted in studying the existing literature in the field of visualization in order to systematically derive a hierarchical formal vocabulary underpinning diagram semantics at their different levels of abstraction. Derived from this analysis, five hierarchically organized ontologies have been implemented (Section 3.5). Namely:

- **Upper Visualization Ontology (UVO)**, whose main resources are laid out in Section 3.6, is the upper ontology underpinning the lexicon that makes up the majority

of diagrammatic representations, including graphic objects, graphic spaces, and visual attributes.

- **Visualization Domain Ontology (VDO)**, Section 3.7, characterizes the properties of the vocabulary defined by UVQ enhancing them with the knowledge necessary for the appropriate interpretation of the diagram. Most notably, VDO describes the graphic relations semantically connecting graphic objects and graphic spaces, and the syntactic roles they may play in their relationships.
- **Data Domain Ontology (DDO)**, Section 3.8, describes the underlying data depicted by the diagram, such as statistical variables, informational roles of graphic objects, and information types, in a manner consistent with the topmost levels of the semantic hierarchy.
- **Domain Ontologies**, Section 3.5.2, may be incorporated to a knowledge base in order to enhance its graphical objects with domain knowledge related to the diagram or to automatically map uncertain elements from a user's query into formal resources of other ontologies.
- **Visualization Tasks Ontology (VTO)**, Section 3.9, represents the formal vocabulary of low-level analytical and wayfinding tasks that end users may perform on diagrammatically displayed information, supporting client applications in combining them by means of composition and superimposition in order to describe user tasks at a higher conceptual level.
- Lastly, **User and System Ontologies**, Section 3.10, are placed at the bottom-most level of the hierarchy and define resources that users and client applications may employ in order to customize the knowledge base according to their specific needs.

This chapter has outlined the principal resources contained at the different semantic levels of the proposed hierarchy for visualization, thereby concluding the design cycle of the subproblem at hand. Note that the treatment validation phase has been resolved during the treatment design phase, since it ensured that the semantic hierarchy developed met the requirements outlined in Section 3.4. The next chapter will tackle the question of how to undertake the actual semantic annotation of diagrams by means of these formal foundations i.e. how to implement the bridging of the semantic gap between graphic primitives and semantic resources.



# Semantic Annotation of Diagrams

The previous chapters have resolved the interest in integrating unstructured image formats with structured content descriptions for the realization of cognitive tasks on diagrammatically displayed information in a non-visual manner. Such interest gave rise to a new design problem; namely, how such integration ought to be performed i.e., how to effectively and efficiently undertake the semantic mapping from graphical primitives to ontological resources, a process known as *semantic image annotation*.

This Chapter is structured as follows. First, the problem is framed by analyzing existing image formats in Section 4.1 and Section 4.2. Technical means for integrating formal semantics into vector graphics are discussed in Section 4.3, where the concept of semantically-enhanced graphics is described in detail. Next, available treatments that enable the authoring of semantically-enhanced graphics are laid out during the main part of the problem investigation task (Section 4.4). Lastly, the treatment design and validation phases, with the goal of designing a tool (Semantic Annotator for Inkscape (SAI)) that allows for seamless integration of ontology resources characterized in Chapter 3 with SVG image content in a single workspace is detailed in Section 4.5.

## 4.1 Digital Image Formats

Digital images are encoded through a number of different standardized specifications, known as image *file formats*, into bits of data for storage [Tan06]. Image file formats may be generally divided into two main types:

- *Raster formats*, also known as *bitmaps*, “make use of pixels or picture elements to define an image as a matrix made up of rows and columns of pixels” [WDH<sup>+</sup>01]. In order to save storage space, some data compression mechanisms are generally applied to the image data. According to the kind of compression mechanisms they employ raster

formats may be further categorized into *lossy* (some information is lost during the data compression process) and *lossless* (the resulting decompressed image is identical to the original one). The usage of lossless compression obviously results in compressed images of a bigger size than if lossy methods were employed instead. Common raster image file formats include JPG, PNG, GIF, TIFF, and BMP.

- *Vector images* represent images mathematically through the use of geometrical primitives such as point, lines, curves, and polygons [Tan06]. This mathematical encoding may then be interpreted by a reader application in order to physically display the image on a computer screen or print it. Vector image file formats include SVG, WMF, AI, and EPS.

The main disadvantage of employing raster formats to store graphics is that they are resolution dependent i.e., the quality of the image degrades if its resolution (the amount of spatial detail [Tan06]) is altered. Consequently, their size also increases with the resolution, leading to files of a large size and a slower processing thereof [WDH<sup>+</sup>01]. On the other hand, vector formats allow for the modification of the image resolution without a corresponding downgrade in their quality. Other advantages of vector formats over raster formats include smaller file size and the property to store shapes, attributes, and, most importantly, metadata information in a standardized manner [ROE04]. The metadata capabilities of the SVG file format and their use in regard to accessibility were already discussed in Chapter 2, Section 2.1.7.

Images on the WWW are most commonly stored in raster formats, most commonly JPG, PNG, and GIF, but they have been evolving from these formats to structured and dynamic formats such as SVG [Dic11]. Moreover, SVG is the recommended image file format for Web graphics by the W3C [RKE05]. Given their prevalence among vector image file formats on the WWW and their extensive support for embedding metadata in line with the requirements of semantically-enhanced graphics, we will expand on the SVG format by briefly describing its main characteristics in the following section.

## 4.2 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is a text-based image file format that follows the SVG syntax. SVG may be included directly within any Web page or HTML document, in which case it uses the HTML syntax. Likewise, HTML markup may be embedded in SVG files. In the same way as HTML documents, SVG files may be styled with Cascading Style Sheets (CSS). In addition, SVG documents may also be dynamically modified through programs (scripts) written in JavaScript and styled with CSS rules. SVG 2 reached “candidate recommendation” stage by the W3C in 2016. The XML namespace for all SVG versions is <http://www.w3.org/2000/svg>, usually shortened to `svg`. In the following we proceed to outline the main elements that make up the SVG format

The root element of every SVG file is the outermost `svg` element, which is conceptually similar to HTML5’s `canvas` element, acting as the container, structural, and renderable

element of the SVG document as a whole. Its `width` and `height` attributes determine the size of the SVG document with pixels as the default unit. Moreover, `svg` elements may be nested. In this case, the `x` and `y` attributes of the contained `svg` elements indicate their top-left corner with respect to their container element. Container elements such as `svg` may be filled with a number of graphics elements such as rectangles, circles, paths, or text. The absolute position of these elements within their container element is specified again by means of `x` and `y` attributes with origin at the top-left corner of the container. Graphics elements include basic shapes such as circles, ellipses, and rectangles; lines and paths, text, and other elements such as audio and video elements.

Graphics elements are drawn onto a target canvas as specified by their attributes. For instance, the geometry properties of a basic `circle` include `cx` and `cy` (*x* and *y* coordinates, respectively, of its center point), and `r` (its radius). In this manner, a simple SVG image containing just a black (the default color) circle in its center could be typified as follows:

```
<svg width="500" height="500">
    <circle cx="250" cy="250" r="50"/>
</svg>
```

Each graphics element may employ different attributes so it can be drawn correctly by client applications. For example, the geometry properties of the `rect` element include the coordinates of the point at its top-left corner within a canvas, its `height` and `width`, and `radii` properties of its corners. `line` elements are expressed by the coordinates of the points at its ends, whereas `text` elements are characterized by the coordinates of their left edge and vertical position of their baseline, and a number of CSS-specified font styles. More powerful graphics elements may require an aggregation of several values into more complex attributes in order to be characterized properly. For example, a path is defined by a number of straight or curved segments joined together in its `d` attribute, whereas `polylines` and `polygons` are defined by listing a number of points which are connected by straight segments in their `points` attribute. This allows for lines and polygons of arbitrary shape to be defined with desired precision.

Graphics elements may be further styled in SVG by means of either CSS properties or specific presentation attributes. The latter are elements particular to SVG whose value is parsed as a value of their property much in the same way of geometric properties. The most commonly employed presentation attributes include `fill` (a color value saturating the whole element), `stroke` and `stroke-width` for defining an element's outline color and width, respectively; `opacity` (from full transparency to full opacity), as well as several font-related properties derived from the CSS language such as `font-size`, `font-family`, and `font-style`. However, the preferred means for styling SVG images is, similarly to HTML documents, by means of CSS markup rules, as they are easier to read and maintain. Like in CSS-styled Web pages, CSS rules may be applied directly to an SVG element through the `style` inline attribute, or they may be written externally to the content and applied through the use of CSS selectors. External style sheets may be included in a SVG document by means of the HTML `link` element with its `rel` attribute set to 'stylesheet' and `href` pointing to the desired CSS sheet, or through the alternative `@import` rule in an inline style sheet. The

drawing order of the elements in a SVG image is implicitly derived from the order they appear in the document. Therefore, in most cases graphic objects carrying a reference informational role (such as grid lines) need to be added last to the document, as otherwise informational objects (for example, the datapoints of the diagram) could be occluded, challenging their proper interpretation.

Element-specific styling is achieved by either `style` element attributes or by referencing elements by their class name in CSS style sheet rules. A class name may be assigned to one or more elements by means of its `class` attribute e.g. `<rect class="container" ... />` adds the “container” class to a specific rectangle in the SVG image. Since SVG documents follow the XML syntax, individual elements within the node tree may be uniquely identified by their `id` attribute. For example, the container rectangle introduced before could be uniquely identified as the main canvas as follows: `<rect id="mainCanvas" class="container" ... />`. In order to comply with the XML syntax an element's `id` attribute must be unique, cannot be the empty string, and must not contain any whitespace. As it will be discussed in next section, this ability to uniquely identify elements is vital for their proper actionability in the context of semantically-enhanced graphics. In addition to employing the `class` attribute, SVG elements may be conceptually grouped together by means of the `g` container element, which may contain further nested `g` elements. Grouping of graphics elements together is generally meant for re-usability as well as animation purposes, as it allows for any number of related elements to be uniquely named as a whole by means of the group's `id` attribute.

The SVG document as whole or components thereof may be given a human readable title with the `title` element, which, additionally, if the image is embedded in a Web site, behaves as the as the alternative text for the element that is displayed by screen readers. Most importantly in the context of semantically-enhanced graphics, SVG offers a metadata container element whose contents must be elements from other XML namespaces conforming with the corresponding W3C recommendation [Bra09]. This entails that a metadata element may contain any kind of (non-renderable) information from XML-compatible metadata languages including semantic vocabularies expressed in RDF-based ontologies (c.f. Section 3.2.2). Therefore, full-fledged knowledge bases may be embedded within a SVG graphic, allowing for semantic resources carrying supplementary information about the visual elements of the image to be included in the image document itself. Moreover, the formal interpretation and the graphic primitives remain decoupled, and vector graphics semantically enhanced in this manner remain renderable by any SVG reader application. This idea was first introduced by Communicative Images, described at length in Section 2.6.

This section has only given a very brief overview of the SVG format in order to acquaint the reader with its main characteristics in the context of this thesis. The most in-depth description of SVG may be consulted in the official SVG 2 specification [Wor18]. In the next section we will take a look at how SVG images and formal knowledge bases cooperate in making up the concept of semantically-enhanced graphics.

## 4.3 Semantically-Enhanced Graphics

This section delves into the notion of embedding relevant semantic knowledge about the concepts depicted in a vector image within the image file itself, enabling complex reasoning and inference activities supported by visualization. This is the idea behind so-called *semantically-enhanced graphics*. In addition, the integration of the formal semantics described in the previous chapter into SVG files in order to constitute semantically-enhanced graphics will be demonstrated.

### 4.3.1 Bridging Graphic Elements and Semantic Resources

Section 4.2 (see also Section 2.6.1) described the fact that metadata elements of SVG allows for the embedding of formal semantics following the XML syntax, such as RDF (Section 3.2.2) and OWL (Section 3.2.4) ontologies, directly into the image file. In the context of semantically-enhanced diagrams, the visualization ontologies described in Chapter 3 (namely; UVO, VDO, and DDO) may be embedded within a SVG document, as shown in Figure 4.1.

Implanting the formal characterization of the depicted objects in the image allows for their proper interpretation by client applications undertaking cognitive and inference tasks on the image data, such as NLIs, something that will be described in Chapter 5. Said characterization is known as *semantic annotation*, since regions or objects in the image are annotated with the addition of high-level semantics or attributes thereof. The types of information addressed by annotations arise from their intended context of usage. Accordingly, semantic image annotations can be divided into four different dimensions with regard to the type of metadata they characterize [DGL<sup>+11</sup>]:

- *Content descriptive annotations* addressing subject matter information. For instance, annotations that characterize graphic objects and their syntactic roles in a diagram, as described in Section 3.7.14, belong to this category.
- *Structural annotations* that describe spatial, temporal, and spatio-temporal decomposition aspects, such as the  $x$  and  $y$  coordinates of a graphic object in the SVG canvas.
- *Media annotations* referring to low-level features. Examples of structural and media annotations could be visual attribute occurrences of the graphic objects in a diagram (c.f. Section 3.6.9), as well as the geometry properties of SVG elements.
- *Administrative annotations* carrying metadata about the annotations themselves such as creation date, author, etc. Ontology resources at the user/system level of the semantic hierarchy described in Section 3.10 belong here.

The formal characterization of image semantics may therefore be achieved at all dimensions by instantiating the available visualization resources in a corresponding manner on a separate

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

---

ontology grounded on UVO (Section 3.6) and its descendants, followed by mapping the graphic elements of the image to formal resources of the embedded knowledge base. Since any SVG element may be added an unique identifier by means of its `id` attribute (as introduced in Section 4.2), the `uvo:hasSVGElement` datatype property of UVO (c.f. Section 3.7.1) allows to semantically connect formal visualization resources to SVG elements.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:svg="http://www.w3.org/2000/svg"
    xmlns="http://www.w3.org/2000/svg"
    width="600"
    height="400"
    sodipodi:docname="Winery Bar Chart">

    <!-- Other metadata (e.g. Inkscape namedview) here -->

    <metadata id="upper_visualization">
        <!-- Embed UVO ontology -->
        <rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
            xml:base="http://www.w3.org/2002/07/owl"
            xmlns:uvo="..."
            xmlns:owl="http://www.w3.org/2002/07/owl#"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:xml="http://www.w3.org/XML/1998/namespace"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
            <Ontology rdf:about="..."/>

            <!-- Graphic Object declaration -->
            <Class rdf:about="uvo#Graphic_Object">
                <rdfs:subClassOf rdf:resource="uvo#Graphic_Thing"/>
                <!-- Rest of uvo#Graphic_Object definition -->
            </Class>

            <!-- Rest of ontology here -->
        </rdf:RDF>
    </metadata>

    <!-- SVG elements of diagram here -->

</svg>
```

Figure 4.1: Embedding UVO in a SVG image.

Figure 4.2 illustrates a semantically-enhanced diagram, more specifically a bar chart, depicting wine sales in a winery. The first three metadata sections of the SVG document embed the Upper Visualization Ontology (UVO), Visualization Domain Ontology (VDO), and Data

Domain Ontology (DDO), respectively. These in turn allow for the semantic grounding of the diagram-specific formal resources appearing in the diagram ontology embedded in the following `metadata` element. As an example, a stacked bar of the bar chart is shown which is made up of two constituent metric bars. The metric bars are displayed as two `rect` SVG elements in the image with `id` attributes `rect3784` and `rect3785`, respectively. Element ID attributes are usually automatically generated by authoring applications such as Inkscape, and therefore no user action is required during the development of the diagram. In addition, an individual of the `uvo:Composite_GO` class (Section 3.6.5) formally characterizes the stacked bar in the supporting knowledge base (shown in Figure 2 in RDF/XML syntax). This individual is mapped to the SVG graphics elements it formalizes by participating in two corresponding occurrences of the `uvo:hasSVGEElement` datatype property. The objects of these occurrences, of datatype `xsd:string`, must match the `id` attributes of the stacked bar's constituent SVG elements.

Besides the corresponding graphics elements that the individual depicts, Figure 4.2 also displays additionally inferred information about the stacked bar that has been added to the formal knowledge base by means of properties. The computed coordinates of its center have been inserted by averaging the center from its constituent elements by means of the `uvo:hasXCoordinate` and `uvo:hasYCoordinate` object properties. Moreover, its length and width properties have also been included, the height of a vertical stacked bar being the sum of the height of its constituent metric bars, whereas its width remains that of its children. The stacked bar also takes part in two labelling graphic relations (as the node being labelled) as shown by the two `vdo:is_labeled_by` property occurrences. Lastly, its syntactic role as characterized by VDO (stacked bar) as well as its informational role as specified by DDO (information object) are also included in the diagram ontology.

As described in Section 3.7.1, graphic relationships between graphic objects may be characterized in the knowledge base by means of instances of the `uvo:Graphic_Relation` class and its subclasses. These instances are embedded in the diagram ontology in a similar manner to the graphic object individual we have just seen. In this case, the `uvo:hasSVGEElement` property occurrences where the instance takes part underpin the SVG elements of the graphic objects taking part in the graphic relation. Alternatively, some graphic relations may be directly instantiated by means of object properties, such as the labelling relationships depicted in Figure 4.2 through `vdo:is_labeled_by` property occurrences (c.f. Section 3.7.12).

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

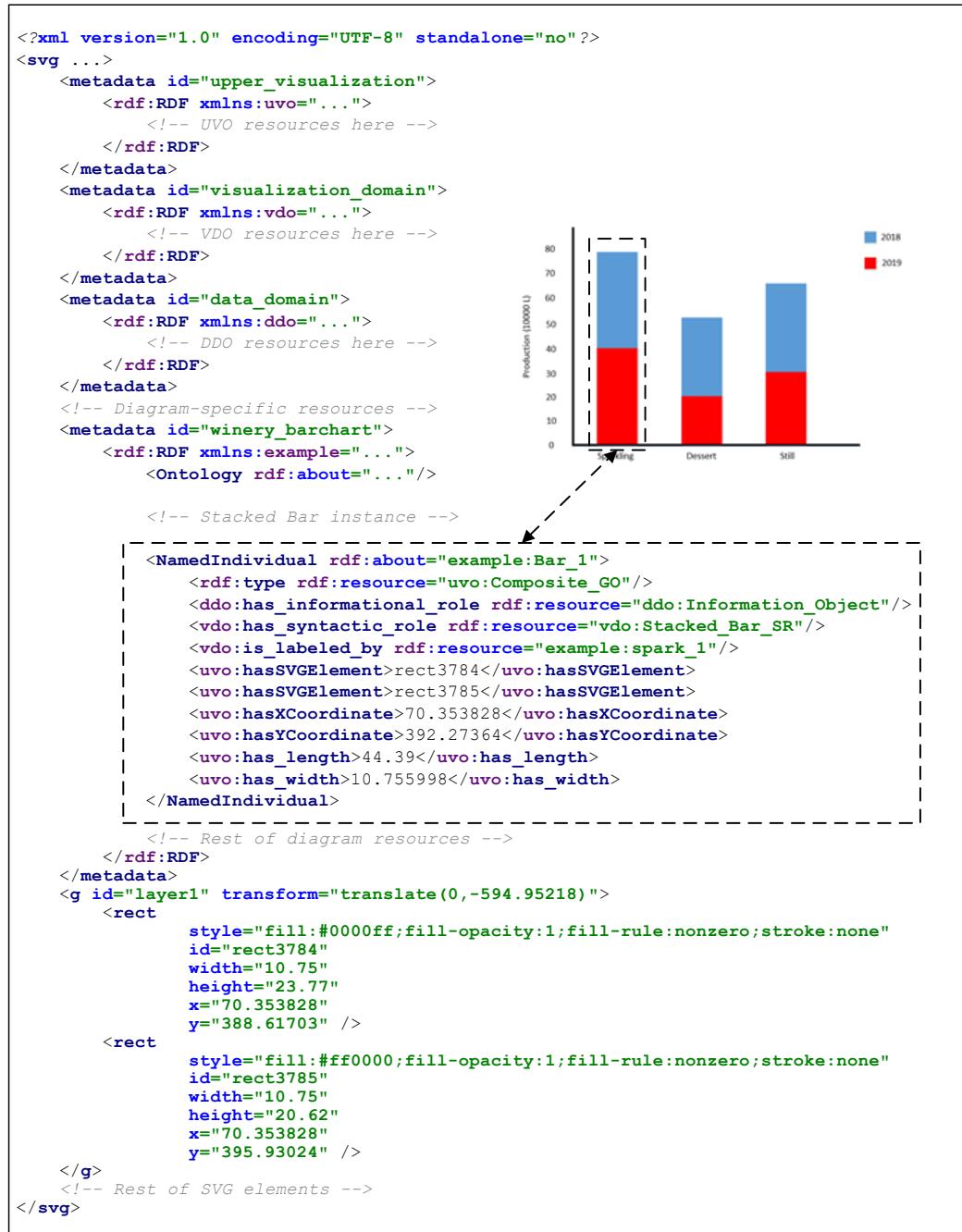


Figure 4.2: A *semantically-enhanced diagram*: formal definitions of a diagram's constituent elements are embedded in a SVG image file.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg ...>

    <!-- Visualization Ontologies (UVO, VDO, DDO) go here -->

    <!-- Import Wine Domain Ontology -->
    <metadata id="ucdavis_wine">
        <owl:Ontology rdf:about="http://library.ucdavis.edu/wine-ontology#">
        <owl:versionInfo>1.0</owl:versionInfo>
        <rdfs:comment>UC Davis Wine Ontology</rdfs:comment>
        <owl:imports rdf:resource="http://library.ucdavis.edu/wine-ontology"/>
    </owl:Ontology>
    </metadata>

    <!-- Diagram-specific resources -->
    <metadata id="winery_barchart">
        <rdf:RDF xmlns:example="...">
            <Ontology rdf:about="..."/>

            <!-- Stacked Bar instance -->
            <NamedIndividual rdf:about="example:Bar_10">
                <rdf:type rdf:resource="uvo:Composite_GO"/>
                <ddo:related_to rdf:resource="http://library.ucdavis.edu/wine-ontology#SparklingWine"/>
                <!-- Rest of properties of individual -->
            </NamedIndividual>

            <!-- Rest of diagram resources -->
        </rdf:RDF>
    </metadata>

    <!-- SVG elements go here -->
</svg>

```

Figure 4.3: Importing an external ontology into a semantically-enhanced diagram.

### 4.3.2 Importing External Ontologies

Embedding a full-fledged knowledge base including domain ontologies in a semantically-enhanced graphic may however be impractical due to the potential sheer size of the interconnected ontologies. Shipping millions of semantic triples along with each semantically-enhanced graphic would result in SVG files of unreasonable size which would greatly hinder their application. Moreover, changes to the domain ontologies, such as new knowledge being incorporated or wrong information corrected, would not be reflected in the image file. Therefore, domain, visualization, and even specific diagram ontologies may be hosted externally to the graphic and imported therein by means of `owl:imports` statements, part of the vocabulary expressed by all major dialects of OWL, included within the ontology header. As an illustrative example, let us consider an individual metric bar (a `rect` SVG element) in the same bar chart we have just examined that depicts the production of sparkling wine produced by a winery in a given year. In addition to the visualization ontologies described in Chapter 3, the UC Davis Wine Ontology<sup>1</sup> could be imported into the semantically-enhanced diagram in order to embed detailed information about the wines depicted in the diagram. The SVG shape characterizing a metric bar could then be enhanced with axiomatic semantics from the UC Davis Wine

<sup>1</sup>Available at <https://github.com/UCDavisLibrary/wine-ontology/blob/master/wine-ontology.owl>

domain ontology by means of `ddo:related_to` object property triples (Section 3.8.3), as shown in Figure 4.3.

### 4.3.3 Annotation of Raster Graphics

So far we have just discussed vector graphics; likewise, raster images may be semantically enhanced adhering to this approach by following the a similar methodology to that proposed by Kopeček and Ošlejšek in their GATE framework [KO08a]. Since the graphic objects making up the image may not be directly inferred from graphics in raster format (as opposed to vector graphics, where graphics primitives are part of the image data), a pre-processing phase needs to take place first whereupon the image is segmented into areas of interest (for more information, see Section 2.6.4). These areas are then marked with invisible SVG geometries depending on their shape. The final annotated picture consists of the raster image overlaid with an invisible hierarchy of annotated regions. Whereas the SVG graphic elements of a semantically-enhanced will be invisible for the reader, they may be employed to associate its regions with formal resources from an embedded knowledge base as we have previously discussed in this section. Current methods for pre-processing of raster graphics that lay the groundwork for their semantic annotation will be briefly discussed later in this chapter.

### 4.3.4 An Illustrative Example

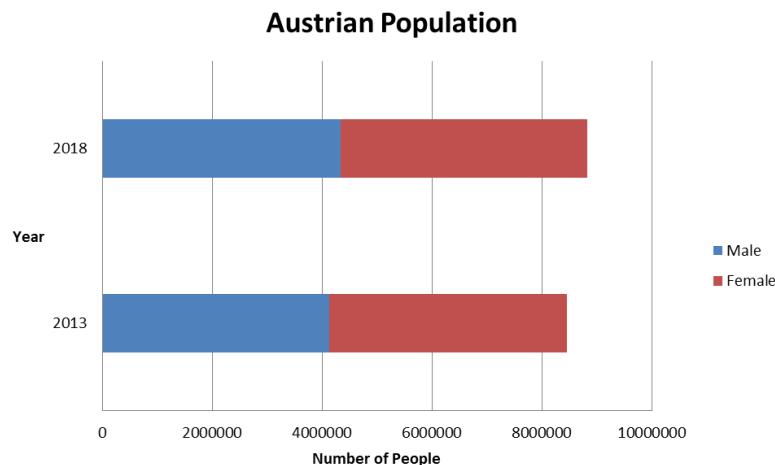


Figure 4.4: A simple SVG stacked bar chart showing the total population of Austria, divided by gender, for the years 2013 and 2018.

This section outlines, for illustrative purposes, the partial process of semantically annotating a simple statistical chart in SVG format by linking its graphical primitives to instantiated resources of the knowledge base introduced in Section 3.5. This process, which some authors

have named “bridging the semantic gap” [KJ18a], is generally performed manually by experts with specialized authoring tools, known as *semantic image annotators*, that will be presented in Section 4.4.5.

The base diagram that will be semantically annotated is shown in Figure 4.4. It is a simple bar chart that displays the total population of Austria for the years 2014 and 2018. Each bar, in turn, is a stacked bar further divided into gender ('Male' and 'Female'). The diagram has a single metric axis (marked with grid lines and numerical labels at given intervals), which depicts the 'Number of People' quantitative variable. In addition, each stacked bar is labeled by an ordinal variable ('Year'), and each constituent metric bar is labelled with a gender via a color-based object-to-object graphic relation with a corresponding key in the diagram's legend. Lastly, the diagram as a whole is labelled by a title ('Austrian Population').

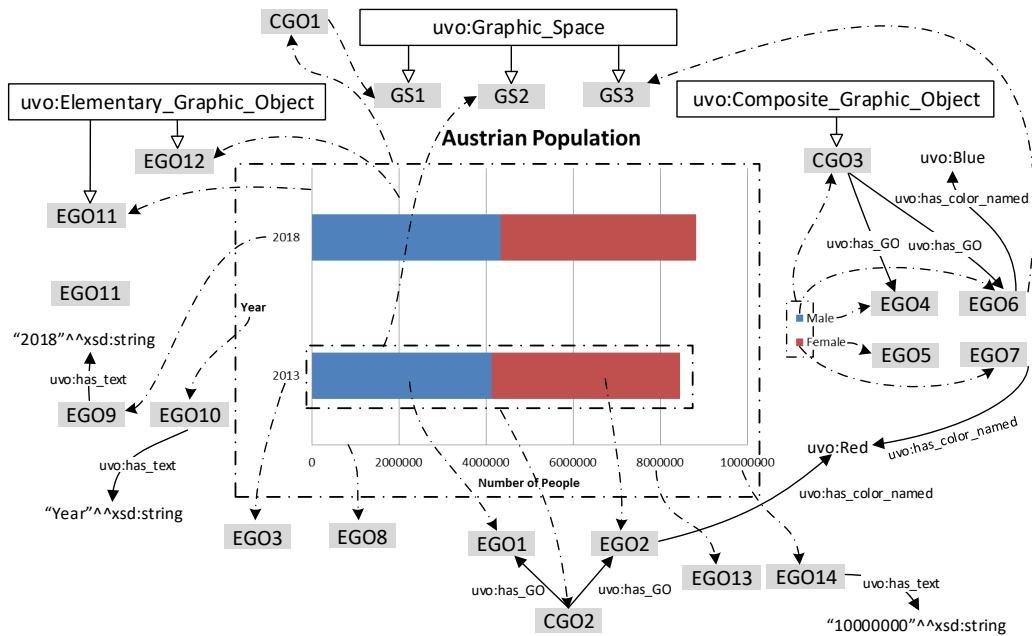


Figure 4.5: Semantic annotation of some selected diagram elements with resources from UVO. Grey boxes represent named UVO class instances. Dashed boxes group SVG shapes into composite graphic objects. Dashed arrows indicate association between SVG shapes and formal resources of the UVO. Solid arrows represent other property occurrences. Note that only a subset of all possible annotations are shown in this figure.

Figure 4.5 depicts the formal relationship between a number of selected constituent elements of the diagram from Figure 4.4 and resources from UVO. Recall from Section 3.5.2 that UVO characterizes the foundational visual primitives of the diagram. Namely, several Composite and Elementary Graphic Objects present in the bar chart are shown being associated to instances of corresponding UVO classes. For example, each stacked bar corresponds to a composite

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

graphic object instance, whereas their constituent metric bars are characterized by elementary graphic objects. Such association takes place by means of `vdo:hasSVGEElement` datatype property occurrences, which relate ontological resources (triple's subject) to SVG shapes as given by their `id` attribute (triple's object). Semantic Annotator for Inkscape (SAI), described in Section 4.5, is a semantic image annotator that has the ability of automatically generating these and other necessary property occurrences in a transparent manner to the human annotator. Three Graphic Spaces are additionally displayed in Figure 4.5. These are GS1, an Arbitrary Graphic Space in which the main diagram and its legend reside; GS2, an Elementary Metric Space in which one dimension (the length of the bars) is subject to strict interpretation; and GS3, a Meaningful Graphic Space that hosts the constituent elements of the diagram's legend.

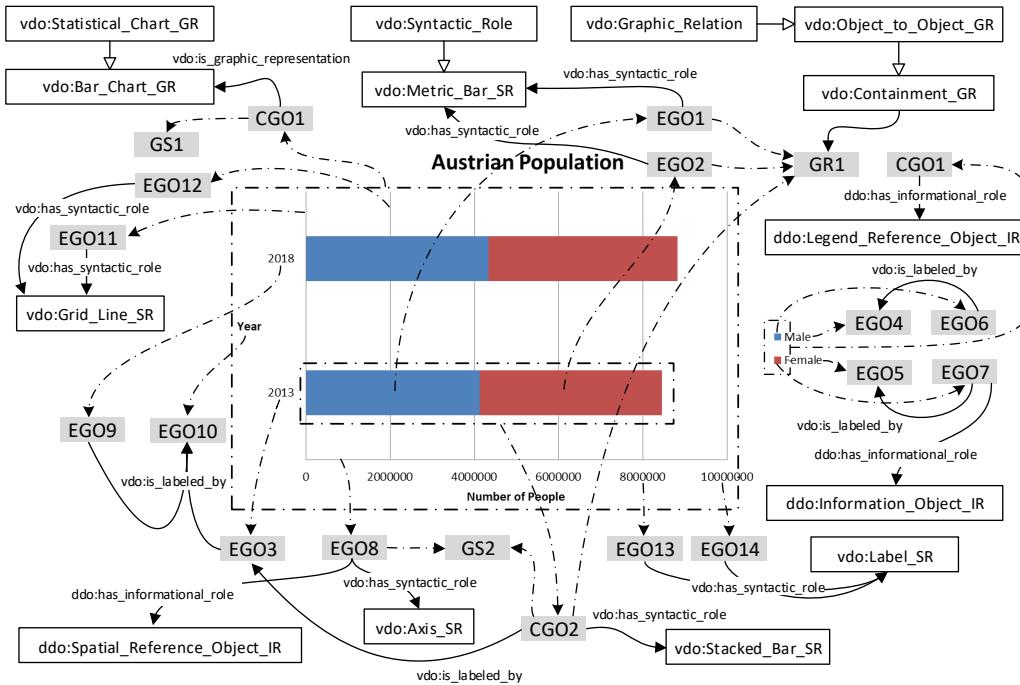


Figure 4.6: Semantic annotation of some selected diagram elements with resources from VDO and DDO. These include Graphic Relations, and Syntactic and Informational Roles of Graphic Objects, among others. This figure follows the same format as Figure 4.5. Note that only a subset of all possible annotations are shown in this figure.

Example annotations involving the next two levels of the conceptual hierarchy (namely, VDO and DDO) in the bar chart are shown in Figure 4.6. These include Graphic Relations between Graphic Objects (e.g. the containment Graphic Relation GR1 or a number of labeling relations), Syntactic Roles of Graphic Objects taking part in Graphic Relations (e.g. the `vdo:Label_SR` role used for labels), and Informational Roles of Graphic Objects, among others. Note that

labeling is a special kind of Graphic Relation which, unlike most Graphic Relation instances e.g. GR1, is not characterized by instantiating a subclass of the `vdo:Graphic_Relation` class. This is because labeling is a relationship of a higher conceptual level than other Graphic Relations underpinned in VDO that can be derived from a basic object-to-object Graphic Relation between the label and the node it labels. Most commonly, this foundational Graphic Relation is either one of clustering, where the label is placed in close proximity to the labeled nodes; or one of linking, where the label and the nodes it labels are connected by connectors (c.f. Section 3.7.12).

This example also serves to illustrate how UVO and VDO leverage the so-called principle of compositionality of meaning, also known as *Frege's principle*, which defines graphic syntax in a recursive manner such that a collection of perceivable objects arranged in a given structure in two-dimensional space may function as a single graphic object at a higher level of abstraction. Frege's principle is characterized in detail in Section 3.5.2. In the example graphic of Figure 4.4, the topmost Graphic Object of the diagram (not shown) encompasses the bar chart as whole. In turn, it can be decomposed into a Graphic Space (GS1), and a set of Graphic Objects; namely, the bar chart itself (CGO1), its legend (CGO3), and its title (not shown). If we now choose to decompose CGO3 one more level, it is made up of a Graphic Space (GS3), and four Elementary Graphic Objects (EGO4, EGO5, EGO6, and EGO7), which are taking part in two labelling Graphic Relations (not depicted in Figure 4.6). Each Composite Graphic Object may be decomposed in this manner until all atomic Elementary Graphic Objects that make up the graphic (corresponding to basic SVG shapes) are reached.

Resources of the next level of the hierarchy, namely VTO, cannot be related to Graphic Objects since they underpin tasks that users may perform on the depicted data, not data itself or its visualization. Tasks characterized by VTO are subclasses of the `vto:Visualization_Task` class, which include both analytical and navigational tasks that can be supported by diagrammatically displayed information (see Section 3.5.2 for detailed information). Given the expressive limitations of Description Logics such as RDF, VTO merely characterizes tasks, but it is not able to execute or process them for a specific diagram being consumed by a client application. End applications, such as AUDiaL (Chapter 5), may employ their own formal languages to action semantically-enhanced graphics. In order to do so they may have to define additionally their own RDF constructs at the User/System conceptual level depicted in. For example, AUDiaL makes extensive use of a proprietary datatype property (`aud:task_has_verbalization`) in order to associate VTO tasks to phrases in natural language. If a user employs a phrase appearing in the subject of an occurrence of this property, its corresponding task may be executed accordingly, see Section 5.9 for details.

Once a diagram has been converted from its raw raster or vector form to a semantically-enhanced diagram following this process, client applications that operate on the semantically-rich ontological underpinnings of the graphic may perform complex inference and cognitive tasks. This resolution will be the focus of Chapter 5. Next in this chapter, tools and methods for authoring semantically-enhanced graphics are studied. The whole range of existing approaches, going from purely automatic to purely manual methods, will be analyzed in detail.

## 4.4 Authoring Semantically-Enhanced Graphics

Earlier in this chapter we have discussed the idea of semantically-enhanced graphics i.e. augmenting vector image data with formal semantics in the form of ontologies, but specific methods for creating semantically-enhanced graphics have not been examined yet. The process of authoring images and ontologies is traditionally performed in a disjoint manner, as these two procedures are more often than not unrelated in conventional use cases. This section sets out to study current and prospective methods that aim to map this gap between authoring (vector) graphics and formal knowledge bases. It therefore corresponds to the Problem Investigation phase of a new design problem; namely, how to efficiently author semantically-enhanced diagrams. We will first give a brief overview on the most prevalent tools and methods for authoring vector graphics and ontologies separately. Next, current methods allowing for the addition of semantics to images including both manual and automatic semantic tagging mechanisms will be discussed. Finally, Section 4.5 will characterize Semantic Annotator for Inkscape (SAI), a solution developed within the framework of this thesis allowing for seamless addition of formal annotations to vector graphics.

### 4.4.1 Software Tools for Vector Graphics Design

The most commonly employed tools for authoring SVG images are Inkscape<sup>2</sup>, Adobe Illustrator<sup>3</sup>, Apache Batik<sup>4</sup>, Raphael JS<sup>5</sup>, Snap.svg<sup>6</sup>, and Google Docs Drawings<sup>7</sup> [SDP<sup>+</sup>19]. Among them, Inkscape and Adobe Illustrator are the most mature professional authoring tools for SVG graphics, whereas the rest of the solutions listed are toolkits, libraries and APIs meant for usage by software developers and technical users and are thus not suitable for the needs of most end-users. Therefore, we will lay the focus of this section on comparing Inkscape and Adobe Illustrator for the purposes of potentially authoring semantically-enhanced graphics.

#### Adobe Illustrator

*Adobe Illustrator* is a professional, commercial vector graphics editor developed by Adobe, Inc. Its latest version, known as Adobe CC, is sold on a subscription-based service model over the Internet. Illustrator offers SVG support as of version 9; however, it uses a proprietary implementation not fully compatible with that of Inkscape and other authoring tools. The preferred image format for graphics authored with Illustrator remains the proprietary Adobe Illustrator Artwork (.ai) format. Its GUI consists of a main toolbar (known as the basic toolbar) on the left-hand side of the screen containing the available tools to work with the document (such as create/select/manipulate graphic objects and paint/draw/sample images). Moreover, additional toolbars may be added to the staging area, called advanced toolbars. Advanced toolbars include all available tools in Illustrator. Furthermore, tools are

---

<sup>2</sup><http://www.inkscape.org/>

<sup>3</sup><http://www.adobe.com/products/illustrator/>

<sup>4</sup><http://xmlgraphics.apache.org/batik/>

<sup>5</sup><http://www.raphaeljs.com/>

<sup>6</sup><http://www.snapsvg.io/>

<sup>7</sup><http://www.google.com/google-d-s/drawings/>

categorized into select, draw, type, paint, modify, and navigate tools. Compared to other vector graphics authoring tools, Illustrator offers a wider array of image manipulation tools including gradient meshes, multiple per-object strokes and fills, color management for print, PMS color, perspective transform, and blends<sup>8</sup>.

The capabilities of Illustrator may be expanded with the installation of third-party plug-ins. Available plug-ins offer a large amount of extra functionality such as expanded 3D design, CAD tools, apparel and textile design, cartography tools, JavaScript programs for dynamic content modification, and additional patterns and textures, among many more. However, we have not been able to locate any plug-in that allows to incorporate external semantic metadata to the working document, in the form of ontologies or otherwise. Furthermore, when outputting a graphic as SVG with Illustrator, the resulting file is not optimized in any way, and moreover its elements' `id` and `class` attributes may be erased depending on the version employed, which greatly hinders the posterior conjunction of its elements with semantic resources as described in Section 4.3.1 of this chapter. Therefore, Adobe Illustrator has been discarded as an advisable software suite for authoring graphics meant to be embedded with external formal semantics.

## Inkscape

A modern alternative to Adobe Illustrator is *Inkscape*, a free and open source graphics editor offering state-of-the-art vector drawing [SDP<sup>+</sup>19] that handles SVG images natively, storing specific data with elements and attributes in custom namespaces (namely, `inkscape` and, for reasons of backward compatibility, `sodipodi`) conforming to the SVG standard. This way, SVG files saved with Inkscape render identically whether they are interpreted by Inkscape or some alternative SVG reader application any may be directly embedded on Websites and properly understood by all major Web browsers. Inkscape is the only professional-level vector graphics editor which is fully open source and cross-platform [Kir09].

In a similar fashion to Adobe Illustrator, Inkscape's window area displays a working area, known as the *canvas*, in the central area of the screen, as well as a number of tools (grouped in *toolbars*) and controls at the edges of the canvas, see Figure 4.7. *Tools* allow to create, select, and modify vector objects. Moreover, the canvas is surrounded by *rulers* that aid with object positioning and *scrollbars* for navigation of the canvas.

Inkscape offers a number of tools which are of great use for authors of semantically-enhanced graphics. Firstly, as opposed to Adobe Illustrator, Inkscape allows the SVG source of the document to be edited directly on its main workspace, which allows metadata tags and other non-renderable markup to be easily edited. In addition, some authoring tools offered by Inkscape but not supported in Illustrator include clones on canvas, keyboard shortcuts to move, rotate, and scale by screen pixels, using shapes as objects, editing gradients with gradients on-canvas, editing SVG nodes by means of the keyboard (making Inkscape a more accessible solution compared to Illustrator), and color painting over objects, among others.

---

<sup>8</sup>A comprehensive list of Illustrator's tools can be consulted at <https://helpx.adobe.com/illustrator/tutorials.html>

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

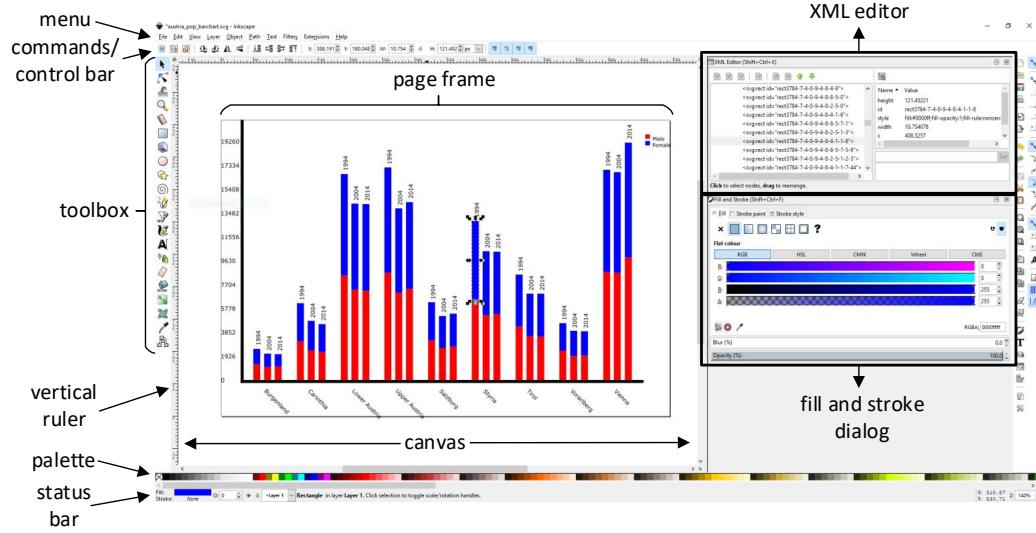


Figure 4.7: Main elements of Inkscape’s Graphical User Interface (GUI).

Most importantly, being free and open source software, developers are actively encouraged to enhance Inkscape’s functionality by means of third-party extensions. *Extensions* are programs, mostly implemented as Python scripts, that can be run from within Inkscape and offer some specific functionality, such as embedding of L<sup>A</sup>T<sub>E</sub>Xmarkup, extended search functionality, or automatic shape generation, among a myriad of other potential use cases. Extensions may modify any path data within the current document, write to/read from external files, as well as install and make use available Python packages. Extensions are generally developed as a means of adding features that may be of limited use for the general community and thus do not grant their inclusion within the standard feature set of Inkscape. Extensions may be divided into six main types, namely [LMS19]:

- *Input* extensions import data from an external file into Inkscape’s workbench. This includes opening SVG files in Inkscape, which is implemented as an internal extension.
- *Output* extensions export data rendered in Inkscape into a file. For example, saving as SVG is implemented as an internal extension in Inkscape. Both input and output extensions may be chained in order to create complex file processing pipelines.
- *Effect* extensions take a document or a selection thereof and alter it in some meaningful way. For example, an effect extension may alter the color of the renderable elements of a SVG document.
- *Print* extensions provide functionality that allows Inkscape to render the document into the extension in order to output it to external devices.

- *Path Effect* extensions provide a direct means for altering paths by adding effects to them such as including hash or other kinds of marks.
- *Tools* allow users to perform a specific drawing function such as creating a path or a geometry. New tools may be added to Inkscape's toolbars by means of extensions.

Because of its capabilities for authoring vector diagrams of arbitrary complexity, wide acceptance, free cost, ability to directly modify source XML code, and ease of extension, we selected Inkscape as the preferred authoring tool for semantically-enhanced graphics. However, at the moment of writing this document there were, to the best of our knowledge, no available Inkscape extensions allowing the embedding of semantic markup to SVG documents in a user-friendly manner, the only available means of doing so being manually editing the source code of the graphic in the XML editor (see Figure 4.7). Manually editing the source code is a slow, cumbersome, and error-prone process that would render the authoring of semantically-enhanced graphics impractical. Therefore, we set out to develop an authoring tool for semantically-enhanced graphics implemented as a Inkscape extension that allows Inkscape users to assign semantic markup to SVG elements in an integrated, user-friendly, and efficient manner. The outcome of this resolution is Semantic Annotator for Inkscape (SAI), which is described in detail in Section 4.5.

#### 4.4.2 Software Tools for Ontology Design

A number of software tools enabling authoring of formal schemas and ontologies, also known as *Semantic Web editors*, are available. Some of these include Protégé<sup>9</sup>, SWOOP<sup>10</sup>, and the NeOn Toolkit<sup>11</sup>. Among them, the most widely employed tool for designing complex RDF/OWL2 ontologies is Protégé [Mus15], developed by the Stanford Center for Biomedical Informatics Research. Protégé is the *de facto* standard in desktop ontology design, having been classified as a “killer app” i.e. one among “highly transformative technologies that create new markets and widespread patterns of behavior” [AOS05]. Due to the ubiquity of Protégé, most of its alternatives either have been abandoned, are updated less often, or offer much reduced capabilities in comparison. As such, Protégé offers its users the widest available array of functionality for ontology engineering, such as a configurable components layout, consistent rendering of ontology resources, alternate views of the same ontology, a Java-based Application Programming Interface (API) for building knowledge-based tools and applications [Ala13], loading of multiple ontologies into a single workspace, ontological refactoring tools, navigation history and search, support for a number of reasoner tools, and many more. Figure 4.8 shows Protégé’s workspace in the process of editing an individual of UVÖ’s (Section 3.6) Elementary Graphic Object class.

Protégé offers two main approaches to the modelling of ontologies: namely, these are the *Protégé-frames* editor and the *Protégé-OWL* editor. The former allows users to design

---

<sup>9</sup><https://protege.stanford.edu/>

<sup>10</sup><http://code.google.com/p/swoop/>

<sup>11</sup>[http://neon-toolkit.org/wiki/Main\\_Page.html](http://neon-toolkit.org/wiki/Main_Page.html)

## 4. SEMANTIC ANNOTATION OF DIAGRAMS

---

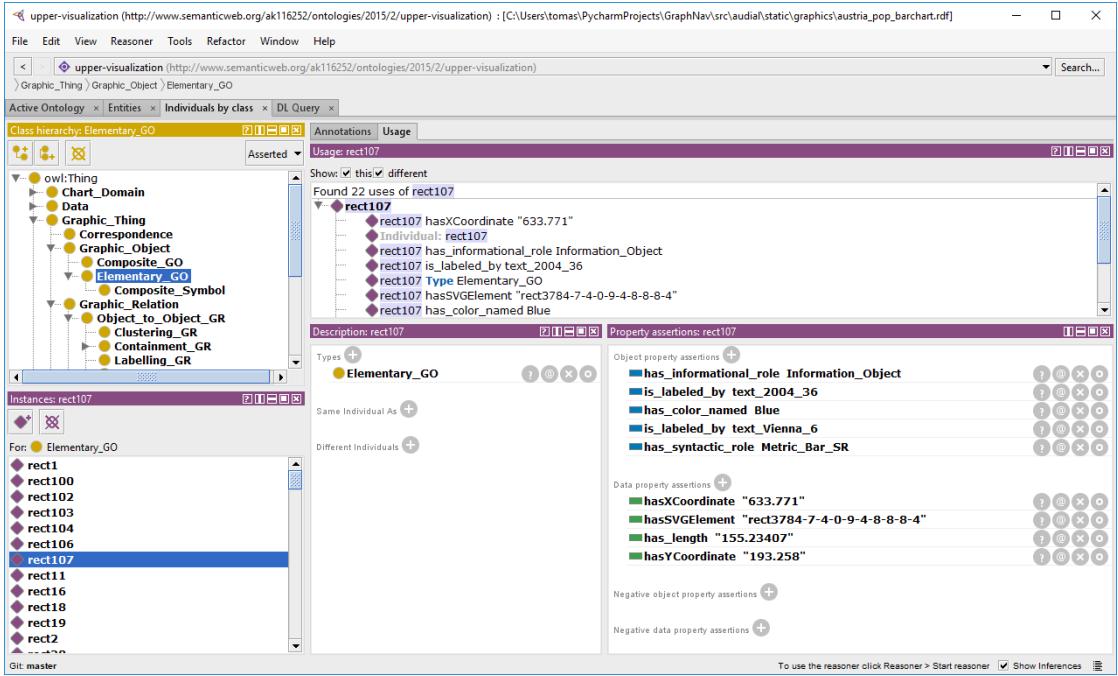


Figure 4.8: A screenshot of Protégé’s main workspace showing an individual of the `uvo:Elementary_GO` class (a bar in a bar chart) being edited.

frame-based ontologies following the Open Knowledge Base Connectivity (OKBC) protocol, whereas the Protégé-OWL editor is meant for the design and implementation of ontologies for the Semantic Web following the OWL syntax already outlined in Section 3.2.4 and that the suggested hierarchy of ontologies for visualization described in Section 3.5 adheres to. Protégé is free and open-source, and is supported by a big number of research institutions, governmental bodies, and private users. Consequently, the most significant advantage of Protégé over its competitors is its scalability and extensibility with the inclusion of third-party plug-ins [Ala13]. The most popular kind of plug-ins for Protégé are the so-called workspace tab plug-ins, which may be accessed through a new tab in the main user interface. Plug-ins extend the default capabilities of Protégé by adding features such as advanced ontology visualization, ontology merging, version management, automatic reasoning and inference, and many more [Ala13].

To the best of our knowledge, neither Protégé nor any other of the available Semantic Web tools and plug-ins allow for the import and edition of vector graphics in order to map their elements to semantic resources. On the one hand, creating a Protégé plug-in for such purpose would be possible. On the other hand, this would entail the design and development of a full-fledged vector graphics editor embedded within Protégé, including tools for creating and editing SVG elements and their attributes, handling of their positioning and shaping within the canvas, and a myriad of other capabilities. Undertaking such a task would require an enormous deal of time and resources. Conversely, the creation of a tool to map already

existing vector elements to semantic resources arises as a much more manageable problem to tackle, as existing libraries for popular programming languages allow for efficient handling of semantic resources by client applications without the commitment to developing much of the needed functionality. As such, we focused our design efforts into the development of a semantic plug-in for Inkscape (a vector graphics editor) rather than for Protégé (an ontology editor). This approach will be exhaustively outlined in Section 4.5 of the current chapter. Nevertheless, due to the ease of use and strong capacities of the Protégé-OWL editor as an authoring tool for developing OWL ontologies, we have employed it during the design and development phases of the hierarchical set of ontologies described in Chapter 3.

#### 4.4.3 Automatic Semantic Annotation

The previous two sections have discussed current manual, disjoint tools that enable mapping of vector graphic primitives to ontological resources by first authoring the graphic content, followed by enhancing it with semantics from a supporting ontology. We will now proceed to outline the opposite end of the spectrum of semantic graphic annotation. Namely, how to automatically perform such mappings through Artificial Intelligence (AI) methods. Given that current methods have mostly laid their focus on reasoning about the content of real-world images, whereas this thesis is concerned with knowledge representation graphics, this section will merely outline the current state of the art in *automatic semantic segmentation* of images, the inclusion of higher-level *a priori* knowledge to improve their results, and will discuss whether these methods may be suitable to performing the semantic annotation of diagrams suggested in this chapter.

A great deal of research has been undertaken in the past two decades with the ultimate goal of building general purpose object detection systems with capabilities rivalling those of sighted humans [LOW<sup>+19</sup>]. Traditionally, *object detection* had been addressed using well-established algorithmic computer vision techniques with handcrafted feature descriptors such SIFT, SURF, BRIEF, etc.; these algorithmic methods are simply referred to as Computer Vision (CV) in the literature [OCC<sup>+19</sup>]. CV methods are generally preceded by a step called *feature extraction*, in which small descriptive or informative image patches (known as *features*) are algorithmically computed. Feature extraction may be performed by using many well-established CV techniques such as edge detection, corner detection, or threshold segmentation [OCC<sup>+19</sup>]. The extracted features from an image form a definition, known as *bag-of-words*, for each object class present in the image. These definitions may then be also searched for in new images, and, if there is a significant match between the features from one bag-of-words in an unseen image, then the image is considered to contain instances of the object defined by said features. Obviously, the most difficult step of these methods is the manual definition of the relevant features, something that may be disregarded by employing modern machine learning methods [OCC<sup>+19</sup>], which will be discussed later in this section.

Another problem tackled by algorithmic CV and closely related to object detection is that of *image segmentation*. Image segmentation consists on deriving low-level features from raw pixel data arriving at region-based image representations, where regions are defined as “connected parts of an image which share a common set of properties” [Tow04]. These shared properties

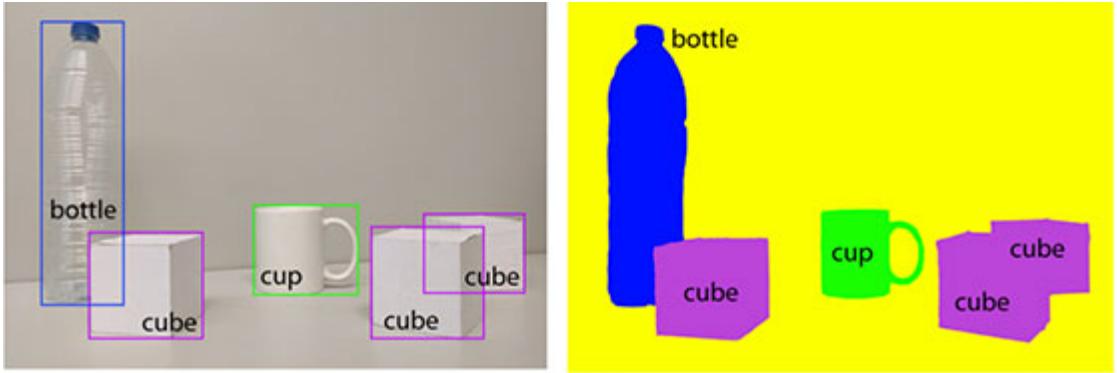


Figure 4.9: Two approaches to instance annotation in automatic object detection: coarse bounding boxes (left) and image segmentation at the pixel level (right) [GGOEO<sup>+</sup>18].

are usually defined in terms of color, shape, and texture. Image segmentation can therefore be defined as “a process of partitioning a digital raster image into a set of non-overlapping regions corresponding to the image scene composition” [32]. In Computer Vision, regions are defined in such a manner that there is greater variation between neighboring regions than within individual ones [Tow04]. As previously noted in Section 2.6.4 in the context of Communicative Graphics, automatic image segmentation is a compelling pre-processing step in the annotation process of semantically-enhanced graphics, since it saves authors substantial time by suggesting automatically defined regions that may be linked to semantic formal resources. In the context of modern CV, the kind of image segmentation in which each pixel in an image is linked to an object class label (e.g. ‘sky’, ‘woman’, or ‘table’) is known as *semantic segmentation*, which we will examine next. An example of semantic segmentation can be seen on the right image of Figure 4.9.

As already introduced in Section 2.1.6 and Section 2.5.3, currently available mechanisms for automatic semantic segmentation of real-world images have reached impressive results that have revolutionized many scientific fields by enabling intelligent systems to reason and infer knowledge about the world they operate in. Most of these approaches aim to solve the so-called *object recognition* problem and the related, higher-level, *scene understanding* problem. Object recognition, considered the “cornerstone of image understanding and computer vision” [LOW<sup>+</sup>19] tries to determine whether there are any instances of objects from given categories in an image (a real-world photograph or realistic drawing in the vast majority of cases), and, if present, to return the spatial location and extent of each object instance [LOW<sup>+</sup>19]. Solving the object recognition problem is of vital importance as a previous step to resolving more complex, conceptually high-level problems such as scene understanding and Visual Question Answering (VQA). Object detection can be divided into two main categories depending on whether the interest lies in detecting broad object categories or specific object instances. The latter is an easier problem in nature as it is basically a matching problem, whereas solving the former problem requires the ability to infer higher-level descriptions of the inner characteristics shared among all objects in the same category.

The spatial location of each detected object in an image is most commonly coarsely defined by means of a bounding box. A *bounding box* is an axis-aligned rectangle that tightly bounds the object. Even if bounding boxes are the most commonly used approach because of their simplicity, the research community is currently moving towards using image segmentation at the pixel level, where each pixel of the image (including those in the background), is assigned a semantic tag (i.e. *semantic segmentation*) [LOW<sup>+</sup>19]. Figure 4.9 shows an example image whose object instances are located using these mechanisms. A middle ground between these two approaches is to define annotations of polygonal shape allowing objects of arbitrary shape to be given a semantic tag with greater spatial precision than by employing a simple bounding box, but without the constrain of having every pixel in the image tagged.

Semantic segmentation can be seen as a natural step in the progression from coarse to fine inference [GGOEO<sup>+</sup>18]. The origin of this progression could be located at the image classification problem, in which a prediction for the whole input (i.e., the image itself) has to be made. For example, an image classification problem could consist on telling photographs from diagrams apart in an image database. This is followed by the problem of localization detection, which involves providing not only the semantic labels but also the approximate locations of the objects present in the image e.g. through determining a bounding box or centroid for each object. However, in order to achieve fine-grained inference, labels for each pixel in the image ought to be inferred [GGOEO<sup>+</sup>18]. Further improvements to pixel segmentation include *instance segmentation*, where each instance belonging to the same class gets a distinct label, and *part-based segmentation*, where constituent elements of each instance get labelled as well.

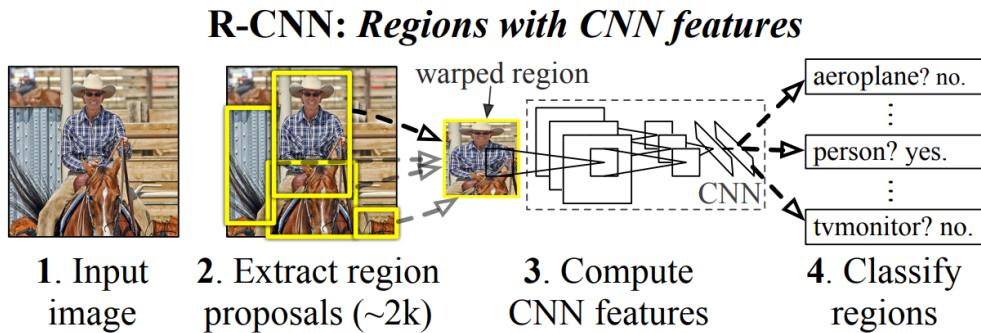


Figure 4.10: Architecture of the R-CNN model; an approach for automatic object detection in images via deep Convolutional Neural Networks (CNNs) [GDDM14].

A technological breakthrough that made reaching general semantic segmentation a more pragmatic enterprise happened in 2012 with the advent of deep learning (DL) methods such as deep CNNs applied to the object recognition problem which yielded a significant improvement in performance compared to the traditional CV methods employed before then [KSH12]. Since then, research on automatic image analysis has focused on DL methods, including research in the domains of automatic object detection, scene description, and VQA. Some

of these approaches based on deep learning (e.g. AlexNet, VGG-16, ResNet, or R-CNN; see Figure 4.10) have made such significant contributions to the field that they have become widely known standards and are commonly employed as building blocks in other segmentation architectures [GGOEO<sup>+</sup>18]. DL methods, as opposed to the descriptive, traditional CV methods, are of a predictive nature i.e., they discover the rules underlying a phenomenon and form a predictive model which minimizes the error between the actual and the predicted outcome considering all possible interfering factors [OCC<sup>+</sup>19]. Regarding semantic annotation, DL methods automatically infer graphical patterns at different conceptual levels by training a large neural network (made up of stacked layers of interconnected neurons that execute a given function to their weighted inputs) on a great number of labelled images, known as *training set*. Therefore, the main advantage of deep learning techniques over traditional computer vision methods is their ability to learn appropriate feature representations of the problem in an end-to-end fashion without the need of using hand-crafted features that require domain expertise, effort, and fine-tuning that greatly hinder their applicability in many common scenarios [GGOEO<sup>+</sup>18].

As opposed to object detection and semantic segmentation in photographs and video content, methods for automatic semantic tagging of diagrams are much harder to come up in the literature. Nevertheless, a few automatic data extraction methods have been explored e.g. [CJP<sup>+</sup>19] and [SKC<sup>+</sup>11], most of which are based on traditional edge detection or vectorization algorithms, which implies that the quality of their output relies greatly on the homogeneity and lack of noise of the input images [JKS<sup>+</sup>17]. Moreover, these methods try to infer the underlying data represented by a statistical chart instead of semantically tagging their constituent graphic objects. Therefore, the high-level conceptual knowledge encoded within the chart may become lost in the transformation process. As discussed in Section 3.1.2, the high-level abstractions that can be inferred from a diagram's constituent graphic objects and their relationships may carry important information for blind users of interactive systems enabling complex interaction with visually displayed information, such as NLIs to diagrams.

It is thus apparent that state-of-the art methods for semantic segmentation present a crucial drawback for the ultimate goal of scene understanding and visual question answering applied to diagrams. Namely, even if they are able to detect discrete objects in photographs with a very satisfactory accuracy, they are not able to explain their interactions or the relationships between them [KZG<sup>+</sup>17]. This hindrance is especially crucial if we aim to design systems that are able to automatically seize high-level inferences from knowledge representation graphics, since the exact nature of the relationships between their constituent graphic objects needs to be recognized. For example, translating a datapoint in a scatterplot by just a few pixels in any direction would involve an immediate modification of its corresponding value. On the other hand, the exact position of a person in a photograph, for example, is not as significant, and small variations of its visual attributes would likely not lead to a different interpretation of the photograph. As pointed out by some authors “the more formal and well-defined the semantics of the annotation vocabulary, the more opportunities for achieving interoperable and machine understandable annotations” [DGL<sup>+</sup>11]. Moreover, DCNN-based approaches to object recognition show a strong bias towards recognizing textures rather

than shape [GRM<sup>+</sup>19]. Given that texture is not a very commonly employed attribute in attribute-based graphic relations in diagrams (see Section 3.6.9), it is not certain whether the generalization capabilities of DCNNs applied to photographic content would translate well to knowledge representation graphics. Therefore, visual perception needs to be complemented with capabilities to describe the detected objects as well as how they interact in order to obtain a cognitive understanding of the world.

Ontologies are a suitable means of embedding such high-level understanding, as already discussed in Section 3.3. However, employing ontologies or some other kind of formal knowledge graphs to embed *a priori* knowledge to the task of automatic object detection has received limited attention in the scientific literature, even though, for example, it has been shown to improve recall while keeping the same level of mean average precision in the object detection problem applied to common photography datasets [FKL<sup>+</sup>17]. To best of our knowledge, no currently available method for the automatic embedding of high-level conceptual knowledge in image content by leveraging ontologies or any other description logic exists. Therefore, at the moment methods for authoring semantically-enhanced graphics need to rely on automatic or semi-automatic mechanisms. They are discussed next.

#### 4.4.4 Mixed-Initiative Semantic Annotation

This section discusses semi-automatic methods for authoring semantically-enhanced graphics, semantic segmentation of images, and other techniques in which a human annotator, in cooperation with automatic methods or other persons, takes part in the process of enabling access to visual content for Visually Impaired Persons (VIPs). The automatic object detection and semantic segmentation mechanisms discussed in Section 4.4.3 are able to outperform the capabilities of the human visual system only in strictly controlled, noise-free environments and are prone to making errors even with the use of large corpora of training data [AT13]. Therefore, in order to improve the results of fully automatic mechanisms for diagram object detection, mixed-initiative approaches involve end users in the annotation process so that they may correct faulty or missing pieces of the automatically inferred information.

Regarding diagrammatic representations, *ChartSense* [JKS<sup>+</sup>17] employs a twofold pipeline to extract data from statistical charts. First, the chart type is identified through deep learning techniques. Then, an interactive data extraction algorithm most appropriate for the identified chart type is applied. Users may interact with the system in order to fine-tune the result of the extraction algorithm in order to improve its accuracy. Other approaches to semi-automatic semantic tagging of diagrams first extract the text from the graphic and then try to automatically assign each individual text to one of a set of pre-defined tags. These tags can in turn be associated to individuals of a domain ontology such as countries and regions from the ISO 3166 Code List of countries. In addition, tags could be manually added or edited when necessary similarly to existing mixed-initiative approaches. In this manner, titles and text boxes appearing in time charts were automatically assigned to the geographical features they represented [DFVR10].

Steering away from methods relying on computer vision but that do not require a single

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

---



Figure 4.11: A screenshot of the *LabelMe* crowd-sourced image annotation Web tool in use [RTMF08].

person to perform the whole semantic labeling of the image we will now briefly discuss existing crowd-sourced methods for semantic annotation of images. One of the most influential approaches to crowd-sourced semantic labelling of real-world images was the *LabelMe* project of Russell et al., in which a web-based tool was designed that allowed human users to recognize and label object classes in photographs [RTMF08]. Users were randomly shown a photograph from a large image database covering a wide range of environments and situations depicted. They were then asked to label new objects appearing in the image by clicking control points along its boundary, as shown in Figure 4.11. Upon completion of drawing the object's boundary, users were required to input its name on a text field. Each user could label as many objects from as many images as they desired by using the web annotator tool. These labels were immediately visible and available for download for any users seeing the same image from the moment they had been incorporated by another user.

Crowd-sourced approaches to non-visual description of photographs can also be found in the accessibility literature. The most popular one is arguably *VizWiz*, a “talking application for mobile phones that offers a new alternative to answering visual questions in nearly real-time [by] asking multiple people on the Web” [BJJ<sup>+</sup>10]. The premise of the *VizWiz* method is simple: blind users of mobile phones may automatically recruit sighted participants through an app in order to obtain help during daily life activities such as shopping. A blind user takes a picture of an object or scene he or she needs assistance with and attaches a question about

the picture through the accessible VizWiz app; the picture-question pair is then sent over to human workers on Amazon Mechanical Turk (the most popular online crowd-sourcing marketplace) which respond with multiple spoken answers to the user's question. Users had to wait an average of 2 minutes per question before they got an answer, with an average cost for the user of \$0.07 USD per question. An update of the project, named *VizWiz::LocateIt*, combines the VizWiz approach with automatic computer vision techniques in order to help blind users locate arbitrary items in their environment [BJM<sup>+</sup>10]. Firstly, it added a web interface enabling human workers to outline objects in photographs belonging to picture-question pairs (similar to the LabelMe annotator, see above). Moreover, the mobile app itself was also improved to assist users in visualizing their environment and locate objects therein after having taken a general picture of their environment. The improved working procedure is divided into three main stages, adhering to the well-known Visual Information-Seeking mantra:

- *Zoom* stage: a sensor module estimates how much the user has to turn his or her mobile phone in the direction of the target object to be described. Audio cues are given to the user so they can move their mobile phone accordingly.
- *Filter* stage: CV techniques are employed to determine how close the current camera view is to the object outlined by an external worker in the web interface. Audible feedback is given next to the user so he or she may proceed closer to the object in order to snap a more detailed picture.
- *Details-on-demand* stage: remote workers are given a clearer picture of the object of interest along with the original question. An answer may then be given to the user following the original VizWiz approach.

Crowd-sourced approaches like those discussed in this chapter are nevertheless not a suitable long-term solution to performing complex semantic annotations of diagrams. The main reason for this is that they only support the creation of general annotations at the image level, as performing a more fine-grained annotation of the image's constituent graphic elements and their relationships would consume an impractical amount of time and would moreover require that the external human annotators were trained on the specifics of the conceptual model underpinning the image semantics. We may therefore conclude that the optimal approach to the annotation of semantically-enhanced diagrams would consist of a semi-automatic tagging mechanism enabling human experts to adjust the annotations output by an automatic tagger trained on a large corpora of semantically-enhanced diagrams. On the other hand, given that no such collection of training samples exists, at the current moment we find ourselves bound to performing a fully manual annotation of semantically-enhanced diagrams. The following sections discuss existing and newly developed tools for executing this task.

#### 4.4.5 Manual Semantic Annotation

As introduced at the beginning of this chapter, ontologies allow for embedding knowledge at different conceptual levels into graphically displayed information. However, as described in

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

---

Section 4.4.3 and Section 4.4.4, no current fully automatic, mixed-initiative, or crowd-based approach to semantic tagging of images supports the inclusion of high-level conceptual annotations underpinning the definitions and relationships between graphic object instances appearing on an image. Therefore, authors of semantically-enhanced graphics need to be able to seamlessly integrate formal knowledge representation with image content by means of some consolidated manual authoring mechanism. This need gets further strengthened by the deployment of semantically annotated images in training and evaluation tasks related to the automatic extraction of image semantics [DGL<sup>+11</sup>]. The necessity for such authoring tools has been noted by researchers in the Computer Vision (CV) and knowledge representation fields and hence a number of semantic image annotation tools are currently available. This section will outline the most relevant ones we have come across in the literature. For a more extensive comparison and analysis of these semantic annotators, the reader is directed to [DGL<sup>+11</sup>].

The K-Space Annotation Tool (KAT) is a “framework for semi-automatic, semantic annotation of multimedia content”<sup>12</sup>. It supports descriptive and structural annotations of images (c.f. Section 4.3.1) by manually localizing image regions using either a rectangle or polygon drawing tool which may be then linked to vocabulary elements from a user loaded ontology. Semantic annotations are based on the Core Ontology of Multimedia (COMM), which models their annotation levels and their linking while providing MPEG-7 based structural and media descriptions of formal semantics [DGL<sup>+11</sup>]. Input ontology languages include RDFS and OWL, with the end annotations being exported to OWL format. Likewise, PhotoStuff [HWGS<sup>+06</sup>] is another image semantic annotator offering very similar features to those of KAT, allowing for the addition of descriptive and structural metadata to photographs. Users can load local domain ontologies to annotate image regions, whereas internally two system ontologies are employed for automatic embedding of structural metadata following the MPEG-7 specification [DGL<sup>+11</sup>]. Content instances i.e. image regions are associated with corresponding domain resources by means of the foaf:depicts and foaf:depiction object properties of the FOAF vocabulary. PhotoStuff supports OWL and RDFS ontologies and outputs RDF semantics. Unfortunately, both KAT and PhotoStuff seem to have been abandoned, and we have not been able to acquire a running copy of the programs.

Other similar image annotator tools include AktiveMedia [Cha10], currently deprecated, and the M-OntoMat-Annotizer [STS<sup>+06</sup>]. An interesting feature of M-OntoMat-Annotizer is its ability to automatically extract low-level descriptors of image regions, such as dominant color, color structure, edge histogram, and contour shape, that can be associated to resources in a supporting ontology, see Figure 4.12. Moreover, it offers an automatic segmentation functionality as well as a magic wand tool that automatically identify salient image regions that may be semantically annotated. In addition, regions can be combined through the use of the region merging functionality, which greatly speeds up the annotation of complex bitmaps. Lastly, regions can be manually specified by means of the rectangle, ellipse, and freehand tools. In order to connect image regions (or the image as whole) with formal resources, the Visual Annotation Ontology and the Visual Descriptor Ontology are employed in a transparent

---

<sup>12</sup><https://launchpad.net/kat>

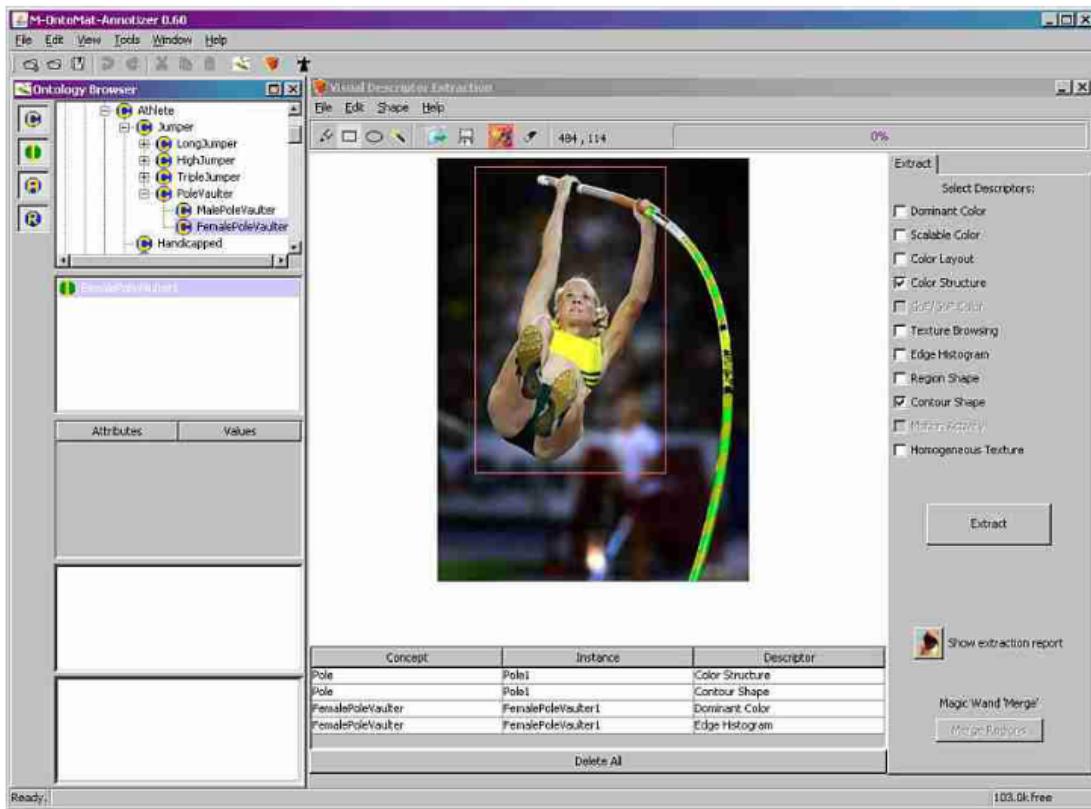


Figure 4.12: A screenshot of M-OntoMat-Annotizer’s workspace, showing its main working area, ontology browser, and visual description extractor [DGL<sup>+</sup>11].

manner to the user. M-OntoMat-Annotizer accepts RDFS and DAML as input languages, generating output annotations in RDFS.

The rest of manual image semantic annotation tools we have come across in the literature are either limited to annotations at the image level e.g. Caliph<sup>13</sup>, or are application-specific, most of which have been adopted in the medical fields. Examples of application-specific semantic annotators include Ratsnake [IGSM14], used in image-based medical computer-aided diagnosis (CAD), and iPad (image Physician Annotation Device) [RRSB08], which aids clinicians in the annotation process of radiological images. These annotator tools generally make heavy use of domain-specific ontologies and taxonomies to support the annotation process. Since these tools offer limited functionality in the context of this thesis, where we aim to semantically annotate general purpose diagrams, we will not delve into further detail. Video annotation tools will also be jumped over since they are not directly applicable to the annotation of static vector graphics. A discussion of available semantic video tagging tools may nevertheless be consulted in [DGL<sup>+</sup>11].

<sup>13</sup><http://www.semanticmetadata.net/features/>

## 4.5 Semantic Annotator for Inkscape

This section describes Semantic Annotator for Inkscape (SAI), an authoring tool for semantically-enhanced vector graphics developed in the context of this thesis<sup>14</sup>. First, the motivation for developing SAI within the currently available alternatives for image semantic tagging will be discussed and its requirements laid out. Next, its main features will be outlined and a typical authoring workflow detailed. Finally, some potential improvements and additional desired functionality to be included in future development iterations will be discussed.

### 4.5.1 Motivation and Requirements

The manual image semantic annotator tools analyzed during problem investigation (Section 4.4.5) suffer from a number of drawbacks that prevented their use for the authoring of semantically-enhanced graphics to be investigated in this thesis. Namely, some of these shortcomings include:

- Exclusive support for images in raster format. As mentioned in Section 4.3, some features of the SVG vector image format are very beneficial in the context of creating non-visual accessible diagrams. Therefore, having the possibility to directly add semantic annotations to individual and grouped SVG elements is of the utmost importance. Moreover, raster images may be semantically enhanced by means of including corresponding transparent vector elements as introduced in Section 4.3.3. On the other hand, the reverse approach i.e., to semantically annotate vector graphics through tools meant for bitmap graphics authoring, is not as straightforward, requiring of complex prepossessing mechanisms such as automatic image segmentation. Therefore, the first requirement of SAI is that it must have the ability of associating resources from an imported ontology to individual SVG elements or groups of elements.
- Limited drawing/tracing toolbox. Given that they image semantic annotators we listed in Section 4.4.5 have been designed from scratch with the sole purpose of adding semantic tags to image regions, they have not placed a strong focus on their drawing, tracing, and other graphic authoring tools. These tools generally offer at most two or three tracing tools; namely, rectangle, ellipse and hand free shapes can be created in most of them. However, it would be very convenient for authors of semantically-enhanced diagrams to be able to draw complex shapes in order to improve the efficiency of the authoring process of intricate diagrams. Accordingly, the second functional requirement of SAI is to provide full-fledged graphic authoring and semantic annotation tools in an integrated workspace.
- Short lifetimes. This point is related to previous one, as the specificity of these tools entails as well that not many user groups benefit from them. As such, their

---

<sup>14</sup>Substantial parts of this chapter were published in: Tomás Murillo-Morales, Jaromír Plhák, and Klaus Miesenberger. Authoring semantic annotations for non-visual access to graphics. *The Journal on Technology and Persons with Disabilities*, 6:398–413, 2018

developer support and general lifetime are limited to the duration of the project they were developed in. Most of the tools listed in the previous section had been already deprecated by the time of writing. The third functional requirement of SAI is therefore that it is integrated in a general purpose, widely-used graphics editor.

In order to overcome these hindrances, Semantic Annotator for Inkscape (SAI), an authoring tool for semantically-enhanced graphics, was developed<sup>15</sup>. Moreover, an additional functional requirement is that it must support the automatic inclusion of structural annotations (c.f. Section 4.3.1) from UVO (c.f. Section 3.6) to the final semantically-enhanced diagram, such as property occurrences underpinning the *x* and *y* coordinates of graphic objects in the SVG canvas or `uvo:hasSVGElement` property occurrences. This aspect is necessary in order to save authors substantial time in the authoring process. Freeing users from having to manually add a vast number of supporting structural properties to the created knowledge base makes the annotation process more efficient and user-friendly, thereby stimulating the creation of semantically-enhanced graphics.

Having described the motivation for the design of an annotator of semantically-enhanced graphics, the following section gives an overview of the main features of Semantic Annotator for Inkscape (SAI), the software artifact designed in order to meet the prescribed requirements.

#### 4.5.2 System Overview

The Semantic Annotator for Inkscape (SAI) prototype has been implemented as an extension of Inkscape, written in Python<sup>16</sup>. As described in Section 4.4.1, Inkscape is one of the most popular graphics editors currently available, working primarily with SVG files. Writing SAI as an Inkscape extension allows us to overcome the drawbacks, outlined in the previous section, that current image semantic annotators have. Namely, because Inkscape is a widely used vector graphics authoring tool with a rich and active developer and user community, the risk of it becoming deprecated in the long term greatly lessens. Moreover, Inkscape has a full-fledged set of tools for authoring vector graphics, which allows authors of semantically-enhanced graphics to create new vector graphics of arbitrary complexity from scratch in the same workspace where semantics are associated to their elements.

In a typical usage workflow, the author may import an already existing SVG file, trace a raster image into its vector counterpart, or create a new SVG image from the ground up using the available graphic authoring tools of Inkscape. At any time during the authoring process, the user can choose to semantically annotate parts of the image by running the SAI extension from the Extensions menu (as shown in Figure 4.14) or via a keyboard shortcut. This launches the SAI window (shown in Figure 4.13). Therefore, complex graphics can be imported or created by the user in the same place where their semantic markup is added with no further software being required, saving the user in both authoring time and complexity.

---

<sup>15</sup>SAI is free software. It is available for download at <https://github.com/tomurillo/SAI/>

<sup>16</sup><https://www.python.org/>

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

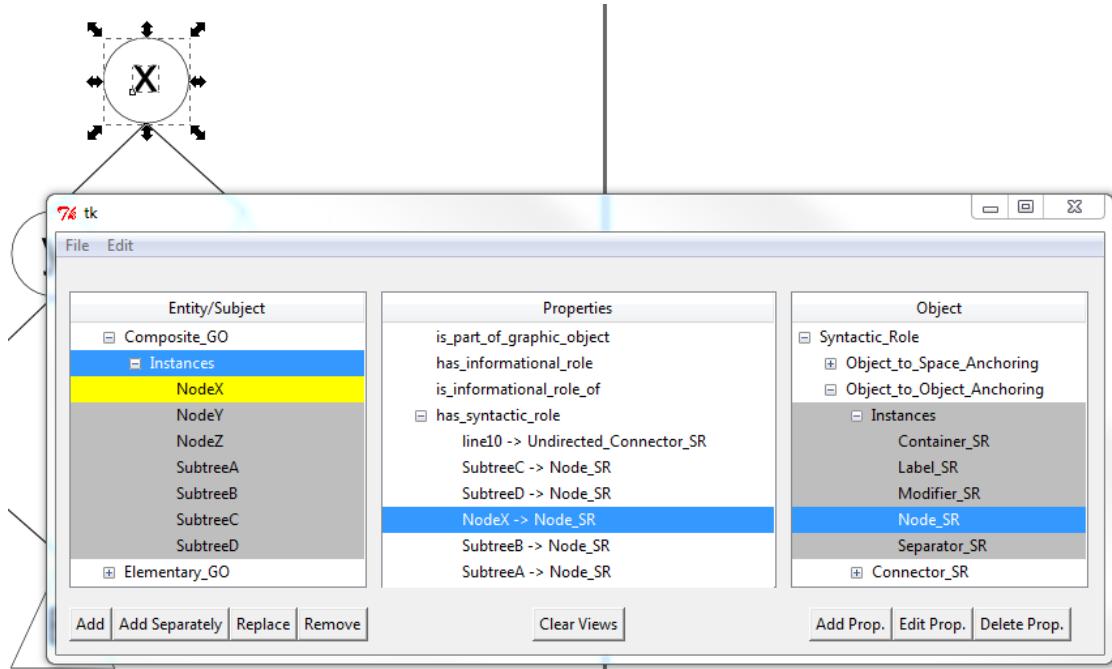


Figure 4.13: Semantic Annotator for Inkscape’s GUI (foreground) in the process of annotating a red-black tree drawn in Inkscape (background).

Before any annotation of the graphic can take place, the user has to load an ontology in RDF format (e.g. UVO, outlined in Section 3.6) by choosing the relevant option under the ‘File’ menu. The extension tries to automatically infer the namespace used by the visualization ontology that semantically underpins the image’s graphic objects and their relationships (i.e. the namespace currently used by the loaded UVO ontology). If the namespace is not found, it prompts the user to manually input it via a text box, as depicted in Figure 4.15. SAI also checks whether the structural datatype properties necessary to link ontological entity instances to SVG elements (i.e. to semantically annotate the graphic) are present in the ontology. As described in Section 4.3.1, formal resources are associated with its constituent SVG elements by means of datatype properties, which are added or edited automatically by SAI in a transparent manner to the user. These properties link each SVG element to the instances they belong to as well as indicate the coordinates in which they are placed on the graphic canvas when it is saved. This way of representing structural and localization information follows the approach of most semantic image annotators, which generally hide the lowest structural descriptions from the user, implicitly adding them in a transparent manner [DGL<sup>+</sup>11]. In SAI, the datatype properties employed for this purpose are `uvo:hasSVGElement`, and, for graphic object spatial localization, `uvo:hasXCoordinate` and `uvo:hasYCoordinate`.

Any number of SVG elements present in the image being edited in Inkscape (paths, shapes, text) may be selected by the author before running the SAI extension in order to relate them to certain ontological elements. A hierarchical list of entities and instances from the ontology

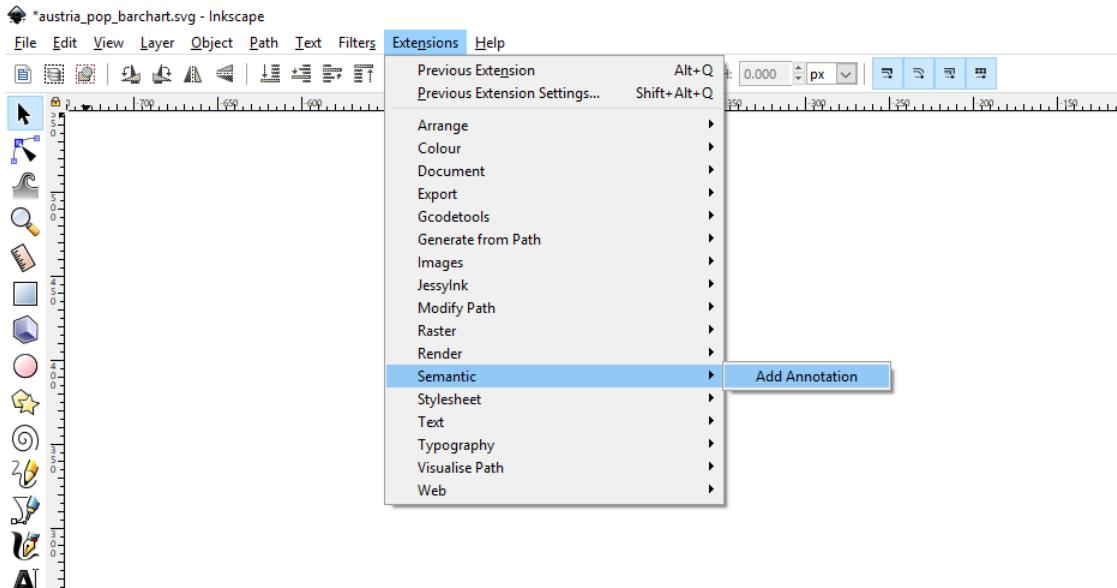


Figure 4.14: SAI is integrated in Inkscape, where it can be launched from the ‘Extensions’ menu item.

appear on the left-hand tree view of its Graphical User Interface (GUI). Instances can be distinguished from entities by their grey background color (c.f. Figure 4.13). At the same time, those instances some of the currently selected SVG elements belong to are highlighted in yellow in the ‘Entity/Subject’ tree view. The author can select entities and instances and then press one of the available buttons underneath to perform one of the following functions:

- **Add:** if an entity has been selected in the ‘Entities’ tree view, a new named instance of said entity associated with the chosen SVG elements will be created. The user will be prompted to add a meaningful name to this annotation. If an instance is chosen instead, the selected SVG elements will be added to the constituent elements that the instance is made up of already.
- **Add Separately:** this functionality is similar to that of the ‘Add’ button when used to create new instances of an entity. However, a new instance will be created for each of the selected SVG elements, instead of grouping them all together under the same instance. An automatic naming mechanism has been implemented in order to make the annotation faster. It takes into account the type of SVG element selected in order to give a meaningful and unique name to it (see Figure 4.16).
- **Replace:** substitutes all SVG elements present in the chosen instance for the currently selected SVG elements in Inkscape’s canvas.

#### 4. SEMANTIC ANNOTATION OF DIAGRAMS

---

- **Remove:** when one or more SVG elements have been selected in Inkscape, it will remove those elements from the chosen instance if possible. If no SVG elements have been selected, it allows the user to remove whole instances from the ontology.

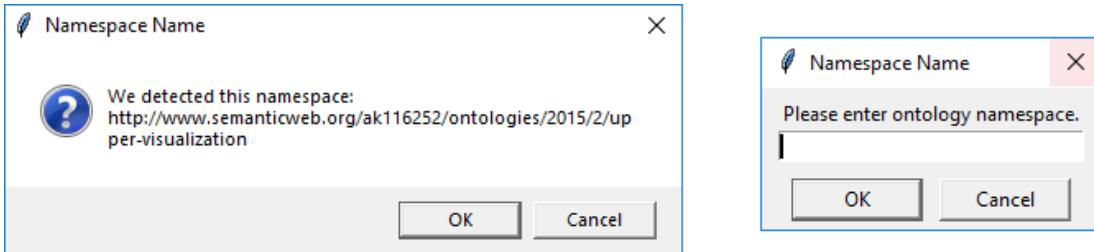


Figure 4.15: Automatically detected visualization namespace (left). SAI users may also choose to input their desired namespace (right).

Once a semantic annotation has been created by defining which SVG elements belong to an entity instance of the ontology, the user may define property occurrences in which the annotation (instance) is involved. The system supports both object and datatype properties. Properties and their occurrences within the ontology are also hierarchically displayed in the middle tree view of the GUI ('Properties' tree view). When a user double clicks on a property, the 'Entity/Subject' (leftmost) tree view gets updated showing only those entities and instances that belong to the domain of the property, if defined. If the chosen property is an object property, the 'Entity/Object' (rightmost) tree view will then display those entities and instances that belong to its range (i.e. the set of possible objects of the property triple). In case the user double clicks on a property occurrence, the leftmost tree view will highlight the triple subject while the rightmost one will do the same for its object. This assists the user when choosing the elements of a triple to be added, edited or removed.

After selecting a property or property occurrence in the 'Properties' tree view, three operations can be carried out depending on which button below the 'Object' tree view is clicked, namely:

- **Add Property:** this function lets the user add a new object property triple by selecting a subject (entity instance), property, and object (entity instance). In case a datatype property is chosen, the rightmost tree view will be empty and the user will be prompted to add a value (e.g. a string or a number) to the triple instead. If the triple already exists, the user is informed and no triple is added.
- **Edit Property:** lets the user select a property occurrence and modify its subject, object (value if the chosen property is a datatype property) or both by choosing a different instance in the 'Subject' and/or 'Object' tree views.
- **Delete Property:** deletes the chosen property occurrence, regardless of the subject and object chosen by the user.

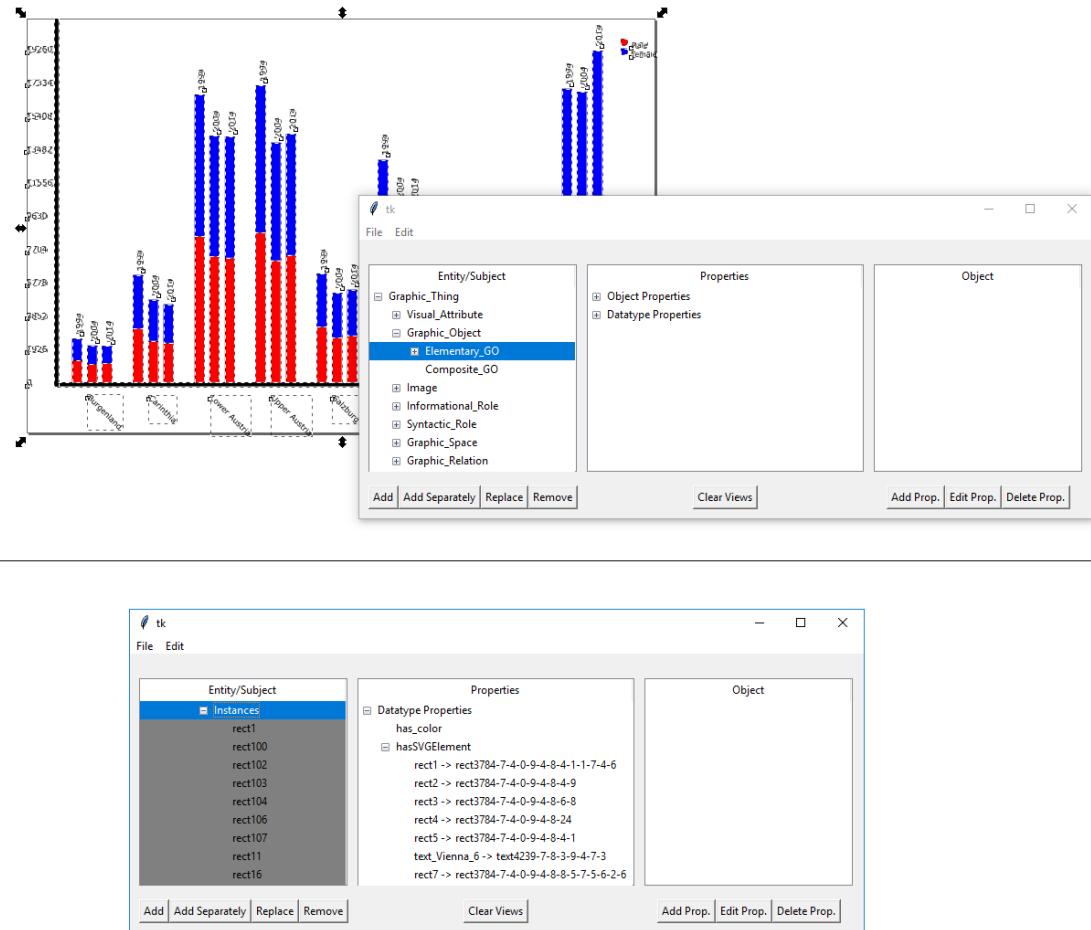


Figure 4.16: Automatic generation of graphic object instances in SAI. The author first selects all relevant SVG elements (top; background). She then chooses the right class for the elements (top; foreground). Lastly, she clicks on the ‘Add Separately’ button. As shown in the bottom panel, an instance for each element has been generated. Moreover, structural descriptions as prescribed by UV0 are added to the knowledge graph in the form of datatype property occurrences.

Once the author has finished editing a semantically-enhanced graphic, changes can be saved into the original ontology or a different one by choosing the appropriate option under the ‘File’ menu item. All new instances and properties will be added into the selected ontology, and any edited/removed ones will be modified as well. At the same time, the current position of each element is added to the ontology via structural properties. The position is computed as the average of the central points of an instance’s constituent SVG elements according to their type, and is added automatically as UV0 datatype properties linked to the instance in the ontology. When possible, additional visual attributes (as prescribed by corresponding UV0

properties) of the saved instances, such as width and height, are automatically computed and included to the saved ontology. Additionally, authors may use the ‘Undo’ functionality under the ‘Edit’ menu item. It allows previously performed actions to be reversed, which is of great assistance when authors make an error in the annotation that would be costly to manually reverse, such as when adding many instances of the wrong class by mistake to a semantically-enhanced graphic.

##### 4.5.3 Validation and Future Work

Heuristic evaluation of SAI with three authors of semantically-enhanced SVG images both at the University of Linz and the Masaryk University Brno has shown a significant increase annotation efficiency, especially in the case of complex graphics. For example, annotation of a diagram with over 500 graphic object instances could be performed in slightly over half an hour. The same annotation performed used the traditional, disjoint approach of creating the vector graphic in Inkscape first, followed by creating a coupled ontology in Protégé inheriting the resources of UVO and VDO, would have taken an impractical amount of time (i.e. several hours).

In the following, benefits of using SAI over traditional semantic annotation workflows that have been noticed from its heuristic evaluation are described:

- Whereas using traditional tools such as Protégé (introduced in Section 4.4.2) requires naming each one of the entities in the ontology individually, SAI’s automatic naming mechanism of ontological elements underpinning elementary graphic shapes along with the ‘Add separately’ feature implemented in SAI saves the user substantial time in the naming process. An example of this mechanism is shown in Figure 4.16, where all elementary graphic object instances of a bar chart are added simultaneously with very little effort from the user.
- Highlight of selected SVG elements in the user interface notably speeds up the authoring of property assertions by selecting their constituent elements in the graphic itself, instead of having to look them up by name (which might be non-descriptive). For example, a user might choose to add a syntactic role to all bars in a bar chart by selecting them in Inkscape and then launching SAI, where all entities they belong to will be highlighted and may be chosen to create new property triples.
- SAI lets the user choose several objects or subjects for the same property to create several assertions of it simultaneously, which noticeably quickens the definition of assertions of properties which are commonly used on a given domain.
- SAI saves the author substantial time when adding several instances of the same entity to the supporting ontology. It remembers the last class chosen when adding a new individual, so its parent Entity is automatically selected in the GUI when relaunching the extension. This way, the author simply needs to click on the ‘Add’ button after selecting the SVG elements that will make up the new individual. This is especially

useful in knowledge representation graphics, as the same graphic relations may apply to many different elements e.g. labelling relations on a map with one label per geographical landmark.

- For every instance in the ontology, their approximate coordinates within the SVG canvas are computed by taking into account the coordinates of its constituents SVG elements and included in the ontology as datatype properties. Authors can make use of the `transform` SVG attribute (automatically applied by Inkscape) to scale, rotate or translate elements, and the resulting end coordinates on the SVG canvas are calculated by using transformation as specified in [Wor11a]. In addition, other useful structural property occurrences are added to the knowledge base in a transparent manner to the user.

SAI is currently in working status and has been used for the creation of the semantically-enhanced diagrams that were employed during the design and validation phases of the main design cycle described in Chapter 5. Nevertheless, the current version is still an early version that lacks many useful features that would be of great use in order to improve the efficiency of the authoring process. In the following we will list some of these desirable features. Being a free, open source project, developers are encouraged to contribute to SAI in order to make these and other features a reality in the near future.

Currently, the input ontology format supported by SAI is constrained to RDF. This restriction is however not very crucial, as the abstract syntax of OWL DL and OWL Lite (c.f. Section 3.2.4) can easily be mapped to the RDF/XML exchange syntax that SAI understands. A detailed description on how such mapping is to be performed can be found in [Wor04a]. In any case, most available Semantic Web editors, such as Protégé (Section 4.4.2), are able to serialize OWL ontologies to RDF/XML format.

In addition, Section 4.4.4 concluded that, currently, the optimal workflow for the semantic annotation of graphics ought to be supported by automatic image segmentation and object detection methods that assist the human annotator. The inclusion of such mechanisms in SAI would contribute to an even greater efficiency of the annotation process in raster images. On the other hand, the annotation process of graphics in SVG format would not benefit from these methods, since their constituent primitive graphic elements can be individually selected, grouped, and assigned to formal resources by a human annotator. Lastly, the automatic inclusion of administrative annotations stemming from file metadata (c.f. Section 2.1.7) is another interesting feature that could be added to a semantic image annotator such as SAI. Some useful metadata (e.g. of the Exchangeable Image File Format (EXIF) image format) that may be automatically parsed and embedded into the semantically-enhanced graphic include capture and last edit dates, GPS location coordinates, author name, copyright details, compass heading, device information, capture information, and original filename. Most of this information is however not included in SVG diagrams, since they are not photographs which must be manually authored using specialized software (c.f. Section 4.4.1).

## 4.6 Summary

This chapter has tackled the problem of annotating semantically-enhanced graphics, with a focus on vector diagrams supported by formal semantics from the hierarchy of ontologies for visualization characterized in Chapter 3. This problem stems from the need of including formal semantics at different levels of abstraction to the raw visual primitives of diagrams in order to enable their effective operation by client applications that can perform complex inference and cognitive tasks on diagrammatically displayed data, such as Natural Language Interfaces (NLIs) employed in the context of accessibility (Chapter 5).

The concept of semantically-enhanced graphics was described in detail in Section 4.3. It consists in embedding a relevant knowledge base (an instantiated ontology) that characterizes the concepts depicted in a vector image within the image file itself. The mapping from visual primitives (and groups thereof) to ontology resources is performed via structural annotations i.e. datatype properties connecting SVG elements determined by their `id` attribute to formal resources. In such manner, the formal semantics of what is depicted by a diagram can be shipped with the diagram itself and consumed by end applications, thereby enabling complex reasoning and inference activities supported by visualization.

Next, the task of authoring semantically-enhanced graphics, known as *semantic image annotation*, was discussed in a problem investigation phase. Existing approaches were classified into four main categories, namely:

- *Disjoint Semantic Annotation* methods author the graphic content first using specialized vector or raster graphics software (Section 4.4.1), followed by an ontology authoring process using also specialized tools such as Protégé (Section 4.4.2). These methods are slow, cumbersome, and very error-prone, and therefore should be avoided and limit the take-up of semantically-enhanced graphics.
- *Automatic Semantic Annotation* methods (Section 4.4.3) employ state-of-the-art techniques in Computer Vision (CV) for automatically segmenting an input image into suitable regions, followed by assigning regions to object categories, or, much less commonly, to resources from a supporting conceptual base as well characterizing relationships between inferred objects. Despite their impressive advance in recent years in photographic domains, these methods are still very rudimentary and inaccurate when employed to annotate diagrams.
- *Mixed-initiative Semantic Annotation* approaches (Section 4.4.4) combine automatic and manual methods that complement each other, or leverage the collaborative effort of several users in order to perform the semantic annotation of an image more efficiently. These methods are nowadays constrained to simple region annotation of images; in addition, automatic segmentation methods are not practical for images in SVG format, since their constituent primitive graphic elements can be individually selected, grouped, and efficiently assigned to formal resources by a human annotator using manual semantic annotators.

- *Manual Semantic Annotation* (Section 4.4.5) software allow authors of semantically-enhanced graphics to map regions or elements from a graphic to resources of a conceptual model using a number of annotation tools. Currently available semantic annotators are however constrained to raster images, have limited drawing/tracing toolboxes, or have been abandoned.

The limitations of currently available manual image semantic annotators, discussed in Section 4.5.1, motivated the introduction of a new design problem. Namely, it involved the design of an effective manual semantic annotator of vector diagrams with automatic support for structural annotations stemming from the hierarchy of ontologies defined in Chapter 3. The resolution of this problem is described in Section 4.5; it consisted in the design and validation of Semantic Annotator for Inkscape (SAI).

SAI is an Inkscape (c.f. Section 4.4.1) extension that enables authors of semantically-enhanced graphics to create complex vector graphics and semantically augment them with resources from an imported ontology in a seamless manner using an integrated workspace. SAI includes advanced authoring features such as automatic instance naming, highlighting of ontology resources by selecting their constituent graphic elements in the SVG canvas, automatic inclusion of useful structural properties to the knowledge base, and more (Section 4.5.2). SAI greatly improves the efficiency and user-friendliness of creating semantically-enhanced diagrams, specially those underpinned by resources from the UVO, VDO, and DDO ontologies described in Chapter 3. The created semantically-enhanced diagrams may then be actioned by client applications such as Accessible Universal Diagrams through Dialogue (AUDiaL), an accessible Natural Language Interface (NLI) to diagrams that is described in next chapter.



# Natural Language Interface to Diagrams

This chapter sets out to describe in detail the treatment design stage taking place within the greater design cycle undertaken in this thesis (c.f. Figure 1.5 from first chapter). Its main outcome is the development of a Natural Language Interface (NLI) prototype allowing effective non-visual access to statistical charts. The pertinent information encoded in diagrams is to be accessed by end users by means of queries expressed in free-formed natural language questions and statements which are fed into an accessible Web interface. This query is automatically interpreted by a server application following a distributed client-server model and an answer (or lack thereof) then communicated to the user. As it will expanded upon in Chapter 6, this prototype is designed to enable its interaction with a problem context with the intended goal of treating the original problem of allowing blind persons to efficiently access statistical charts.

The design process of the NLI prototype, named *Accessible Universal Diagrams trough Dialogue (AUDiaL)*, first involved studying the existing groundwork in NLIs, and, more specifically, NLIs to structured knowledge bases, endeavors described in Section 5.1 and Section 5.2, respectively. It may be regarded as the final step in the problem investigation phase of the design cycle that precedes the treatment design phase. The beginning of this chapter thus expands upon the conceptual framework that we have been constructing in previous chapters with architectural definitions specific to the design of a NLI to conceptual models such as ontologies.

After the necessary conceptual framework has been sufficiently characterized, the treatment design phase of the NLI artifact is laid out, starting in Section 5.3. We will commence by specifying the requirements for the prototype and how they contribute to our problem goals. After having a clear understanding of the prototype's architecture and functionality, the rest of the chapter gives an in-depth specification of its final design. The outcome of this chapter

consists in a working, accessible, Web-based NLI to semantically-enhanced diagrams which are underpinned through the hierarchy of ontologies for visualization outlined in Chapter 3 (i.e. AUDiaL). AUDiaL allows users to perform complex analytical and wayfinding tasks which are usually assisted by external, diagrammatically displayed information.

The NLI prototype designed in this chapter will be employed for the next phase of the greater design cycle, known as the empirical cycle. The empirical cycle deals with producing answers to knowledge questions about artifacts (i.e., the designed NLI) in a context (i.e., blind persons accessing diagrams) [Wie14]. It therefore enables us to answer the principal research questions of this thesis. Its main result consists on the validation of the designed NLI, described in this chapter, in order to study whether natural language is a suitable means for Visually Impaired Persons (VIPs) to autonomously access diagrammatically displayed information, and how such a NLI ought to be accommodated for the specific needs of its intended user group.

## 5.1 Natural Language Interfaces

Before delving into the specifics of the design of the developed Natural Language Interface (NLI) prototype, in the following sections we give a concise characterization of current research and applications in NLIs in its relevant context as part of the broader required problem investigation preceding artifact design. First, an introduction to the fields of dialogue and NLIs within the general field of Human-Computer Interaction (HCI) is given. Next, a specific account of NLIs to conceptual models, and more specifically, to ontologies (c.f. Section 3.2), will be laid down. The ideas gathered from this analysis will serve as the foundations upon which the dialogue interface eventually developed to enable blind users interact with semantically-enhanced diagrams will lay.

### 5.1.1 Motivation

In its most basic definition, a *Natural Language Interface (NLI)* or *dialogue system* is a computer system that interacts with the user in multiple turns through utterances expressed in natural language via text or speech input and output. NLIs enable users to communicate with computer-related devices without the use of specific physical devices such as a mouse, keyboard, or buttons. By leveraging an innate human skill such as language, NLIs do not require of complex preferences and customization features thus allowing its users to forego most of the training time which would be otherwise required to operate a complex Graphical User Interface (GUI). As “time spent in learning and operating the personalization features is time mostly wasted from the task at hand” [Ras00], NLIs offer an efficient mechanism for operating software in a non-visual manner. In addition, NLIs, as opposed to traditional GUIs, do not employ modes (a mode in a GUI refers to a specific way in which the GUI reacts to the same sequence of user actions). A NLI, on the other hand, always react in the same manner when given the same user query. Research in HCI shows that modes in user interfaces are a significant source of “errors, confusion, unnecessary restrictions, and complexity” [Ras00], and thus a pure NLI may enable users to perform tasks more efficiently than by employing

alternative complex GUIs. Some authors go as far as stating that NLIs “can provide the most useful and efficient way for people to interact with computers” [OB97].

NLIs are among the technologies that best embody the recent paradigm shift in interface design from the traditional “computer as a tool” metaphor to the modern “computer as an agent” analogy. The latter allegory regards the human-computer interaction as a cooperative activity between two agents, namely, the user and the system. In this type of agreement, the system mediates between the user and the back-end knowledge, and assists users in achieving their goals [Jok09]. This paradigm is especially suited for engaging with complex domains that are difficult to model in the detailed and exhaustive way required by state-based technologies [Jok09]. A prime example of such a domain would be that of diagrammatic representations, as they may be employed to visually codify numerical information of arbitrary complexity from any aspect of human knowledge.

Semantically-enhanced diagrams (described in Section 4.3), having their relevant semantics encoded at a higher conceptual level than that of raw pixel or vector data, enable complex access means through the use of AT and other client applications. Among them, potential applications include didactic software employed in a classroom setting and accessible sonified, haptic, or hybrid alternative versions that can be understood by Visually Impaired Persons (VIPs). However, as it was discussed at length in Chapter 2, haptic and sonified diagrams present a great deal of drawbacks hindering their take-up by the visually impaired population. Chapter 2 also resolved some of the potential benefits of employing a NLI in the context of non-visually accessing visually displayed information. Here it suffices to say that, according to current research, “verbal communication must be employed as the primary means to present [...] visual information to people who are blind” [Kea14]. This statement is however constrained to static descriptions of visually displayed information. As already discussed, a much greater deal of interaction between the user and the information being consulted is necessary if its full, functionally equivalent, understanding is desired.

Therefore, enabling diagrams to be accessed through a NLI appears to be a fitting approach that may contribute to the goals of potential blind readers of diagrams. This leads us to the main empirical knowledge question of this work (c.f. Section 1.4). Namely, we are concerned with examining whether NLIs are a sufficient means for accessing statistical charts by blind computer users. The rest of the present chapter thus focuses on the following design goal: to design an artifact (a NLI prototype) such that low-level analytical tasks commonly carried out by sighted persons with external support from statistical charts can be performed non-visually by exclusive means of natural language. Nevertheless, this is not an idea that comes without limitations, as many of the cognitive advantages of visualization can only be partly compensated, if at all, when employing a NLI. The question of how to adapt a NLI for the specific needs of blind persons will be discussed in detail in Section 6.1 of the following chapter. Moreover, current limitations of NLIs to diagrams will be studied later on during the validation process of the NLI prototype described in Section 6.9.

### 5.1.2 Overview

This section outlines the fundamental conceptual constructs employed by the majority of Natural Language Interface (NLI) systems. The main components of the architecture of a typical NLI, as depicted in Figure 5.1, include the following entities and processes, grouped into five theoretical modules [Plh14]:

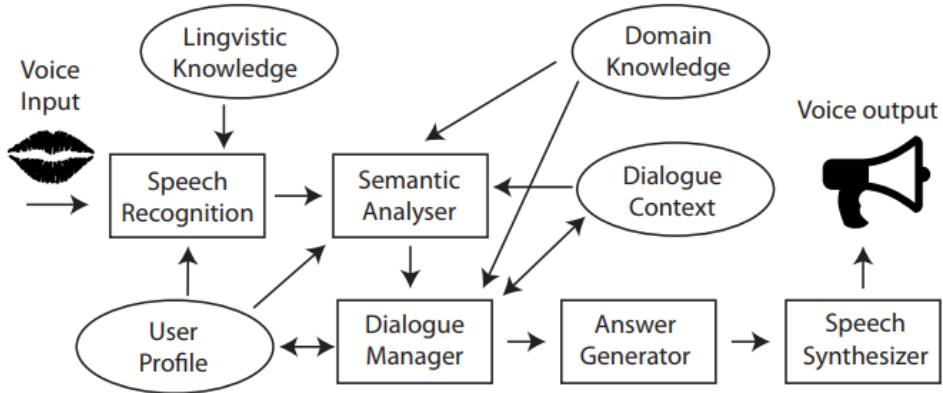


Figure 5.1: General architecture of a Natural Language Interface (NLI) [Plh14]. Note that not all the modules shown here may be present in a NLI.

1. The first module of the processing pipeline is the *speech recognition module*. It is in charge of recognizing the spoken utterances of the user and converting it into text. A collection of *linguistic knowledge* may be queried by the speech recognition module to interpret natural language in a given language such as English or German. Moreover, this information may be enhanced with *user profiling* capabilities in order to improve its accuracy for a given user. In dialogue systems where user input is given as free-formed text queries, such as Web-based NLIs, a speech recognition module is not present.
2. The *semantic analyzer* is the component that interprets the semantics of a given utterance. Often, semantic analysis may be preceded by a *syntactic analysis* of the query, in which corresponding syntax trees are computed by working out the grammatical structure of each sentence (for example, through a probabilistic natural language parser that computes the most likely analysis of a given sentence, see Section 5.6.1 for an example). The semantic analyzer then assigns higher conceptual meaning to each clause of the input text by associating them with instances from a domain *Knowledge Base (KB)* e.g. a domain ontology. Syntactic ambiguities (e.g. “we saw her duck”), anaphora resolution (“her dog”), and other surfacing dislocations may be automatically resolved from the current dialogue context and domain knowledge. Otherwise, the incomplete information is sent over to the dialogue manager for future resolution.

3. The *dialogue manager* is the stateful component managing all aspects of the dialogue with the user, such as the *dialogue strategy* to be followed and the response to a user's query. The dialogue manager enables the dialogue system to retrieve missing pieces of information from the user in order to carry on with its dialogue strategy (auto-correction). A dialogue manager should be capable of reacting to the following events [Plh14]:

- *No match*: the system is not able to understand the given utterance and the user needs to rephrase her query.
- *No input*: the user does not respond after a given time period or the system is provided an empty input.
- *Request for help*: the user asks for more detailed information about a certain aspect of the discussed subject.
- *Repetition*: the user asks the system to repeat the previous information.
- *Re-establishing context*: one or more pieces of information that had been previously established are presented to the user.
- *Starting over*: the user wants to restart the entire dialogue or a specific sub-dialogue.
- *Auto-correction*: validation of the user utterance based on internal rules, dialogue context, and domain knowledge.

4. The *answer generator* translates the information provided by the dialogue manager into a natural language utterance in the language of the user.
5. Lastly, a *speech synthesizer* converts the output utterances of the answer generator into audible speech that is heard by the user. For an overview on the functioning of speech synthesizing, see Section 2.1.3. A speech synthesizer may be left out in systems outputting the system utterances in Braille form (c.f. Section 2.1.2 for a discussion of Braille) or in text that is to be read by a sighted user.

The majority of Natural Language Interfaces focus on answering *factoid questions* i.e., questions that can be answered as simple facts expressed in short texts [JM18]. For example, factoid questions include questions such as "what is the average calorie content in a banana?", "what is the capital of Austria?", or "who is the founder of Amazon, Inc.?". Factoid-answering dialogue systems thus do not require of complex dialogue strategies with its users, laying their focus on correctly interpreting the user's query and retrieving the relevant pieces of information from their domain knowledge. Dialogue systems tackling factoid questions may follow one of two main paradigms or a combination thereof, based on the source of the domain knowledge they draw from. Namely, these paradigms are [JM18]:

- *IR-based question answering systems* first find and retrieve relevant text documents within an existing text corpora such as a collection of Web pages. Then, reading

comprehension algorithms and methods are employed to read the retrieved documents and draw an answer directly stemming from certain spans of text therein.

- *Knowledge-based question answering systems*, on the other hand, start by building a semantic representation of the query, by mapping the unstructured natural language queries to a logical representation. These computed meaning representations are then used to query knowledge bases such as relational databases of facts or domain ontologies. The NLI prototype designed in this chapter, AUDiaL, is an example of such system.
- Moreover, *hybrid systems* may perform a combination of these paradigms by using both text datasets and structured knowledge bases in order to answer the user's questions. Most large industrial systems, such as the *DeepQA* system in IBM's Watson, adhere to some sort of hybrid paradigm [JM18].

Another common and critical design choice in NLIs is whether to prompt users to select an utterance from fixed menus, known as *system initiative*, or to determine user interactions from unrestricted utterances, known as *mixed initiative* [SKLW00]. AUDiaL, the NLI prototype whose design will be characterized later in this chapter, has been designed as a mixed initiative NLI in order to enable users to operate it to access diagrams of disparate nature without the need for undertaking lengthy and cumbersome training beforehand.

As a final remark, it is a well-known fact that both practical and research prototypes of NLIs very often fail to reach the necessary level of user satisfaction that would allow users to enjoy the interaction [Jok09]. Some of the most commonly cited reasons behind user dissatisfaction with NLIs include bad speech recognizer performance, a problem that has been solved for the most part in recent years with the rise of more powerful deep learning models for speech recognition, and the straightforward nature of end dialogues that have been designed with the unique goal of solving a specific task. In addition, NLI users are commonly expected to follow the system designer's conceptual models, instead of being able to express their intentions towards a goal in their own way [Jok09]. The design of a user-friendly NLI therefore requires these common pitfalls to be considered and tackled effectively.

## 5.2 Natural Language Interfaces to Conceptual Models

In the context of this thesis, the hierarchical arrangement of ontologies described in Chapter 3 enables visualization semantics to be formalized. Moreover, the semantics of the graphic objects present in a diagram may be enlarged with the inclusion of domain ontologies in the domain of what the objects represent. Therefore, to develop a Natural Language Interface (NLI) to semantically-enhanced diagrams as proposed in this thesis is to develop a NLI to ontologies. Consequently, this section describes current methods of NLI to conceptual models, and more specifically, ontologies, as an important contribution to the conceptual framework for the realized work outlined in the remaining of the present chapter.

The development of the Semantic Web (c.f. Section 3.2.1) has led to an explosion of available semantic Knowledge Bases (KBs) containing linked structured information about a myriad of

fields such as the medical and biological sciences, most commonly in the form of ontologies. The ideas and methods behind the Semantic Web, as well as a detailed characterization of ontologies, their relevance in the context of this thesis, and their different serialization formats were some of the main topics of Chapter 3 of this document. This unprecedented availability of linked data in the so-called “Web of Data” has led to the emergence of a number of question answering systems to make this information available to end users, which may not be familiar with formal query languages, such as SPARQL, traditionally employed to inquire about the information contained in a knowledge base. NLIs to KBs therefore allow users to find relevant information in an arbitrarily large knowledge base by asking questions in natural language [DLSM17]. The problem of finding correct answers to natural language questions inquired about information stored as linked data is known as *Question-Answering over Knowledge Graphs (QA-KG)* [HZLL19]. Notable examples of QA-KG systems include the QALL-ME framework [FSK<sup>+</sup>11], PowerAqua [LFMS12], FREyA [DAC12], and the GATE project [KO08a].

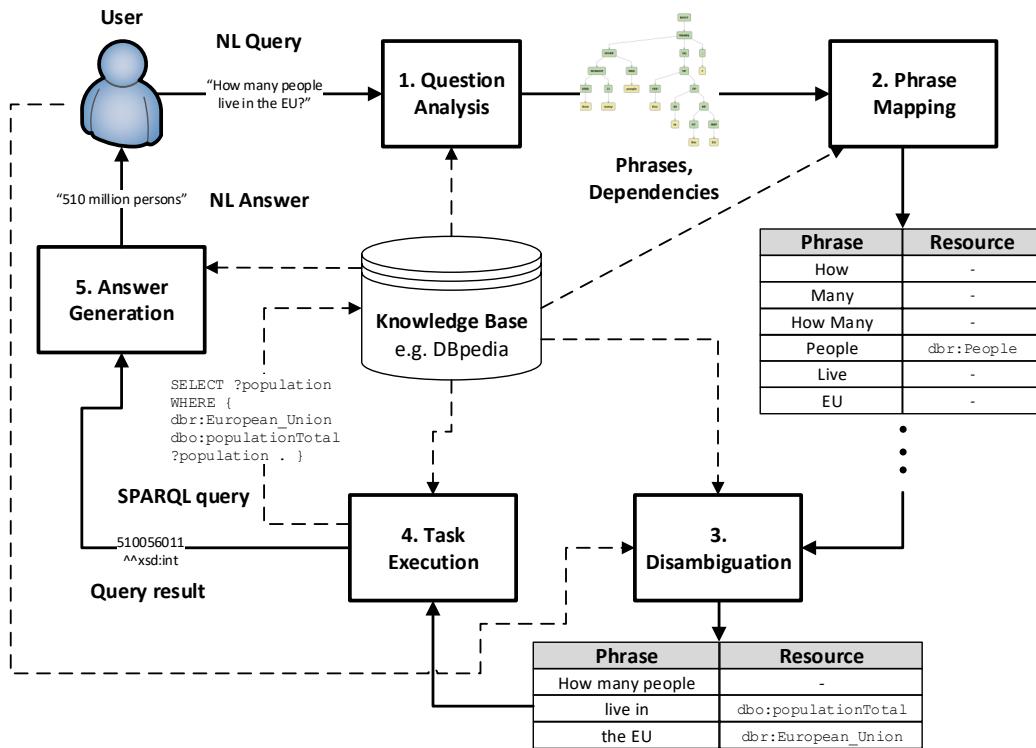


Figure 5.2: Prototypical process flow of a system for question answering over knowledge bases.

In QA-KG systems, a user's query in natural language (NL) is generally converted to a SPARQL query that is then executed against a given KB in order to retrieve the requested

information [DLSM17]. The process of converting a NL query to SPARQL is a complex, multi-step pipeline that can greatly vary depending on the selected methodology. It is usually divided in the following phases (c.f. Figure 5.2):

1. *Question Analysis.* This step analyzes the syntactic features of the input natural language query. Analyzing the syntax of the given query may be useful in order to segment it into phrases, determine in future steps which phrases correspond to which specific resources in the KB, and to find dependencies between phrases. The support from the KB in this phase is kept to a minimum. Some approaches may even forego consulting the KB altogether until the Phrase Mapping phase. Commonly employed question analysis techniques include [DLSM17]:
  - *Potential Named Entity recognition:* named entities are contiguous spans of tokens in the input query that refer to a resource. The most common and straightforward approach to recognition of potential named entities is to fetch all *n-grams* (groups of  $n$  consecutive tokens) appearing in the input query, for all  $n \in \mathbb{N}$  between 1 and the number of tokens in the sentence. This approach is known as *n-gram strategy*.
  - *POS tagging:* each token in the input query may be augmented with its corresponding *part-of-speech* (POS) tag based on its definition and context. This information can be used later on to discriminate between relevant tokens (nouns, adjectives, etc.) and those that may be safely discarded (some articles, prepositions, etc.). POS-tagging may be performed with the use of handmade rules, hidden Markov models, or a variety of machine learning methods.
  - *Dependency identification:* parsers based on formal grammars may be employed during the Question Analysis phase to identify the relations between different chunks of the sentence. *Formal grammars* consist of symbols and production rules that describe how to produce valid sentences following a language's syntax. Given the input query, a parser may then work out the combination of grammar rules that are used to generate it i.e. the grammatical structure of the input sentence. This information may be of use to improve the quality of the segmentation. For example, no named entities will be inferred from n-grams spanning more than one *noun phrase* (NP).
2. *Phrase Mapping.* The phrase mapping step tries to map each potential named entity i.e. phrase,  $s$ , in the input query to a set of resources of the underlying KB which correspond to  $s$  with a high probability. Note that  $s$  may be mapped to zero formal resources, in which case the system has to decide whether it can be ignored or to ask the user to resolve it somehow. Conversely,  $s$  may be mapped to more than one formal resource of the KB. In this case, a disambiguation phase needs to take place (see next). In addition, the *lexical gap* i.e. dealing with synonymous phrases, nicknames, misspellings, and other kinds of semantically similar but lexically different phrases, needs to be bridged in some way during this phase. Lexical databases such as WordNet and string similarity measures are generally employed to deal with this kind of problem.

3. *Disambiguation.* This step deals with ambiguities arising at any point during the previous two steps. During the question analysis step, both the segmentation of the query into phrases and the dependencies between phrases may return ambiguous results. For example, in the query “how many albums have been sold by the Rolling Stones?”, “Rolling Stones” might be interpreted as one grouped phrase or as two separate phrases. Conversely, the Phrase Mapping phase may return more than one semantic resource of the KB that are associated with a single phrase of the input query. Ambiguities may be resolved automatically or by asking the user for explicit feedback. In the former case, local disambiguation techniques such as string or semantic similarity measures between phrases and resource labels are commonly employed. The graph structure of the KB may also be used to automatically disambiguate between resources by, for example, searching for KB subgraphs that are isomorphic to a given phrase graph. Other complex ambiguity resolution methods include techniques based on Hidden Markov Models (HMM), Integer Linear programs, Markov Logic networks, and structured perceptrons [DLSM17].
4. *Task Execution.* Once all relevant resources appearing in a user’s query (plus any additional ones needed to execute the task at hand) have been found, the Task Execution phase describes how to construct one or more SPARQL queries to be executed against the knowledge base in order to fetch resources relevant to the query’s answer. A substantial challenge in the generation of SPARQL queries from user-given input is that the KB may encode relevant bits of information using different lexicalizations to those adopted by the user, a problem known as the “semantic gap” (note that this is a different notion from the semantic gap in visualization discussed in Section 3.1.2). This problem is especially significant in a NLI to diagrams, as most user queries will not include terms related to the low-level formal resources that underpin visualization. Therefore, the necessary resources of UVO, VDO, and DDO (c.f. Chapter 3) will often have to be inferred from formal resources and user terms of a higher conceptual level. Moreover, oftentimes special operations have to be performed on the retrieved resources in order to formulate the answer. For example, the average of these resources may have to be computed or a correlation analysis may take place. The execution of most of these tasks is beyond the capabilities of a query language such as SPARQL, and thus they must be carried out by client applications before returning the final answer. Query construction approaches generally fall into one of the following categories [DLSM17]:
  - *Template-based:* this kind of system uses templates to generate SPARQL queries, this is, a set of predefined queries with slots that need to be filled with query-specific information. Communicative Images, described in Section 2.6, is an example of a NLI system that employs a template-based approach to SPARQL generation.
  - *Guided mechanisms:* these systems use a heterogeneous set of techniques in which SPARQL queries are constructed with some guidance from the Question Analysis phase. For example, variables may be included in a formal query according to

their position in the parse tree of the input question in relation to the type and position of the resources identified after the Disambiguation phase.

- *Semantic parsing*: semantic parsers combine syntactic rules with semantic composition in order to parse an input query. Semantic parsers may use different kind of parsing grammars such as combinatorial categorical grammars, feature-based context-free grammars, and lexical tree-adjoint grammars. The intermediate semantic representation of the input query, expressed in a formal system e.g. lambda calculus, is then translated to the final SPARQL query.
  - *Keyword-based approaches*: in systems that do not expect syntactically valid input queries but a list of keywords, the implicit dependencies between keywords have to be inferred from the knowledge base, eschewing the Question Analysis phase of the QA-KG pipeline. Handmade rules or HMMs are employed in order to generate formal triple patterns from the input query. The main problem of this kind of approach is that ignoring the input query's syntax may result in a wrong interpretation thereof.
  - *Non-SPARQL*: some approaches do not execute a SPARQL query against the KB, instead relying on navigating it by dereferencing resources. A central term in the input query is detected in the first place. Next, using the dependency tree of the query and starting from this central term, a sorted list of phrases found in the query is generated. Then, a candidate KB resource for the central term is retrieved and its corresponding URI dereferenced to search all corresponding properties, which are afterwards compared with the next element in the list (a potential triple object). If a property is considered to be similar to the potential object, the triple (central term, property, object) is considered. This process continues with each element in the list until all have been processed.
5. *Answer Generation*: After executing the query generated in the preceding phase against the KB, a number of triples relevant to the proposed answer to the input query have been retrieved. The purpose of the Answer Generation phase is to conform these formal triples into a coherent answer in natural language that is to be displayed to the user. The *question focus*, determined during the Question Analysis phase, is the central element of this phase [DAC10a]. The focus of the question is one or more words indicating what the question is looking for. For example, in the question “what is the monthly energy generation of countries neighboring Austria?”, the focus is *monthly energy generation*. The *answer type*, which may be a class or a datatype property, is then inferred from the focus [DAC10a]. For example, the answer type for the query we just saw may be ns:generates\_energy. The answer type is then used for generating the final answer. For example, if it is a datatype property, its values may be listed to the user [DAC10a]. Moreover, the labels (`rdfs:label`) of each output resource, if available, are fetched to generate the final answer to the user. Combining these steps, a simple answer to our example query could be along the lines of “Germany consumes energy: 1,400 TWh; Croatia consumes...” and so on. The answer is lastly shown to the user, who may now proceed to input a new query into the pipeline.

It is important to note that some recent approaches to NLIs to large KBs, such as [HZLL19] and [LFLA17], do not implement such a pipeline, choosing instead to train end-to-end models, most commonly a neural network architecture, on large training corpora such that the model is able to automatically configure itself during the training process in order to generalize correct answers to yet unseen questions. However, methods based on machine learning techniques require a large corpora of training data in the form of correct question-answer pairs, none of which, as far as we know, have the accessibility of visually displayed information as their goal. In addition, most of these approaches, being in their infancy, are restricted to simple questions that only require the retrieval of a single fact (i.e. one triple) from the KB. As many of the analytical low-level tasks outlined in Section 3.9 implicate retrieving, manipulating, and reasoning with several knowledge triples, a rule- and heuristic- based pipeline like described above was considered in the design of the NLI to KB prototype designed during the development of this thesis. Such undertaking is treated in the following section.

### 5.3 AUDiaL: Accessible Universal Diagrams through Language

In the following sections we thoroughly describe *Accessible Universal Diagrams trough Dialogue (AUDiaL)*. AUDiaL (Figure 5.3) is the artifact designed within the problem context of this thesis in order to answer our main research questions, c.f. Section 1.4. In the remainder of this chapter AUDiaL will indistinctly be referred to as the “prototype” or “artifact”. Recall from Section 1.5 that the design science methodology asks for an artifact such that the interactions between the artifact and the problem context help relevant stakeholders to achieve their goals [Wie14]. We therefore aim to develop a prototype of a Natural Language Interface (NLI) to statistical charts, which are the most common type of diagram that can be found in daily life, allowing Visually Impaired Persons (VIPs) to perform common analytical tasks on diagrams through an accessible Web interface. After having completed its design, the next step of the design cycle, treatment validation, will take place in the following chapter (from Section 6.9 onward), where AUDiaL will be formally validated to resolve our initial technical research question and the related research questions that stemmed from it.

The design of AUDiaL will adhere to the schema for expressing design problems shown in Table 5.1. The problem context, stakeholders, and their goals were already outlined in Chapter 1. In short, the problem context consists on blind computer users accessing visually displayed information in statistical charts, whereas the stakeholders are the blind users themselves and the scientific evaluators of the prototype. The main goal of the blind users is to access diagrams in a effective, efficient, and autonomous manner. The goal of the evaluators is for the system to be testable with regards to its effectivity, user satisfaction, and efficiency in user-centric validation sessions with blind participants.

We observe that, at this point, there are two placeholders in Table 5.1 whose values have not been characterized yet. Namely, the requirements that the prototype must satisfy have to be specified, and a corresponding prototype design must be devised afterwards. The next section

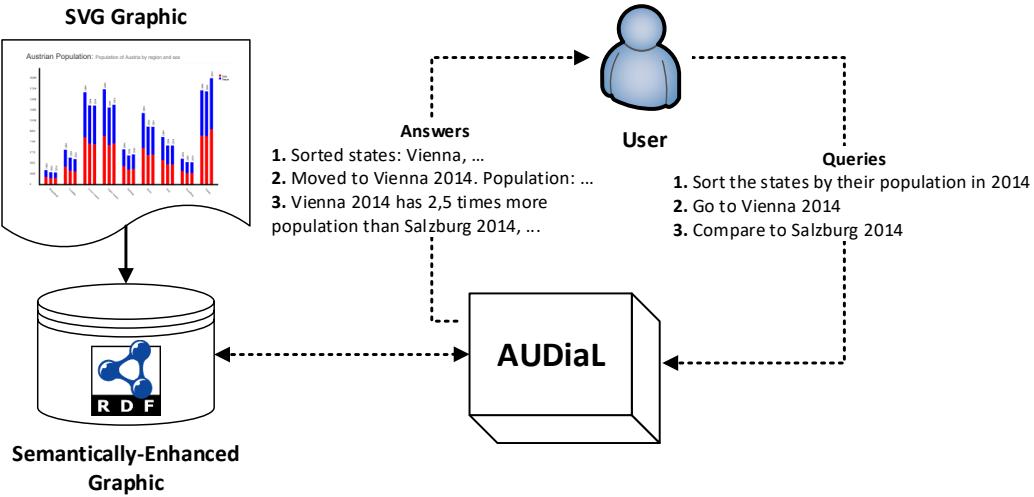


Figure 5.3: High-level overview of AUDiaL in its problem context. AUDiaL enables (blind) users access information that is visually displayed in statistical charts by means of queries performed in natural language. Example diagram, user queries, and system answers are shown.

deals with the requirements specification part of the template, whereas the sections thereafter will outline the design process of the prototype. Its validation and relevant conclusions will be discussed in Chapter 6.

- 
- Improve <a problem context>
  - by <(re)designing an artifact>
  - that satisfies <some requirements>
  - in order to <help stakeholders achieve some goals>
- 

Table 5.1: Template for design problems according to the design science methodology [Wie14].

## 5.4 Requirements Specification

Before commencing the design of a specific treatment to our problem, we must provide useful guidelines for searching possible treatments to the problem at hand. The resulting treatment i.e., AUDiaL, is the result of a design process that takes these guidelines into account. In design science, these guidelines are known as requirements. A requirement is “a goal for the to-be-designed treatment” [Wie14]. Requirements are produced by design actions, and therefore they are not meant to be found as answers to knowledge questions. Instead, they are

predictions arguing for the inclusion of an artifact in a problem context in order to contribute to stakeholder goals. Recall that the stakeholders, the stakeholder goals, and the problem context that we are considering here were previously described in Chapter 1.

In the following we outline the requirements that arose as part of the design activity of AUDiaL before its actual design took place. To justify the choice for their inclusion, each requirement is followed by one or more contribution arguments, this is, an argument that a prototype satisfying said requirement would contribute to a stakeholder goal in the problem context [Wie14]. A contribution argument has the following from [Wie14]:

*(Artifact Requirement) × (Context Assumptions) contribute to (Stakeholder Goal)*

We will begin by enumerating functional requirements. *Functional requirements* are requirements for desired functions of the prototype, where a prototype function is a terminating part of the interaction between the prototype and its context that contributes to a service to a stakeholder [Wie14]. Each functional requirement is given an identifier in the following form: **FR#n**, where **#n** is an integer such that each requirement may be uniquely identified. Moreover, this integer corresponds inversely to the priority of the requirement i.e. the importance of its goal. Following the functional requirements of the prototype we will enumerate its *quality requirements* i.e. requirements that the prototype has a specified quality property. Quality properties are artifact properties which are not a function. Quality requirements are given an identifier of the form **QR#n**, following an analogous pattern to that of identifiers for functional requirements.

**FR1:** The designed artifacts must be able to understand and resolve natural language queries aiming to perform the low-level analytical tasks, characterized by VTO, most commonly performed on statistical charts. These tasks are to be executed against a formal knowledge base containing a given chart's semantics, i.e. a semantically-enhanced diagram.

This is the main functional requirement that needs to be satisfied by the prototype. We operate under the assumption that the low-level tasks defined by the Visualization Tasks Ontology (VTO), described in Section 3.5.2, largely capture the activities performed by most (sighted) people which are supported by means of visually displayed information in statistical charts. This assumption is supported by our previous literature research on analytic activity in diagrams. Moreover, we assume that the statistical charts to be accessed have their underlying semantics (*ABox*) complying to the hierarchy of ontologies developed in Chapter 3 (*TBox*), following the annotation process outlined in Chapter 4. Therefore, the ability of supporting the realization of each of these tasks by means of natural language on semantically-enhanced statistical charts is the foundational operational layer of the prototype, being of utmost importance in order to fulfill the main stakeholder's goals of accessing and interacting with the semantics at different conceptual levels which are embedded in statistical charts. A more formal way of expressing this requirement is by stating that the functional domain of the designed NLI must include all user tasks characterized by VTO.

**FR2:** The designed artifacts must, to a reasonable extent, compensate for the lack of sight of their users when interacting with diagrams.

## 5. NATURAL LANGUAGE INTERFACE TO DIAGRAMS

---

A crucial limitation in FR1 is that it only considers the *functional domain* of the artifact without taking into account its *usability* from the point of view of the visually impaired population. Most of these covered functions (tasks) can be undertaken in an instinctive and efficient manner by sighted individuals just by looking at the diagrams by virtue of a number of cognitive benefits that exploit the perceptual, cognitive, and memorial capacities of human beings and which are enabled by means of sight. As we will discuss in greater depth later in Chapter 6, an accessible NLI to statistical charts therefore needs to implement mechanisms that support and guide blind users during the exploration of diagrams and the realization of analytical tasks on them in order to compensate for the lack of these cognitive benefits. This requirement contributes to the stakeholder goal of accessing diagrammatic representations in an interactive, efficient manner that somewhat resembles the way sighted users consume them.

**FR3:** Queries and actions performed by users on the prototype must be logged and safely stored. Timestamps for each query and action, tied to the specific user undertaking them, need to be included as well.

The main reference of system usability is the ISO 9241-11 standard [Int18]. It is made up of three different usability metrics; namely, effectiveness, efficiency, and user satisfaction. In order to measure effectiveness i.e. the rate of user task goal completion, artifact evaluators need to analyze the users' input and the system output to these inputs in order to study whether users were able to complete the given tasks using the prototype and the degree of accuracy of their answers. Therefore, all interactions between a user and the prototype during the task solving process need to be logged by the system (bound to the individual user undertaking the task) and properly stored. In addition, in order to measure time-based efficiency i.e. the amount of time it took users to solve each task, timestamps need to be included on each system log. In this manner, the total time it took a given user to solve a certain task can be measured by subtracting the timestamps between the initial user query input and the final response output by the system. In order to measure user satisfaction user surveys will be employed, and as such no special considerations need to be taken during prototype design regarding this last usability metric.

This requirement operates under the assumption that measuring and analyzing these standardized metrics on the designed prototype while interacting with a model of the problem context will suffice to justify that such treatment would contribute to stakeholder goals when implemented in the problem context in order to obtain answers to the knowledge questions initially posed. This requirement contributes to the stakeholder (the researcher) goal of answering a knowledge question.

**FR4:** All phases of the question answering over knowledge graphs pipeline (c.f. Figure 5.2) must be included in the design of the artifact.

The designed prototype needs to be realized enough as to be usable by participants during the design validation phase without the arousal of any significant functional issues that may interfere with the evaluation results. Therefore, all aspects of modern natural language interfaces to knowledge bases must be available in the final prototype. Approaches for each

phase of the natural language to SPARQL pipeline outlined in Section 5.2 stemming from those found in the literature will have to be implemented and tested on knowledge bases complying with the hierarchical structure of formal diagram semantics defined in Chapter 3. Satisfying this requirement would contribute to the quality of the design validation results, which is a vital stakeholder goal from the point of view of the researcher.

**FR5:** The prototype needs to learn and adapt itself to the language of its users by automatically mapping newly seen meaningful concepts to resources in the KB. A learning subsystem must be included to the design of the artifact.

Section 5.2 briefly introduced the prevalent “semantic gap” problem in the NLI to KB context. The semantic gap refers to the need for bridging the “gap between the way the user thinks about the domain of discourse and the way information about the domain is structured for computer processing” [GAMP87]. For example, answering the question “How many people live in Vienna” on a given KB about Austrian demographics may involve retrieving the following triple: (ns:Vienna, ns:has\_inhabitants, “1800000”^^xsd:integer). A *transportable* (or, equivalently, *habitable*) system must be able to ascertain that the ns:has\_inhabitants datatype property is relevant to the user’s query, even if the word “inhabitants” does not appear anywhere in it.

A common way of solving the semantic gap problem is by employing clarification dialogues [DACB13]. This technique engages the user in a dialogue whenever the system fails to solve any arising ambiguities automatically. Potential interpretations of the query are shown to the user, who is then in charge of selecting the correct one. Moreover, the system can learn from the answers given to clarification dialogues by its users which user lexicalizations can be mapped to which formal resources in a given KB, reaching the point where common mappings can be performed automatically by the system. In doing so, the collaborative intelligence of its users can be exploited to improve the prototype’s recall, which in turn should improve the general user satisfaction of the system. This requirement is related to FR4 in that both requirements aim to enhance the usability of the prototype as to not to hinder its future evaluation, since the habitability of a NLI affects its usability [DACB13]. Satisfying this requirement would thus similarly contribute to the quality of the design validation results.

**QR1:** The designed artifacts should be fully accessible to visually impaired computer users using standard Assistive Technology (AT). Any Web content must meet the required WCAG success criteria in order to reach level AA.

The principal quality requirement for the artifact is to ensure its proper operability by the visually impaired population by providing full co-operation with standard AT, more specifically, screen readers such as Jaws and NVDA (c.f. Section 2.1.5). This requirement can be achieved by following the latest version of the Web Content Accessibility Guidelines (WCAG) [Kir18] during the design process of the artifact, as introduced in Section 2.7.1. Reaching the level AA of the WCAG 2.1 guidelines ensures that the artifact will meet the access needs of its users during evaluation without having to carry out the more constraining and time consuming optional success criteria required for a AAA level of compliance.

This requirement operates under the assumption that the participants that will interact

## 5. NATURAL LANGUAGE INTERFACE TO DIAGRAMS

---

with the artifact during validation sessions are proficient screen reader users. Satisfying this requirement contributes to the stakeholder goal of accessing the Web interface of the artifact in an intuitive manner employing everyday tools for navigating the WWW.

**QR2:** The designed artifacts should be intuitive to use by the stakeholders. More specifically, end users must be able to interact with it without undertaking any prior training specific to the software. The language supported by the prototype must be *habitable*.

An initial version of AUDiaL, not described here, included one input text field for each of the tasks characterized by VTO<sup>1</sup>. The NLP capabilities of this initial prototype were based on a simple keyword search, and users could solve each task by inputting the right keywords on the text field corresponding to the task they aimed to solve. For example, retrieving the average energy production of the year 2010 involved typing something along the lines of “energy 2010” in the “Average” text field. However, initial heuristic evaluation with sighted participants yielded that users could become easily confused by the amount of available input fields and the reduced syntactic and lexical domains covered by the prototype’s NLP capabilities. This would have resulted in very low efficacy and usability scores in the evaluation of the final prototype.

Therefore, an important quality requirement for the designed artifact is that users must be able to construct expressions in natural language that they have not previously encountered i.e. the language supported by the artifact must be *habitable*. This requirement operates under the assumption that stakeholders will not have any available external help during usage of the artifact. For example, participants taking part in evaluation sessions must not be allowed to ask the researcher or any sighted persons for help during the full duration thereof. Satisfying this requirement would, on the one hand, contribute to the researcher’s goal of answering a knowledge question. On the other hand, satisfying this requirement also contribute to the stakeholder’s goal of accessing diagrammatic representations in an interactive, efficient, and autonomous manner.

**QR3:** Answering times to most user queries should be kept very short (average response time  $\leq 2$  seconds). In complex queries, answering times may take longer, but must stay under 10 seconds in any case. Feedback during waiting times should be directly provided to end users.

Studies in Human-Computer Interaction (HCI) have shown that the tolerable waiting time for information retrieval tasks performed on Web sites is approximately 2 seconds, where tolerable waiting time is defined as “the amount of time users are willing to wait before giving up on the download of a Web page” [Nah04]. Assuming that these tolerable waiting times are also pertinent to content that is being dynamically embedded into a Web dialog system as a response to a user’s query, we stipulate that the average time that passes between a user inputs a query in the prototype and a response (such as an answer or a clarification dialog) is output by the system should be kept under 2 seconds. Exceptionally, user queries mapped to

---

<sup>1</sup>A paper on the initial prototype AUDiaL is built upon was published in: Tomás Murillo-Morales and Klaus Miesenberger. Non-visually performing analytical tasks on statistical charts. In P. Cud and L. de Witte, editors, *Studies in health technology and informatics*, volume 242, pages 339–346. IOS Press, 2017

complex background tasks inflicting high server computational load may take longer than 2 seconds. In these cases, waiting time should be kept under 10 seconds, as this has been shown to be the maximum tolerable delay before user's attention on the dialogue is lost [Nah04]. Moreover, system status feedback should be given to users during waiting periods, as this has been shown to prolong Web user's tolerable waiting time and improve system usability [Nah04, HMSD10]. Satisfying this requirement would contribute to the overall usability of the prototype, which in turn would ensure that its future usability evaluation will not become influenced by external factors unrelated to the artifact's functionality.

**QR4:** Synchronous page loads should be kept to a minimum. User queries sent to the server as well as responses received from the server should be performed in an asynchronous manner.

When using a client-server architecture, the look and feel of the client Web page should resemble that of a traditional desktop application in order to improve the overall usability of the system [HMSD10]. This calls for removing, where possible, the synchronous communication between the client and the server in favor of asynchronous exchange mechanisms such as AJAX. Employing AJAX in the architecture of Web applications have been shown to improve their learnability, efficiency, and memorability [HMSD10]. In turn, a more usable system allows stakeholders to focus on aspects related to the technical research questions at hand, without becoming distracted by unrelated usability issues, thus contributing to the main stakeholder goals of the design cycle. This requirement operates under the assumption that stakeholders are proficient Web users that will be operating the prototype through a combination of commonplace operating system, Web browser, and assistive technology.

## 5.5 NLI Prototype Design

After having defined the requirements of AUDiaL in the previous section, all the necessary information to begin with the treatment design phase of the design cycle (see Table 5.1) has been gathered. This phase started, as outlined in Section 5.5.1, with general architectural considerations that concluded on the need of implementing the prototype as a Web application where most of the workload is tackled by an external Web server and supporting services. The accessibility of the Web user interface, discussed next in Section 5.5.2, is a critical aspect to be considered for the proper actioning of the system by its target users. Afterwards, the design of each individual phase of the natural language to KB processing pipeline (c.f. Figure 5.2) will be characterized in corresponding sections.

### 5.5.1 General Architecture

The prototype designed in this chapter needs to implement a Natural Language Interface (NLI) to semantically-enhanced diagrams, more specifically semantically-enhanced statistical charts (following the format described in Chapter 4), such that Visually Impaired Persons (VIPs) may effectively perform common analytical and wayfinding tasks on them. In order to improve its organization, ease of debugging and maintenance, and future collaboration among developers, AUDiaL has been designed following a modular architecture, in which the

functionality of the artifact is divided into independent, interchangeable modules specialized on specific tasks such as interacting with a particular Knowledge Base (KB), processing natural language, storing data, or logging user information. As such, the main component of the system is the software *controller* which is in charge of orchestrating the constituent modules of the system so that the required functionality can take place.

An initial consideration is where to place the controller and dependent modules with respect to the source files containing the image data and its semantics that will be queried by the user. Kopeček and Ošlejšek discuss two possible configurations for the location of the code in charge of handling the interaction between a user and a Communicative Image [KO11]:

- Embedding JavaScript code within the SVG mark-up of the image. This solution presents several hindrances for its consideration in a scalable and usable framework. Namely, this approach cannot be employed in commonly used raster image formats (c.f. Section 4.1). Moreover, the computational capabilities of a script programming language running locally on the user's computer may be too limited for some complex tasks supported by the image's formal semantics to be performed.
- Bundling the code in a Web browser extension that handles tasks performed on images appearing in Web content. This approach has the advantage that the user may interact directly with images that he or she encounters during normal Web browsing activities. However, having the dialogue strategy and task handling take place within a local extension also presents many drawbacks arising from the limited computational capabilities of the user's local set up. This idea has been reinforced by other research where it has been concluded that software tools for non-visual access to statistical charts should not be implemented as standalone applications, but instead as Web applications [FLST13].

In order to overcome the limitations of the approaches we have just discussed, a client-server application model for AUDiaL, as shown in Figure 5.4, was selected. On the client-side of the model, the user navigates by means of an accessible Web page using any common Web browser (such as Mozilla Firefox<sup>2</sup> or Google Chrome<sup>3</sup>) with the aid of everyday assistive technology such as a screen reader and text-to-speech (TTS) software. Once on AUDiaL's Web front-page (described in the following section), the user may perform tasks on the available semantically-enhanced graphics by means of natural language queries through a standard HTML text field for input. The query is then sent to an external server hosting the application's back-end logic.

The server will generally have a much higher processing speed and memory capacity compared to the client's computer, which results in faster task completion speeds and decreased waiting times than if the logic was fully executed on the client's side. In addition, the Question Analysis phase of the natural language to KB pipeline (see Section 5.2) can be relegated to

---

<sup>2</sup><https://www.mozilla.org/en-US/firefox/new/>

<sup>3</sup>[https://www.google.com/intl/en\\_us/chrome/](https://www.google.com/intl/en_us/chrome/)

another external Web service, such as a Stanford CoreNLP Server [MSB<sup>+</sup>14] instance, which also helps increase the overall computational capacity of the system. AUDiaL's server side has been developed using the modern WSGI Web application framework known as Flask<sup>4</sup>, which enables Web applications to be developed in a quick, easy, and scalable manner. Moreover, being a Python framework, it allows developers to employ many state-of-the art Python libraries for natural language processing, RDF manipulation, and machine learning.

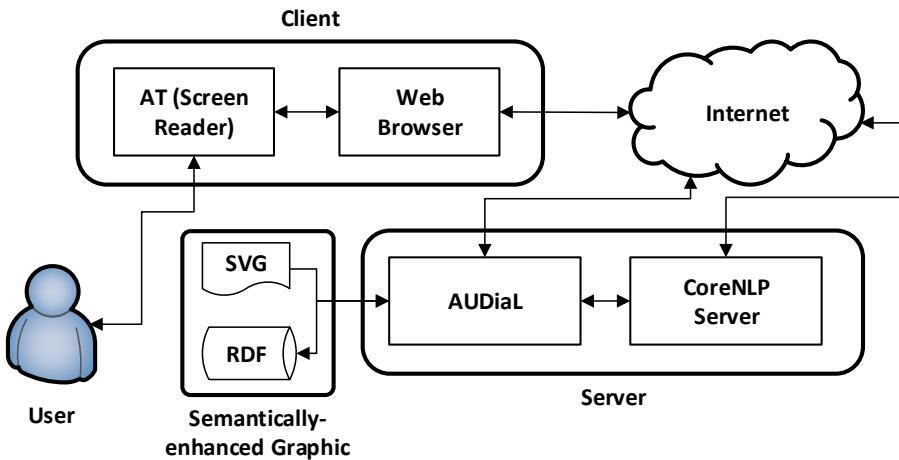


Figure 5.4: Client-server application model of AUDiaL.

Besides the initial HTTP request that the client must perform in order to load the Web page, further communication between the client and the server is done by means of methods of the XMLHttpRequest (XHR) standard API<sup>5</sup>. Each user query is therefore sent to the server in an asynchronous manner without the need for a page reload. Moreover, responses from the server (text-based, natural language answers to user's queries) are also received via XHR calls and dynamically fed into the Web page interface. By employing a client-server architecture in conjunction with asynchronous communication mechanisms between them we ensure that the resulting artifact satisfies quality requirement QR4 as well as aiming for the fulfillment of requirement QR3 as described in Section 5.4.

One aspect of the prototype not discussed thus far is the placement of the semantically-enhanced graphics (both the SVG data and its associated KBs) within the client-server architectural model of the system. As the system logic is hosted in the server-side of its architecture, the most suitable approach in order to meet the relevant quality requirements is to host the semantically-enhanced graphics also on the server in order to prevent additional overhead in the form of network messages that would have to be sent between the controller and the queried knowledge which is stored in the client. Therefore, the prototype may allow

<sup>4</sup><http://flask.pocoo.org/>

<sup>5</sup><https://xhr.spec.whatwg.org/>

users to upload their own semantically-enhanced graphics to the server via an accessible Web interface, but will not operate on any data which is locally hosted in the client.

### 5.5.2 Accessible User Interface

The main Web interface of AUDiaL, as rendered by Mozilla Firefox, is shown in Figure 5.5. It is developed using standard HTML5<sup>6</sup> elements with separate styling rules defined with CSS<sup>7</sup>, a language for describing the rendering of structured documents such as HTML.

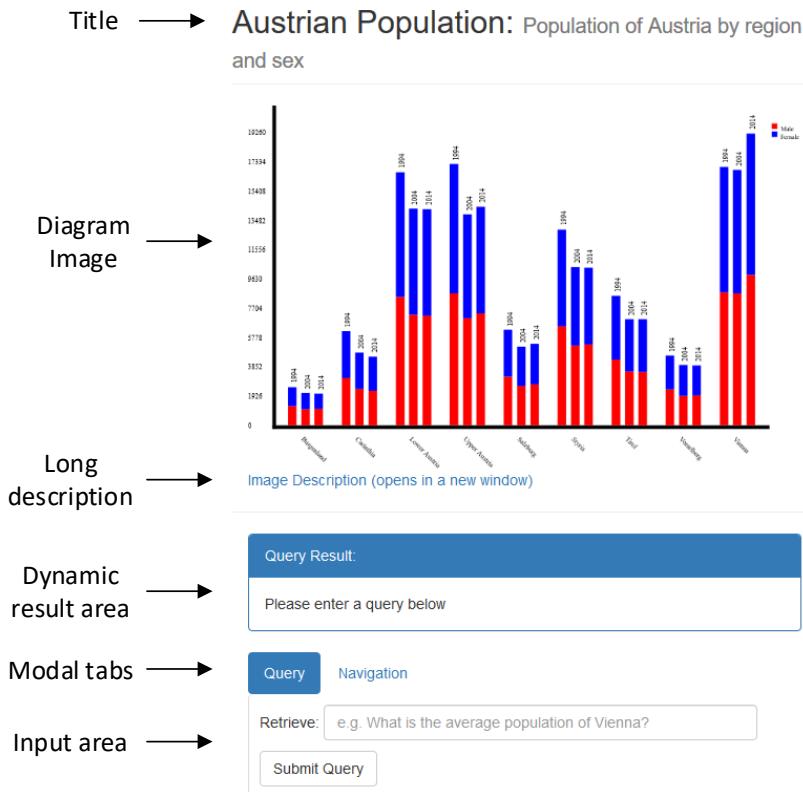


Figure 5.5: Main elements of AUDiaL’s Web interface (cropped). The GUI is shown before the user has performed any action.

The Web user interface is comprised of six main sections, as follows:

- Title and subtitle of the diagram at hand, described by an `<h2>` HTML tag.

---

<sup>6</sup><https://html.spec.whatwg.org/>

<sup>7</sup><https://www.w3.org/Style/CSS/>

- Image data of the diagram in SVG or another supported image format, embedded in an `<img>` tag (optional). If available, the `alt` attribute of the image will provide any assistive technology client with a short description of the diagram contents.
- A hyperlink to the diagram's long description, if available, following the W3C recommendation for non-visual alternative description of complex images [Egg14].
- Result area. This is the place where any answer or feedback generated by the back-end is dynamically fed into the user interface via XHR calls (see Section 5.5.1). This includes textual answers to a user's query, waiting and error feedback, user task results, and any dialogue with the user.
- Modal tab area. This area allows users to select the current mode of the user interface. UI modes are kept to a minimum in order to reduce user confusion; the need for their inclusion in the prototype will be discussed later on.
- Input area. This section displays the necessary HTML elements that allow users to perform tasks or ask natural language queries on the current diagram. The main input area consists of a text field where users can input a textual query in natural language and a submit button to send the query to the back-end for processing.

In order to satisfy requirement QR1, as specified in Section 5.4, the user interface has been designed following best practices with the goal of meeting WCAG 2.1 up to AA level. Due to the dynamic nature of the designed interface, relevant aspects from the WAI-ARIA suite have been included to the GUI elements as well. These aspects of AUDiaL's interface will be discussed in detail in Section 6.8. In the following, we describe the basic specific technical solutions included in AUDiaL's user interface in order to make it accessible to blind persons using a screen reader to navigate it.

The source HTML code of the UI's dynamic result area is shown in Figure 5.6. WAI-ARIA roles are employed throughout the main container in order to add accessibility semantics to its descendant elements. The 'Query Result' container is given a secondary heading role (as the main page header is the diagram title and subtitle shown in Figure 5.5) by means of `role="heading"` and `aria-level="2"` attributes. In this manner, screen readers are informed of the semantic role of the `<div>` element allowing blind users to focus on the element and quickly access it by means of keyboard shortcuts. The dynamic result area itself consists of the following `<div>` element (with id 'query-result'). Each time a XHR response is received, the `div` element's HTML content is automatically updated with the help of the JQuery JavaScript library<sup>8</sup>. Adding an `aria-live` attribute to the container converts it to an ARIA live region that notifies screen readers of any new changes in its content. Setting the `aria-live` attribute's value to "polite" does not interrupt users during their browsing activity, because screen readers will only speak out any changes when the user is idle. Moreover, the inclusion of the `tabindex="0"` attribute allows Web browsers to

---

<sup>8</sup><https://jquery.com/>

```

<div id="result-container" class="container-fluid">
  <div class="panel panel-primary">
    <div id="query-heading" class="panel-heading" role="heading" aria-level="2">
      Query Result:
    </div>
    <div class="panel-body" id="query-result" aria-live="polite" tabindex="0">
      The average is 747231.67 persons<br>
      <h5>The following 3 bars have been considered:</h5>
      <ul>
        <li>Stacked bar with labels: 1994, Tirol (849663 persons).</li>
        <li>Stacked bar with labels: 2004, Tirol (695998 persons).</li>
        <li>Stacked bar with labels: 2014, Tirol (696034 persons).</li>
      </ul>
    </div>
  </div>
</div>

```

Figure 5.6: Source code of AUDiaL’s dynamic result area displaying the answer for the input query “what is the average population of Tirol?”.

focus on the result area. In this way, focus to the new content is given programmatically after submitting a query in order to guide users to their query’s answer.

Figure 5.6 also shows the answer to a typical user query embedded into the dynamic result area container element. We ensure that all textual answers given to the users are fully accessible by keeping them simple and using the innate semantic structure of HTML elements. In the example shown, the answer is divided into three parts; the main factoid-type answer to the user’s query is given first (“the average is...”). Next, additional information is provided as part of the full answer. Any supplementary information output by AUDiaL is preceded by a `<h5>` header tag so that screen reader users can quickly and efficiently jump from one to another by means of keyboard shortcuts. Information requiring of more elaborate markup, such as lists, tables, and long summaries, are always formatted using the correct HTML elements for their intended purpose, such as `<ul>` and `<li>` elements for unordered lists, as shown in the example.

The HTML code for the modal tabs area of the GUI is shown in Figure 5.7. Their containing `<ul>` element is assigned the `tablist` ARIA-role, which defines a list of tab elements. Each tab within the tab list is characterized by a `<li>` element with a child `<a>` (anchor) element that the user may click or press enter on when focused so that the associated tab panel (a `<div>` element with ARIA-role `tabpanel`) is displayed. The content of each tab panel is also made accessible by means of corresponding ARIA-roles (e.g. group to group related interface elements together) and HTML elements (e.g. `<button>`). Moreover, the user focus is assigned automatically to the relevant element, such as the dynamic result area, after a button or tab in the user interface is clicked. The elements in the ‘Navigation’ pane, shown in Figure 5.7, allow users to perform certain navigational tasks, such as quick jumps between chart elements, and to annotate individual chart elements with user tags and special

roles. The reasons for the inclusion of these components to enhance the basic capabilities of the NLI and their functionality will be discussed at length in Section 6.1.

```

<ul id="navtabs" class="nav nav-pills" role="tablist">
...
<li role="presentation" class="">
    <a href="#overview" aria-controls="overview" role="tab"
       data-toggle="tab" aria-expanded="true">
        Navigation
    </a>
</li>
...
</ul>
<div class="tab-content">
...
<div role="tabpanel" class="tab-pane query-box active" id="overview">
...
<div class="row">
...
    <div class="col-xs-4 col-md-3 text-center">
        <h4 class="text-center text-info">
            Quick Jumps
        </h4>
        <div class="btn-group-vertical" role="group"
             aria-label="Quick Navigation Links">
            <button id="submit-first" class="btn btn-default clr-result">
                Go to First Bar
            </button>
            <button id="submit-last" class="btn btn-default clr-result">
                Go to Last Bar
            </button>
            ...
        </div>
    </div>
</div>
</div>

```

Figure 5.7: Partial source code of AUDiaL’s modal tabs and input areas.

### 5.5.3 Natural Language to Knowledge Base Pipeline

After a natural language query has been input by the user in the input area of the accessible user interface (Section 5.5), a XHR request containing the query is sent to the server-side of the prototype for further processing. There, as per requirement FR4 (c.f. Section 5.4), the input query is handled by a processing pipeline, like the one shown in Figure 5.2, that attempts to compute an answer to the input query by consulting and treating the available Knowledge Bases (KBs) underpinning diagram semantics and, optionally, any available domain ontology.

In the forthcoming sections the following example input query will be considered for illustrative purposes: “*What is the average women’s population in Vienna after 1994?*”. This query will be executed against the semantically-enhanced bar chart whose image is shown in Figure 5.5,

which displays the population for the years 1994, 2004, and 2014 in each Austrian state, divided by gender. We will study how this input query is processed and transformed by each sub-module of AUDiaL’s processing pipeline in order to execute the relevant task on the semantically-enhanced diagram and return an answer to the user. As it will become apparent while analyzing the way it is processed by the pipeline, this input question, although rather simple, is a very illustrative example that allows us to demonstrate the inner workings of the NLI prototype in a detailed manner. The first phase of the processing pipeline, as shown in Figure 5.2, is the Question Analysis phase. The design of this phase in AUDiaL will be described in next section. Each subsequent section will illustrate the remaining phases of the pipeline.

## 5.6 Question Analysis Phase

The first phase of AUDiaL’s processing pipeline, namely *Question Analysis*, is illustrated in Figure 5.8. It can be observed that it is composed of two independent modules. As already discussed in Section 5.5.1, these two independent modules run separately in order to improve the response times of the prototype. In the following these modules are described along with their sub-components, which, unless otherwise stated, are executed in order and fed the output of the previous one

### 5.6.1 CoreNLP service

The *CoreNLP service* module is an external instance of the Stanford CoreNLP Natural Language Processing Toolkit [MSB<sup>+</sup>14], which is in charge of generating the syntax Parse Tree (PT) and Part-of-Speech (PoS) tags of the input query, described next. Its methods are accessed by AUDiaL via the Stanford CoreNLP API available in Python’s NLTK<sup>9</sup> library (`nltk.parse.corenlp`). This structuring of the input query will serve as basis for most of the question analysis done by AUDiaL immediately afterwards.

- A *Parse Tree (PT)*, or syntax tree, is an ordered, rooted tree that represents the syntactic structure of a sentence according to some context-free grammar. The PT for the user’s query is a constituency-based parse tree whose leaf nodes consist of the individual lexical tokens of the input query and whose interior nodes group their descendant nodes into non-terminal grammar categories of the Penn Treebank II tag set [TMS03]. For example, a ‘NP’ node indicates that its descendant leaf tokens (e.g. “the average women’s population”) belong to the same *noun phrase* chunk<sup>10</sup>. The Stanford Parser is a Java implementation of a number of probabilistic natural language parsers that try to produce the most likely analysis of new sentences.
- *Part-of-Speech (PoS)* tagging refers to the process of assigning the corresponding part of speech, such as noun, verb, or adjective, to each lexical token of a sentence. This

---

<sup>9</sup><https://www.nltk.org/>

<sup>10</sup>See <https://gist.github.com/nlothian/9240750/> for the complete list of Penn Treebank II tags.

information can be derived from the PT of the sentence by observing the parent node of each leaf node of the syntax tree. Another possibility is to employ the Stanford Log-linear PoS Tagger, which is a Java implementation of the PoS tagger described in [TKMS03]. The PoS tags used by the Stanford CoreNLP service are a subset of the Penn Treebank II tag set [TMS03].

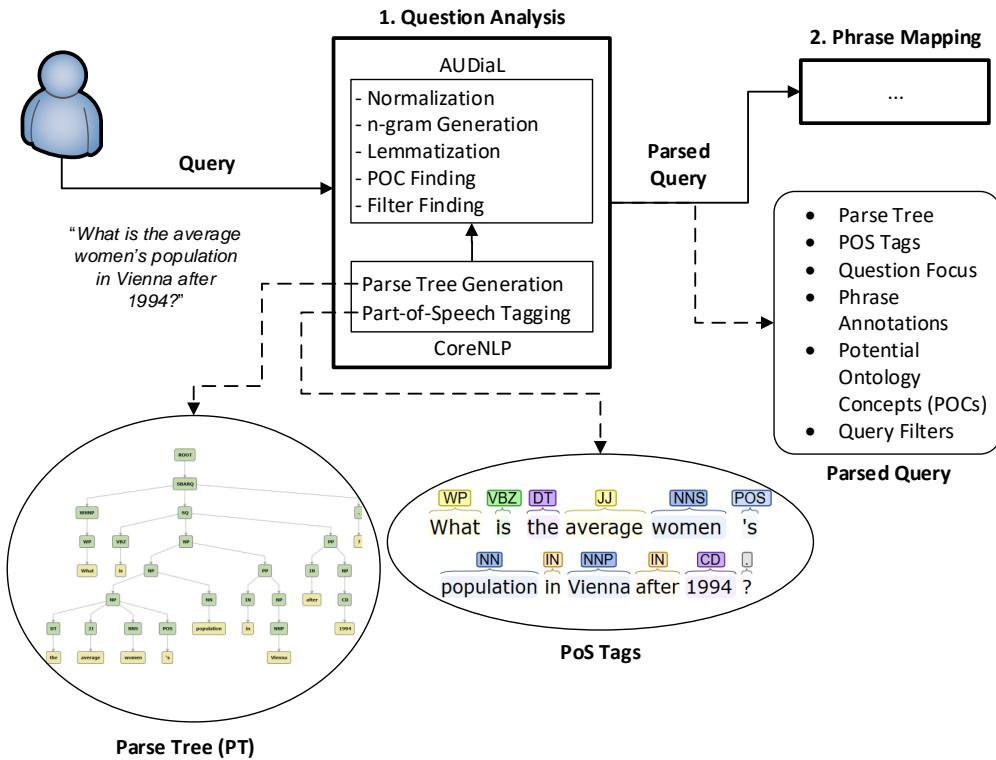


Figure 5.8: Question analysis phase of AUDiaL. A raw user query is processed resulting in a parsed query object.

### 5.6.2 AUDiaL Question Analysis

This module prepares the input query to be further processed in the Phrase Mapping phase by creating a *parsed query object* containing the tokens and phrases that will be searched in the knowledge base as well as additional syntactic information, such as the *question focus*, that will be necessary later on during the processing pipeline. The most important concepts derived from this phase are Potential Ontology Concepts (POCs) and Query Filters (QFs), which are the syntactic foundations of the entire question answering process. The Question Analysis phase is divided into six main steps, namely:

- *Normalization.* First, the raw input query is pre-processed in a fast manner by converting it to lowercase format, removing punctuation and excess whitespace characters, and removing trailing question or exclamation marks. The normalized text, now in a simple canonical form which is easier to process, is delivered to the CoreNLP service previously discussed (Section 5.6.1) as well as to the word n-gram generation phase, see next.
- *Word N-gram Generation.* During this phase, the normalized input query is divided into a number of sequential tokens of variable length  $n$  (i.e. *n-grams*), where  $n$  may be any integer between 1 and the total number of lexical tokens in the sentence as given by its parse tree. Instead of considering every possible combination of  $n$  contiguous words, n-grams are generated by aggregating the leaf nodes of all sub-trees of the parse tree. N-grams are further augmented with information about their start and end *offsets* in the ordered list of lexical tokens of the full normalized sentence, their corresponding sub-tree of the PT, and whether they have to be lemmatized later on. A given n-gram is considered for lemmatization if it is a single-token plural noun phrase (NP), a higher-level NP, or a clause with an NP gap. We call an augmented n-gram an *annotation*.
- *Lemmatization.* Those n-grams that have been marked for lemmatization during the previous phase are divided into individual tokens and each of them is then sent to NLTK's WordNet Lemmatizer to be lemmatized. *Lemmatization* is a special case of normalization that reduces the morphological variation of a phrase by finding the root form of its inflected words. For example, the word “women” has “woman” as its lemma. This is required as a previous step to ontology search of phrase synonyms during the Phrase Mapping phase, as the key in synonym search cannot be an inflected form of the word. Using a lemmatizer (e.g. WordNet Lemmatizer) instead of a stemmer yields better results since the word context is taken into account through its PoS tag and the use of a corpus such as WordNet.
- *Question Focus Identification.* As introduced in Section 5.2, the *focus* of a question is “the word or sequence of words which define the question and disambiguate the question by indicating what the question is looking for” [MHP<sup>+</sup>00]. In our example question, the focus is *average women’s population*, see Figure 5.9. Determining the user focus is useful to identify the *answer type* of the question as well as to prearrange ambiguous concepts to be resolved during the Disambiguation phase of the NLP pipeline. AUDiaL follows the method described in [DAC10a] to identify the question focus and the *head* of the focus, based on a simple syntactic analysis of the query's parse tree.
- *Potential Ontology Concept (POC) Generation.* Derived from their use in FREyA [DAC12], POCs are inferred from the PT of the query in an unsupervised manner. The idea behind POCs and their identification procedure are outlined in subsection “Potential Ontology Concepts” below.
- *Query Filter (QF) Identification.* Query filters specify actions for refining data from the knowledge base before performing an analytical task. They are described last in this section.

## Potential Ontology Concepts

*Potential Ontology Concept (POC)* are question terms (word n-grams), derived from the parse tree of a natural language query, that have the potential of being linked to formal resources of the KB [DAC10b]. The identification of POCs is done before the KB is consulted in the Phrase Mapping phase of the NLP pipeline by following a number of heuristics on the parse tree of the query. Most noun phrases, nouns, adjectives, and some verbs and adverbs appearing in the query will be mapped to a corresponding POC by AUDiaL's POCCreator module.

The pseudo-code of the implemented POC finder is shown in Algorithm 5.1. It follows similar heuristics to those of FREyA with some alterations. The most important modification we have included to the original algorithm is the additional consideration of verb phrases whose PT roots appear in a pre-terminal position in the original PT of the input query. This is because the user tasks to be performed will be searched against formal resources characterized by VTO (c.f. Section 3.5.2), which are often expressed in a verbal form. The POC finding algorithm, although simple, offers the advantage of being able to process ill-formed questions and question fragments as well as syntactically valid queries in a very quick manner [DAC12]. Each identified POC must be resolved later on, during the Phrase Mapping phase of the NLP pipeline, to zero or more resources of the underlying KB, which are ultimately employed to compute the final answer to the query. Therefore, the identification of POCs is of utmost importance to the future resolution of the query. A POC consists of a sub-tree of the input query's PT, which is automatically augmented with the following fields:

- *Head*: the head, or *nucleus*, of a phrase is the word that determines the syntactic category of the phrase. For example, in the noun phrase (NP) “the average women’s”, the head of the phrase is “women”, as it is the only noun in the phrase. A widely used method for finding the head of a NP can be consulted in Appendix A of Collin’s dissertation [Col99]. The head of a POC is a POC instance as well.
- *Modifiers*: the modifiers, or *dependents*, of a noun phrase are words which modify the head of the phrase by changing, clarifying, qualifying, or limiting its meaning in some manner. For instance, in the noun phrase “the average women’s” modifiers are “the average” and “’s”, as they modify the head of the phrase in some way. They have been implemented as a list of parse sub-trees of the POC’s PT.
- *Start and end offsets*: each POC is assigned start and end indices, according to the position that their tokens occupy within the global input sentence’s token list. This is necessary as the same sequence of words may appear more than once in a sentence and must be told apart from each other.

**Algorithm 5.1:** Potential Ontology Concept (POC) identification

---

```

1 function get_POCs (query);
  Input : A query in natural language query
  Output : List of POC instances pocs
2 PRE_PRETERMINAL_LABELS := ["NP", "NN", "NX", "ADJP"];
3 PRE_PRETERMINAL_WH_LABELS := ["WHNP", "WHADJP", "WHAVP",
  "WHAVDP"];
4 CHILD_IGNORE_LABELS := ["PRP", "EX"];
5 pocs ← array();
6 p_tree ← parseTreeOfQuery(query);
7 pre_pre_terminals ← prePreTerminalsOfTree(p_tree);
8 foreach ppt ∈ pre_pre_terminals do
9   | children_trees ← getChildren(ppt);
10  | children_labels ← getLabels(children_trees);
11  | if (ppt.label ∈ PRE_PRETERMINAL_LABELS) and not (any
    |   children_labels ∈ CHILD_IGNORE_LABELS) then
12  |   | pocs.append(ppt);
13  | end
14  | else if ppt.label ∈ PRE_PRETERMINAL_WH_LABELS then
15  |   | if any (contains_adjective(children) or contains_adverb(children))
    |     | then
    |       |   /* Split parse tree into sub-trees containing
      |         |   adjectives, adverbs, and the rest */  

16  |       |   | poc_JJ, poc_RB, poc_other ← split(children);
17  |       |   | pocs.append(poc_JJ);
18  |       |   | pocs.append(poc_RB);
19  |       |   | pocs.append(poc_other);
20  |     | end
21  | end
22 end
23 pre_terminals ← preTerminalsOfTree(p_tree);
24 foreach pt ∈ pre_terminals do
25  | if pt ∈ pocs then
26  |   | go to 24;
27  | end
28  | else if is_noun_(pt) or is_adverb(pt) or is_verb(pt) then
29  |   | /* Certain common verbs and adverbs are ignored in
    |     |   order to reduce the amount of POCs */  

30  |   | if not ignore(pt) then
31  |   |   | pocs.append(pt);
32  |   | end
33 end

```

---

## Query Filters

*Query Filters (QFs)* are terms appearing in the input query that modify the selection of triples retrieved from the knowledge base before executing a task or the triples that are output as a result of executing a task. As introduced in Section 3.9.1, filter low-level tasks (also known as *geometric zooming* in the HCI literature) are characterized by the `vto:Filter_Task` class of VTO. They are however handled differently than other VTO tasks by AUDiaL due to their elementary nature and precedence over other tasks. We distinguish two types of QFs depending on their filtering target, namely:

- *Nominal Query Filters (NQFs)*. These terms filter the Knowledge Base (KB) by matching datatype triple objects against a nominal value. For example, in the query “What is the average women’s population in Vienna after 1994?”, potentially identified NQFs would be “women”, “population”, “Vienna”, and “1994”. Because NQFs need a fully consolidated query, they are computed during the Task Execution phase of the NLP pipeline (Figure 5.2). Otherwise, NQFs would be created for POCs that would subsequently be resolved to resources in the KB, wasting computational time.
- *Cardinal Query Filters (CQFs)*. These filters match a set of KB triples against an *operator* and an *operand* which together constitute a logical condition. A CQF indicates that those resources for which the condition does not hold true must be filtered out of from a triple set. For example, in the query “What is the average women’s population in Vienna after 1994?”, the only CQF is the phrase “after 1994”, in which the word “after” is mapped to the greater than ( $>$ ) operator, and “1994” is considered as its right-hand side operand. The operators detected by AUDiaL’s `filterCreator` module are ‘greater than’, ‘greater than or equal to’, ‘equal to’, ‘less than’, and ‘less than or equal to’. Moreover, each operator may be preceded by an *approximation* or a *negation* operand which alters its application. For example, compound operands may be ‘approximately greater than 1000’ or ‘not equal to 1000’. Each of these operators and a corresponding operand may be mapped to one of more words of the user’s query following the method shown, in simplified version, in Algorithm 5.2. Note that the filter creator algorithm does not determine the datatype property (or query result) to which the filter applies, as doing so would require querying the KB.

Query Filters are class instances that incorporate additional information about the filtering operation to be carried out. Namely, they include a copy of the query annotation they were inferred from, the start and end offsets of the query token subset that the filter encompasses, and a list of any POCs contained within the parse tree of the QF. Moreover, CQF instances are augmented with information about whether they apply to a datatype property of the KB (and if so, to which one) or to inferred values of graphic objects (more on this in Section 5.8.2); and, in case that the operators of the CQF include an approximation, the *tolerance* level of included filtered results, expressed as a percentage with respect to its operand.

Nominal Query Filters, on the other hand, incorporate two additional class attributes that had to be included to deal with two common filtering cases in AUDiaL. The first case

---

**Algorithm 5.2:** Cardinal Query Filter (CQF) identification

---

```

1 function get_CQFs (query);
  Input : A query in natural language query
  Output : List of CQF instances query_filters
2 query_filters  $\leftarrow$  array();
3 foreach annotation  $\in$  query do
4   operators  $\leftarrow$  array();
5   consider  $\leftarrow$  consider_PT_for_CQF(annotation);
6   if consider is TRUE then
7     pt  $\leftarrow$  annotation.parseTree();
8     label  $\leftarrow$  pt.label;
9     pre_terminals  $\leftarrow$  preTerminalsOfTree(pt);
10    negate  $\leftarrow$  FALSE;
11    operand  $\leftarrow$  NULL;
12    foreach ppt  $\in$  pre_terminals do
13      if operand is NULL then
14        operators_ppt, operand  $\leftarrow$  ops_in_PT(ppt, label);
15        operators.add(operators_ppt);
16      end
17    end
18  end
19 end
20 if operators is not empty and operand is not NULL then
21   if is_negated(annotation) then
22     | operators = negate(operators);
23   end
24   qf  $\leftarrow$  create_filter(annotation, operators, operand);
25   query_filters.append(qf);
26 end

```

---

comprises the filtering of graphic objects having a specific value as specified by a metric axis of the diagram. For example, in the (somewhat contrived) query “what Austrian state has 1,690,000 inhabitants?” performed on our example bar chart, the number ‘1,690,000’ could be interpreted as the operand of a NQF. However, this number may not appear in any of the KB’s triples, since it might have to be dynamically computed by comparing the length of a bar with its corresponding metric axis. Therefore, we account for this anomaly by creating an extra NQF that refers to an exact value associated with graphic objects taking part in a graphic relation with a meaningful space that contains a labelled axis. Lastly, filters may also refer to user-generated labels external to the KB; this feature is described in Section 6.4.

Note that during the Question Analysis phase, AUDiaL does not have access to the underlying knowledge base. Therefore, at this moment in time it is not possible yet to infer the resources

---

**Algorithm 5.3:** Helper method of Cardinal Query Filter (CQF) identification for detecting potential CQF phrases

---

```

1 function consider_PT_for_CQF (annotation);
  Input : An annotated phrase from a Parse Tree annotation
  Output : Whether the input phrase may potentially be a CQF consider
2 TOP_LABELS := ["PP", "ADJP", "QP", "VP", "FRAG"];
3 consider  $\leftarrow$  FALSE;
4 pt  $\leftarrow$  annotation.parseTree();
5 if pt.label  $\in$  TOP_LABELS then
6   | consider  $\leftarrow$  TRUE;
7 end
8 else if pt.label == "NP" then
9   | first_child_label  $\leftarrow$  getLabel(getChildren(pt)[0]);
10  | if first_child_label == "QP" then
11    |   | consider  $\leftarrow$  TRUE;
12  | end
13 end
```

---

that a Query Filter applies to and thus some of the attributes of identified Query Filters, such as the properties of the KB they refer to, are left empty for later resolution. The end result of the Question Analysis phase for an example query is partly shown in Figure 5.9. It consists on a Parsed Query object that contains POCs and CQFs to be resolved against formal resources, as well as question terms (annotations) to be searched for matching elements in the KB. The following section describes how this process takes part within the Phrase Mapping phase of the NLP to KB pipeline.

**Algorithm 5.4:** Helper method of Cardinal Query Filter (CQF) identification for detecting potential operand and operators of a CQF.

---

```

1 function ops_in_PT (pt, label);
    Input : A Parse Tree for a phrase pt
    Input : The label (grammar category) of the Parse Tree's parent PT label
    Output : List of CQF operators found in the input phrase operators
    Output : CQF operand found in the input phrase operand
2 NEG_LABELS := ["RB", "DT"];
3 COMP_LABELS := ["JJR", "RBR", "JJ", "IN", "JJS"];
4 OPERAND_LABELS := ["CD", "NN*", "VBN", "JJ"];
5 operators  $\leftarrow$  array();
6 pt_text  $\leftarrow$  getLeaves(pt);
7 if (label  $\in$  NEG_LABELS) and is_similarity_text(pt_text) then
        // Similarity operators
8     | operators.append('approximate');
9 end
10 if label  $\in$  COMP_LABELS then
        // Rest of operators
11     | operator  $\leftarrow$  parse_operator(pt_text, pt);
12     | if operator is not NULL then
13         |     | operators.append(operator);
14     | end
15     | else if label  $\in$  OPERAND_LABELS and is_quantity(pt_text) then
16         |     | operand  $\leftarrow$  pt_text;
17     | end
18 end
19 else
20     | operators.remove('approximate');
21 end
```

---

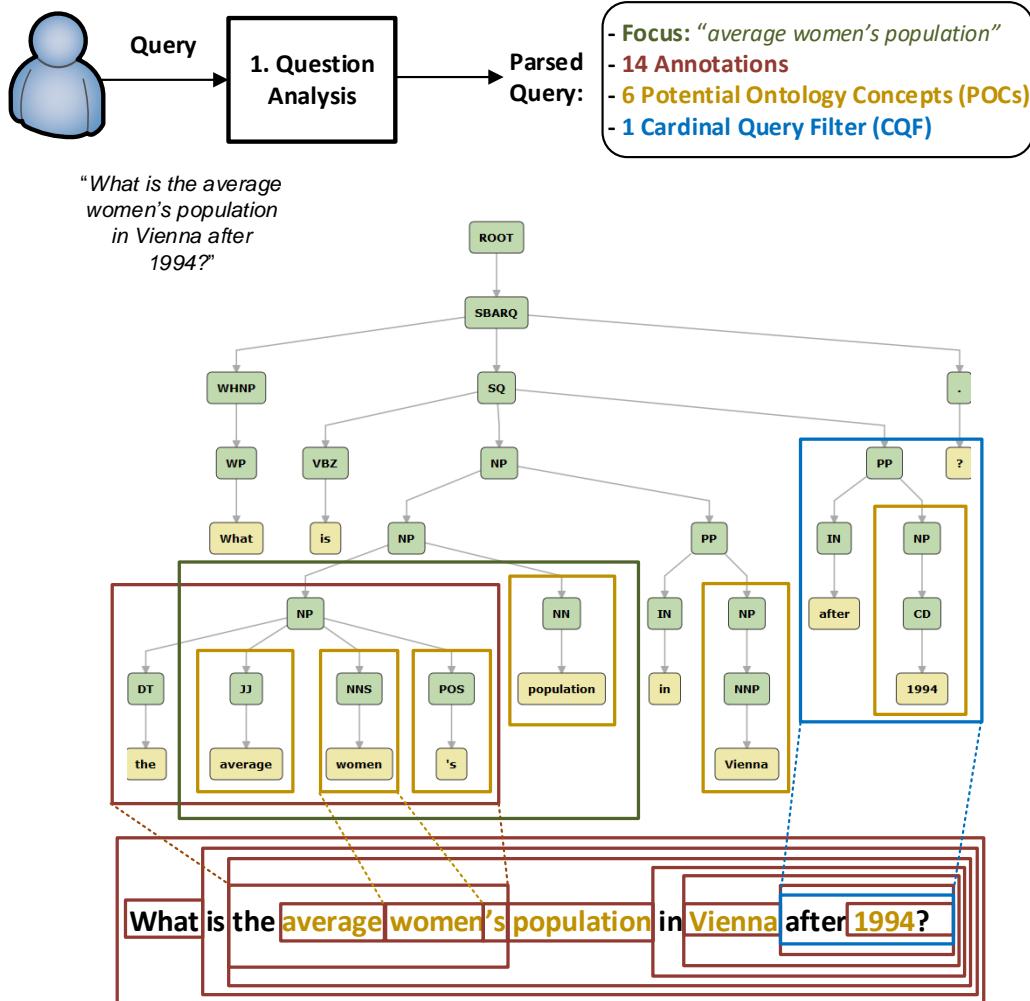


Figure 5.9: The result of the Question Analysis phase is a *Parsed Query* instance containing *Annotations* (red boxes, for clarity only one is overlaid on the PT), *Potential Ontology Concepts* (POCs; yellow boxes), *Cardinal Filter Queries* (CFQs; blue box) and additional metadata, such as the *question focus* (green box), inferred from the syntax of the input query. The knowledge base is not consulted at this stage of the NLP pipeline.

## 5.7 Phrase Mapping Phase

The main purpose of the *Phrase Mapping* phase of the NLI to KB pipeline (see Figure 5.10) is to provide the query annotations generated during the Question Analysis phase with formal semantics. This information stems from the knowledge bases embedded in a semantically-enhanced graphic or from an available supporting domain ontology (see Chapter 4 for more information). The result of the Phrase Mapping phase consists of a number of Ontology Concept (OC) instances, which underpin formal resources that match phrases, or derivations thereof, appearing in the user query. Note that POCs and CQFs are not resolved yet, as this phase is only concerned with querying the knowledge bases in the search for resources which are relevant to the user query. The Phrase Mapping phase is divided into four main sub-phases, described next.

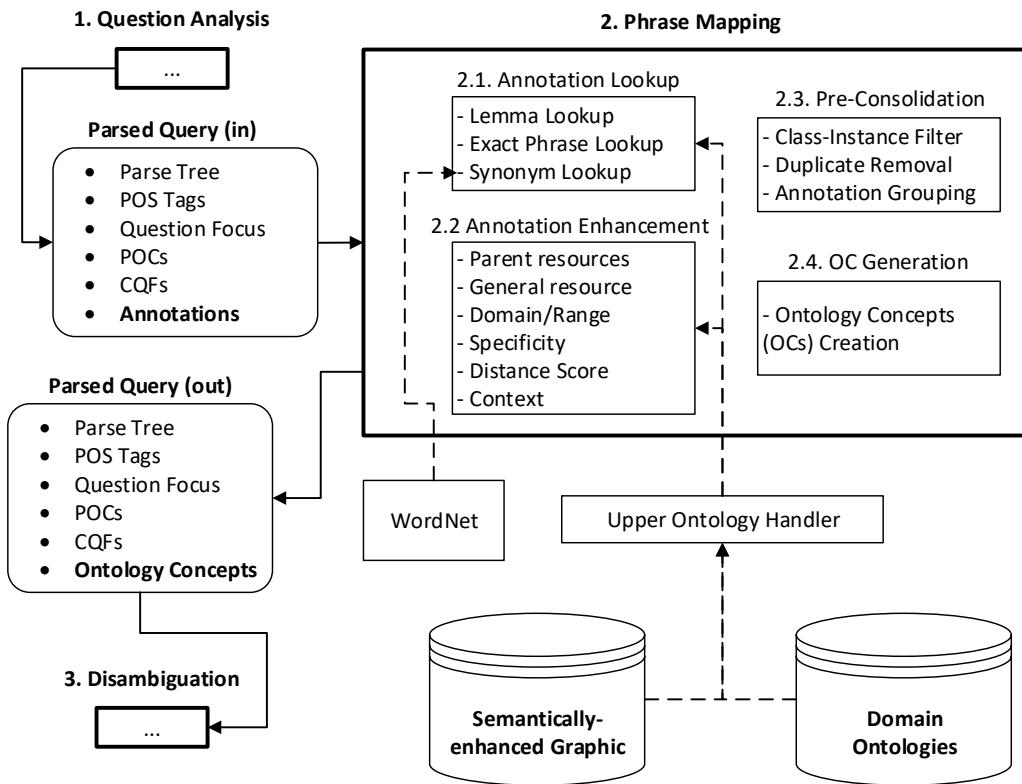


Figure 5.10: Phrase Mapping phase of AUDiaL. The Parsed Query object generated in the Question Analysis phase is enhanced with information from the Knowledge Base (KB).

### 5.7.1 Semantic Annotation Lookup

The first step in phrase mapping consists on searching in the available KBs for potential matches for each of the unresolved phrases (i.e. annotations) of the user's query detected during the Question Analysis phase (Section 5.6). This search is dependent on the particular lexicalizations used for the resources of the KB. However, a number of techniques are employed in order to increase the chances of matching disparate query phrases to their respective formal resources. Three different kinds of lookup are performed which slightly modify each phrase's lexicalizations:

- *Lemma lookup*: first, phrases are searched in the KB having previously replaced each word that was lemmatized during the Question Analysis phase with its lemma. Hence, searching of inflected forms, which are not likely to have been included in the KB, is avoided.
- *Exact lookup*. If the lemma lookup did not return any match in the KB for a given phrase, its text is then searched for as-is.
- *Synonym lookup*. For each phrase, whether the original or its lemmatized version, each sub-tree of its constituent Parse Tree (PT) is searched for in the WordNet lexical database and its most common synonyms (if any is found in a *synset*) are fetched. Each phrase's synonym is then looked up in the KB as usual.

Each phrase lexicalization produced in this sub-phase is sent to AUDiaL's *Upper Ontology Handler*, a logical layer in charge of performing low-level queries against the KB. This layer will further adapt each given string so that it has a higher chance of being found in the KB. For example, strings are converted to lowercase, camel-case, or hyphenated in different ways depending on which kind of ontological resource (class, property, etc.) is currently being searched for. The Upper Ontology Handler leverages Python's *RDFLib* package for most of its tasks. Input strings are matched against most resources of the underlying KB, including classes, instances, properties, and datatype property value literals. More specifically, input strings are compared to all KB *resource fragment identifiers* i.e. a resource's name with its namespace removed.

While the Upper Ontology Handler tries to fetch elements from the given ontologies in an efficient manner, e.g. by employing Python generators instead of iterators where possible, it has not been designed to handle very big KBs, due to the prescribed nature of the ontologies we have chosen to employ during the evaluation of the prototype in order to satisfy requirement QR3, outlined in Section 5.4. A common method for accessing large KBs consists of creating a gazetteer from the KB contents and then accessing the constructs from the gazetteer, thereby bypassing the slow direct querying of the ontology. A *gazetteer* consists of "a set of lists containing names of entities such as cities, organizations, days of the week, etc." [CMB<sup>+</sup>14]. As opposed to more formal conceptual models such as ontologies, gazetteers have no inference mechanisms defined. Consequently, while having a much reduced expressive

power, data stored in a gazetteer can be retrieved much faster by client applications than more expressive information contained in an ontology. For an example of a dynamically generated gazetteer capable of generating ontology-based entries based on given content with regards to a given KB, see the *OntoRoot* gazetteer developed by Cunningham et al. ([CMB<sup>+</sup>14], section 13.8). In addition, caching methods, SPARQL selective patterns, and other optimization mechanisms can be used to improve SPARQL query execution times on large KBs while preserving the semantics of the original query.

Each annotation without a corresponding formal resource in the KB is later discarded, as it is very unlikely that it carries any meaning necessary for the correct resolution of the user query. Even if this approach may lead to a small proportion of user queries not having the potential of being resolved due to the disposal of phrases carrying crucial semantics, in our experiments we have found that this situation happens only rarely.

### 5.7.2 Annotation Enhancement

The remaining constituent user query phrases (i.e. annotations) are next enhanced with information about their underlying formal resources that will be of use during later stages of the processing pipeline. Namely:

- *Resource type and URI*. Each annotation keeps a list of matching formal resources, whether they matched its exact text, its lemmatized version, or a synonym. Resources are stored via their full URIs and to which resource type they refer.
- *Parent resources*. Annotations underpinned by an ontology instance store the URI of the class the individual descends from. If an individual has more than one parent, the one with the highest specificity score (see below) is selected.
- *Generic resource*. A topmost, generic, resource is generated for each annotation. It is computed as follows, depending on the type of the annotation's resource:
  - Class and properties: their generic resource is their topmost ancestor class or property found in the hierarchy, other than `owl:Thing`.
  - Instances: the generic resource of an instance is the generic resource of its parent class, except for `owl:NamedIndividual`.
  - Literal: the generic resource of a literal taking part in a (subject, predicate, literal) datatype property occurrence is the generic resource of the triple's subject.
- *Property Domain and Range*. For each annotation underpinned by a property, the URIs of the entities belonging to its domain are stored. If it is an object property, the URIs of the entities belonging to its range are fetched, too.
- *Literal Context*. If the annotation matches a literal e.g. '1994', it is augmented with all the triples of the KB in which said literal takes part as their object.

- *Specificity Score (spec)*. If available, the specificity score for a class or property is added to any annotation underpinned by it. This score is based on the *specificity distance*, an integer representing how many generations of parent elements above the resource exist. For example, given the class hierarchy `owl:Thing`  $\leftarrow$  `uvo:Graphic_Thing`  $\leftarrow$  `ddo:Informational_Role`, the specificity distance of `ddo:Informational_Role` is 2. If the resource's ancestor tree branches out into several lineages, the longest path is considered. The specificity score for each class or property is then calculated by dividing its specificity distance by the maximum class or property specificity distance found in the KB.
- *Distance score (dis)*. Annotations underpinned by a property are augmented with its distance score. The distance score of a property is defined as the average of the specificity scores of the classes in its domain and range. This score is similar to the distance score in that properties with a higher distance score can be considered more informative than those with a smaller distance score, as they associate resources with a higher specificity. The distance score,  $dis(p)$ , of a property  $p$  is computed as follows:

$$dis(p) = \frac{1}{N_{domain}(p) + N_{range}(p)} \left( \sum_{c_d \in domain(p)} spec(c_d) + \sum_{c_r \in range(p)} spec(c_r) \right)$$

### 5.7.3 Automatic Pre-consolidation

The pre-consolidation sub-phase aims to dispose of some redundant POCs and annotations in a quick and automatic manner, this is, without user feedback. The first step consists of filtering out annotations underpinned by a class of the KB which appear right next to an annotation underpinned by an instance of said class. ‘Right next to’ means that the start and end offsets of either annotation differ by exactly one. For example, in the sentence “the Nile river”, we would remove the annotation that characterizes ‘river’, as ‘Nile’ refers to a specific instance of the `River` class, and as such ‘river’ provides only redundant information not necessary for the resolution of the input query.

For each annotation removed during the pre-consolidation phase, each matching POC is removed from the Parsed Query instance as well. A POC matches an annotation if their normalized texts and offsets are the same. Next, annotations are sorted according to their offsets, in which an *overlapping annotation* i.e. an annotation that contains other annotations within its offsets, is considered to appear before each of its overlapped annotations. Lastly, as a pre-processing step before generating Ontology Concepts (OCs), annotations are brought together into groups of overlapping annotations. Each annotation strictly contained (according to their offsets in the input query) in another annotation is grouped underneath it in a hierarchical fashion. This process results in a list of nested annotation lists that will enable, during the next phase, to hierarchically structure the elicited OCs. This hierarchy of OCs will be employed during the Automatic Consolidation sub-phase of the broader Disambiguation phase (c.f. Section 5.8.1).

### 5.7.4 Ontology Concept Generation

The Ontology Concept (OC) generation step is in charge of converting each annotation with associated formal underpinning in the KB into a type-aware object, and disposing of those annotations without a corresponding formal resource. Given that each annotation has been previously augmented with relevant information during the Annotation Enhancement phase (Section 5.7.2), generating Ontology Concepts from annotations is a straightforward process, in which an instance of a corresponding sub-class of `OntologyElement` is created and its attributes initialized with values from the annotation. Lastly, all `OntologyElements` with the same underlying normalized text are grouped together for their later resolution. Moreover, all generated instances of the `OntologyInstanceElement` class, which represent annotations with an associated individual in the KB, are grouped into a single `OntologyInstanceElement` instance if they are individuals of the same class. The original annotation instance is attached to its corresponding `OntologyElement` instance so that its metadata, computed during the previous phases, is now associated to the Ontology Concept that stems from it.

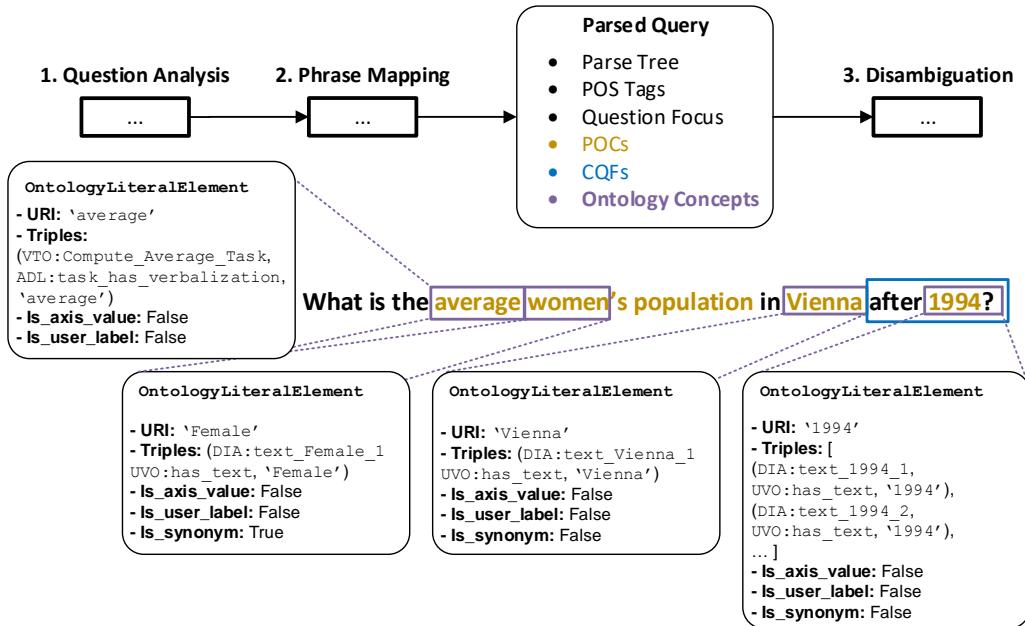


Figure 5.11: Example Ontology Concept (OC) instances generated during the Phrase Mapping phase of AUDiaL’s NLP pipeline.

Figure 5.11 displays the Ontology Concept (OC) instances generated after an input query has gone through the full Phrase Mapping phase. In this example, all formal resources retrieved are instances of the `OntologyLiteralElement` class, as they are underpinned by literals i.e. objects in datatype property occurrence triples. This is commonly the case in user

queries that only mention resources from a semantically-enhanced diagram (as opposed to a supporting domain ontology), since instance names are generally non-descriptive (e.g. they are automatically generated by SAI), and, together with class/property names, are employed for structural rather than descriptive purposes, which results in end users rarely enquiring about them. Section 5.9 discusses the two approaches to query resolution considered by AUDiaL, which depend on the kind of resources present in a fully resolved query.

The OCs generated in the example shown in Figure 5.11 also show new additional metadata being ascribed to `OntologyLiteralElement` instances. The first of them, `is_axis_value`, distinguishes literal numbers that do not explicitly appear in the KB but that can be inferred from the implicit value of a position in a metric space labelled by a metric axis. This process is described in the description of the Disambiguation phase of the natural language to KB pipeline (Section 5.8). The second one, `is_user_label`, characterizes string literals that also may not be present in the KB, but that correspond to user-input labels of a graphic object instance. User labels are described in Chapter 6. Lastly, note how if a synonym of an annotation matches a resource in the KB, a corresponding OC gets created. This mechanism aims to reduce user effort during the Disambiguation phase, but may result in incorrect phrase resolutions. In order to minimize the chance of wrong phrase resolutions, the synonym finder module only considers the most relevant WordNet synset for each given phrase.

The example in Figure 5.11 additionally introduces a datatype property incorporated by AUDiaL into the topmost level of the hierarchy of ontologies for visualization (the User/System level), as described in Section 3.10. Namely, the `adl:task_has_verbalization` datatype property (where `adl` is AUDiaL's associated namespace) characterizes natural language phrases in which users may refer to a particular VTO task, allowing for task requests expressed in natural language to be mapped to formal VTO resources during the Phrase Mapping phase. In the example above, the `vto:Compute_Average_Task` may be referred to by calling for an 'average' in the user's input query. Note that the presence in the KB of literals referring to task verbalization may result in ambiguities with other formal resources therein e.g. if there was a label in the diagram with text 'average' perhaps labelling a metric bar instance (an example of such ambiguity is shown in Figure 5.14). These ambiguities would have to be resolved during the manual Disambiguation phase discussed in next section.

At this moment in time, the Parsed Query object contains the following attributes that will have to be processed to compute an answer, see Figure 5.13. Namely:

- A list of Potential Ontology Concepts (POCs): phrases of the input query which have the potential of being linked to formal resources.
- A list of Cardinal Query Filters (CQFs): phrases in the input query that may modify the selection of KB triples before executing a task or the triples that make out its answer.
- A list of Ontology Concepts (OCs): phrases of the input query for which a matching resource has been found in the KB. Note that a single phrase may be associated with more than one Ontology Concept e.g. if a Literal and an Instance with the same name exist in the KB.

Following the generation of Ontology Concepts, there is a high probability that there exist ambiguous or unresolved elements in the Parsed Query object. The following phase in the natural language to KB pipeline, namely the Disambiguation phase, is concerned with cleaning up these ambiguities so that the query can be resolved and its answer computed.

## 5.8 Disambiguation Phase

The purpose of the *Disambiguation phase*, shown in Figure 5.12, is to resolve the ambiguities that may have arisen during any of the previous phases of the natural language to KB pipeline. A Parsed Query object may incorporate ambiguities in any of the following elements:

- Potential Ontology Concepts (POCs) are ambiguous by definition, as they are unresolved phrases that may contain necessary semantics for the resolution of the query.
- Cardinal Query Filters (CQFs) may refer to either a datatype property literal (e.g. numerical labels in a diagram), or to the implicit value of a position in a metric space, labelled by a metric axis, in which a graphic object lies (e.g. the numerical value carried by a bar in a bar chart according to some axis).
- Ontology Concepts (OCs) present ambiguities when two or more OC instances refer to the same phrase in the user's query.

Where possible, these ambiguities are resolved automatically by detecting redundant elements in the Parsed Query object or by AUDiaL's learning mechanism, described in Section 5.8.4, which infers new disambiguation rules as a result of the collective intelligence of the system's users. However, at times users will be requested to explicitly teach the system how certain ambiguities ought to be resolved via accessible feedback and clarification dialogues following the approach of FREyA [DACP13], which we have adopted and expanded to fit AUDiaL's needs.

### 5.8.1 Automatic Consolidation

The *Automatic Consolidation* sub-phase aims to resolve as many ambiguities of the Parsed Query object as possible in an automatic fashion by performing a joint analysis of its POCs, QFs, and OCs. The following steps are undertaken, see Figure 5.12.

- Removal of CQFs contained, by operator, within another CQF. Some CQF operators are derived from a combination of two atomic operators e.g. GEQ ( $\geq$ ) and NEQ ( $\neq$ ). In this case, any CQF contained (by offset, see below) therein, whose operator is one of the atomic constituent operators of the container CQF, is automatically removed.
- Deletion of CQFs containing, by offset, other CQFs ( $QCF_i \supseteq QCF_j$ ). If the tokens that make up a CQF are contained within the span of tokens covered by another CQF,

the container CQF is removed. For example, given the query “what is the average population after 1994 in regions with more than 100,000 inhabitants?”, only two filters should remain. Namely, those corresponding to the phrases ‘after 1994’ and ‘more than 100,000 inhabitants’. This rule aims to ensure that no nonsensical super-filters remain in the consolidated Parsed Query object.

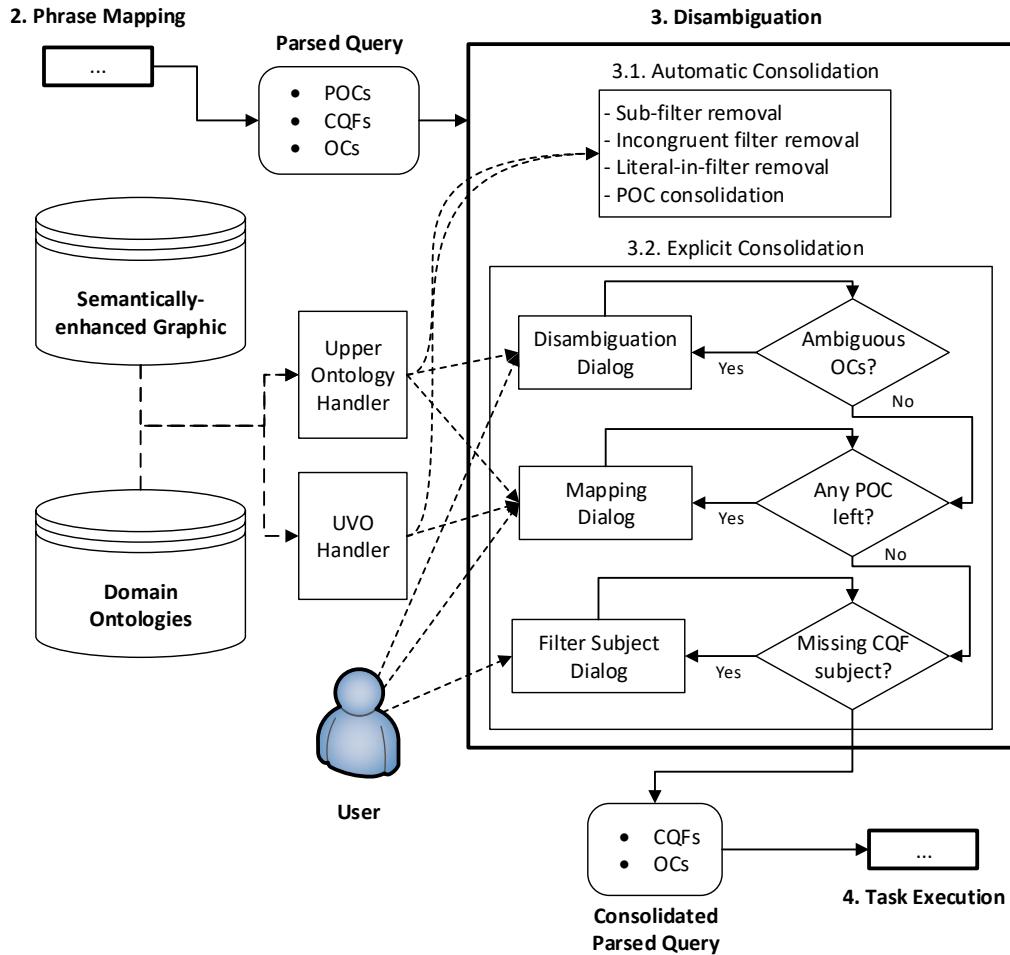


Figure 5.12: Disambiguation phase of AUDiaL. Any ambiguities and unresolved terms in the Parsed Query are settled by a combination of automatic and user feedback mechanisms. It results on a consolidated Parsed Query object that can be resolved into a task to be executed during the Task Execution phase.

- Removal of POCs contained within an OC. A POC is considered to be contained in an OC in the following two cases:
  - Its start offset is greater than or equal to the OC's start offset, and its end offset is

lower than or equal to the OC's end offset ( $POC \subseteq OC$ ). In the example shown in Figure 5.11, the POC corresponding to the word 'Vienna' would be automatically removed since it matches an `OntologyLiteralElement` instance.

- The POC, after having certain phrases removed, such as *wh-adjective* phrases, personal and possessive pronouns, and determiners, is contained within the OC following the rule above. For example, as shown in Figure 5.11, the POC corresponding to 'the average' is to be discarded, since, after having had its determinant ('the') removed, matches an OC ('average').
- Discarding of POCs contained within a CQF ( $POC \subseteq QCF$ ), following the same rules as in the removal of POCs contained within an OC outlined above.
- Deletion of OCs contained by offset within CQFs ( $OC \subseteq QCF$ ). In the example shown in Figure 5.11, the `OntologyLiteralElement` instance corresponding to the token '1994' would be discarded, since it is contained within a CQF instance.
- Removal of incongruent CQFs in consonance with the KB. Each generated CQF is inspected against the potential datatype properties or metric axes it may refer to. If the operation expressed by the CQF falls beyond the range of any datatype property or the range of values covered by any metric axis, the CQF is discarded. For example, consider the following input query, performed against the semantically-enhanced bar chart shown in Figure 5.5: "which states have more than 3,000,000 inhabitants?". The chart's metric axis has an approximate range of [0, 2000000], and the remaining numeric literals in the graphic are '1994', '2004', and '2014', all of which are lower than 3,000,000. Therefore, the CQF corresponding to the phrase "more than 3,000,000" is not consistent with the information expressed by the chart, and as such, it would be removed from the Parsed Query object. This case would moreover result on an immediate resolution of the query, in which all further processing would be skipped and a negative answer e.g. "no bars match your query" would be returned to the user.

After automatic consolidation is completed, a Parsed Query object will likely have less POCs and redundant elements. Hopefully, by now there will be no POCs left and every single OC and CQF will refer to a single formal resource in the KB. In this case, the query is ready to be resolved, and the Task Execution phase (Section 5.9) immediately takes place. However, the Parsed Query may still present ambiguous elements and POCs that need to be mapped to formal resources that could not be automatically settled. AUDiaL will in such case resort to explicitly enquiring the user about the best course of action. This process is carried out by the explicit consolidation phase, discussed next.

The example in Figure 5.13 shows the remaining elements in the Parsed Query of our example input query after having gone through the Automatic Consolidation sub-phase. POCs have gone from six to two instances, OCs from four to three instances, and CQFs have remained unchanged. Note that this is a rather simple example that has been constructed for didactic purposes. POCs and OCs may be generated not only from single words, but also from longer phrases, as well as from verbs, adverbs, and other tokens associated to diverse PoS tags.

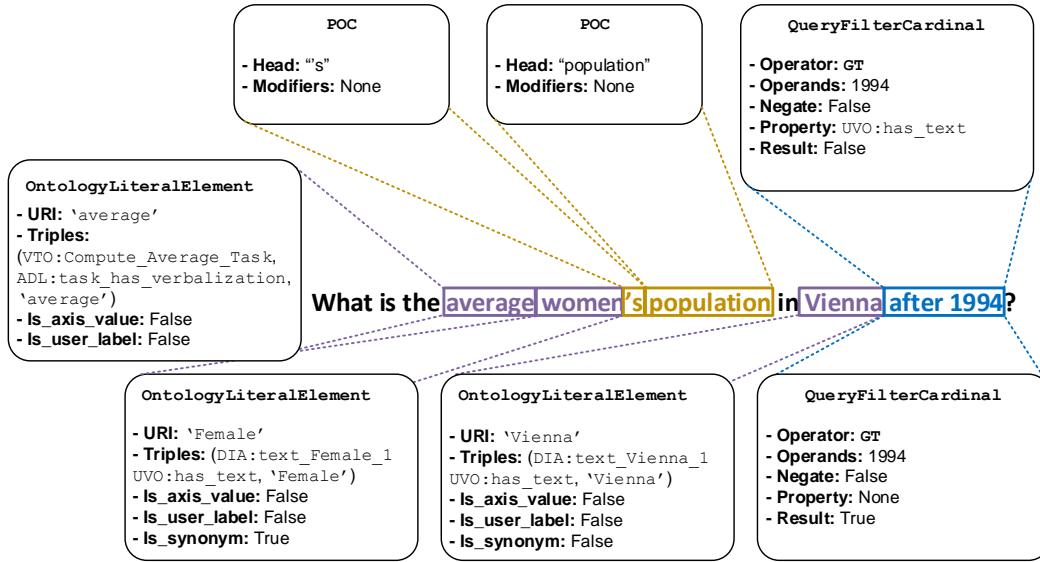


Figure 5.13: Elements (POCs, OCs, and CQFs) of the Parsed Query remaining after automatic consolidation.

### 5.8.2 Explicit Consolidation

The *Explicit Consolidation* sub-phase requests the user for manual resolution of any ambiguous or unknown terms remaining in the Parsed Query object after the automatic consolidation phase has failed to resolve them. Showing the system’s interpretation of an input query to the user in an understandable format is known as *feedback*, and has been shown to increase the users’ confidence in using natural language systems and to enable users to use them more effectively [DACP13]. The main components that make up AUDial’s explicit consolidation mechanism are displayed in Figure 5.18. As shown in Figure 5.12, explicit feedback is given to the user in the form of three different sub-dialog types, each of which can be observed zero or more times until each unresolved element has been settled. First, the remaining POCs are each mapped to zero or one formal resource in the KB. Then, the subject of any uncertain CQF is decided. Lastly, ambiguous OCs are resolved by selecting the correct one among them. Let us now describe each of these steps in detail.

#### Disambiguation Dialogs

The first type of dialog that may take place between the system and the user is known as the *disambiguation dialog*. Here, the user is tasked with disambiguating between identified OCs during the Ontology Concept Generation phase (Section 5.7.4) that refer to the same span of tokens in the user’s query. In our example query no disambiguation dialogue would take place, as it contains no ambiguous OCs. However, let us suppose that the bar chart

shown in Figure 5.5 had an additional stacked bar labelled ‘average’ which depicted the average population in Austria divided into male and female sub-populations. In this case, the consolidated query would have two OCs assigned to the unigram ‘average’, as Figure 5.14 illustrates. The user would be subsequently enquired to resolve the ambiguity between the two OCs by selecting the right one in an accessible dialog embedded in AUDiaL’s interface, also shown in Figure 5.14. More specifically, the process that generates a disambiguation dialog in AUDiaL operates as follows:

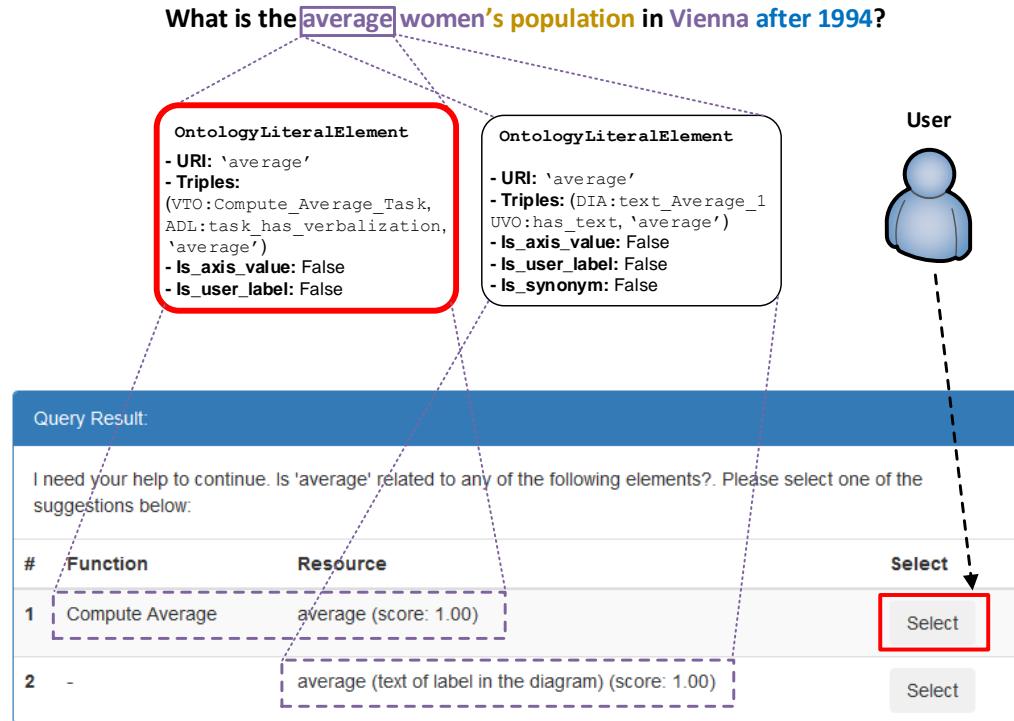


Figure 5.14: A disambiguation dialog. Two ambiguous OCs are present in the user’s query (top). The user chooses the Literal OC corresponding to a VTO task (red) instead of the Literal OC corresponding to a textual label by clicking on the relevant row of the accessible dialog window (bottom).

1. The list of ambiguous OCs present in the user’s query is considered. While any ambiguous OCs remain in the Parsed Query, the set of ambiguous OCs referring to the phrase that lays closest to the question focus is chosen (the distance between two n-grams of the question is calculated by navigating its Parse Tree (PT) and counting the number of nodes in between the root nodes of each n-gram).
2. A list of dialog suggestions is generated based on the ambiguous OCs selected. If

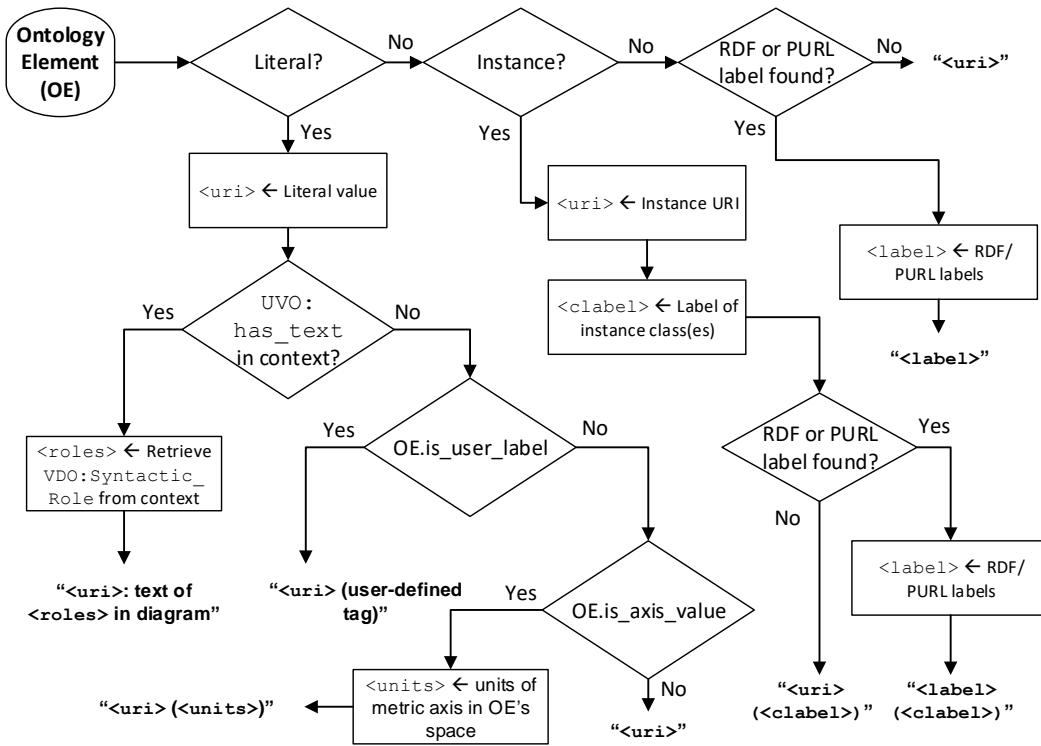


Figure 5.15: Approximate workflow for the generation of the label of an Ontology Concept (OC) to be displayed in a user dialog.

similar disambiguation dialogs had previously been resolved in the system (by any user), the scores for each suggestion are updated accordingly (AUDial’s learning subsystem is described in Section 5.8.4). If the difference in scores is big enough, the ambiguity between OCs may be resolved automatically and no user interaction is necessary. Otherwise, each ambiguous OC is listed on the dialog. For each OC, the following information is given:

- *Function*: the VTO task that the OC refers to, if any; or an operation to be undertaken on the resulting triples of a domain ontology: ‘max’, ‘min’, ‘sum’, or ‘average’.
  - *Resource*: a human-friendly label of an OC’s underlying formal resource, according to the workflow depicted in Figure 5.15.
  - *Score*: the current learning vote score of the element. Vote scores are computed during learning as described in Section 5.8.3 and Section 5.8.4. This score increases with the amount of users that choose an OC in disambiguation dialogs in a given context, given higher priority to, or even automatically resolving, those OCs that are preferred by most users when resolving similar queries.

3. The user selects one of the suggested OCs as the correct one. Every other ambiguous OC is removed from the Parsed Query object, and corresponding voting scores for each suggestion are computed and stored.
4. If any ambiguous OCs remain in the Parsed Query, step 1 is repeated again. Otherwise, the system checks for the presence of POCs that need resolving. This process is described next.

### Mapping Dialogs

After all OCs in the query have been disambiguated, the user may have to engage in a number of *mapping dialogs*. These dialogs ask the user to map each of the unresolved POCs in the Parsed Query object to a formal resource in the KB (or to none of them). The end result of a series of mapping dialogs is the removal of all POCs from the Parsed Query. In addition, as a result of resolving each mapping dialog, a new OC instance may be added to the Parsed Query object, if the user chooses to map its POC to some resource in the underlying KB.

In our example query, as shown in Figure 5.13, there are two POCs that need to be manually resolved. Namely, the possessive ending ‘s’ and the noun ‘population’. These two phrases were detected as potential carriers of important semantics for the resolution of the input query by the POC identification method (c.f. Algorithm 5.1), but could not be automatically mapped to KB resources during the phrase mapping phase (Section 5.7). Consequently, as shown in Figure 5.16, the user is tasked with mapping each of these phrases to a formal resource in the KB (or to none, in which case the corresponding POC will be ignored). Note that the mapping dialog shown in Figure 5.16 displays the case in which the word ‘Wien’ had been input by the user instead of ‘Vienna’. Since no resource named ‘Wien’ exists in the ontology, a POC would be created, which should in turn be manually mapped by the user to the ‘Vienna’ Literal OC instance shown in Figure 5.13.

Given that the vast majority of ontologies that make up the KB will likely contain too many resources to be displayed in a mapping dialog, a subset thereof needs to be selected that can be processed by users in a manageable, efficient manner. This restriction is even more so crucial if we take into account that most users of AUDiaL will be navigating its user interface in an almost sequential manner with the use of screen readers. A ranking mechanism of KB resources in relation to the ambiguous POC phrase was thus developed so that only the topmost  $n$  resources most similar to the POC are retrieved from the KB and displayed in the mapping dialog. During our experiments, we found  $n = 20$  to be a good compromise between a high likelihood of obtaining the most relevant resource without overwhelming users with an excessive amount of choices, but this number can be adjusted in AUDiaL’s settings. The ranking mechanism for suggestions implemented in AUDiaL will be described in next section.

Suggestions for resources to be mapped to a given POC are generated following a similar method to that of FREyA [DACP13], that has been adjusted for the most common use case of AUDiaL, namely, accessing diagram data embedded in a semantically-enhanced graphic. For each POC in the Parsed Query object, mapping suggestions are created as follows:

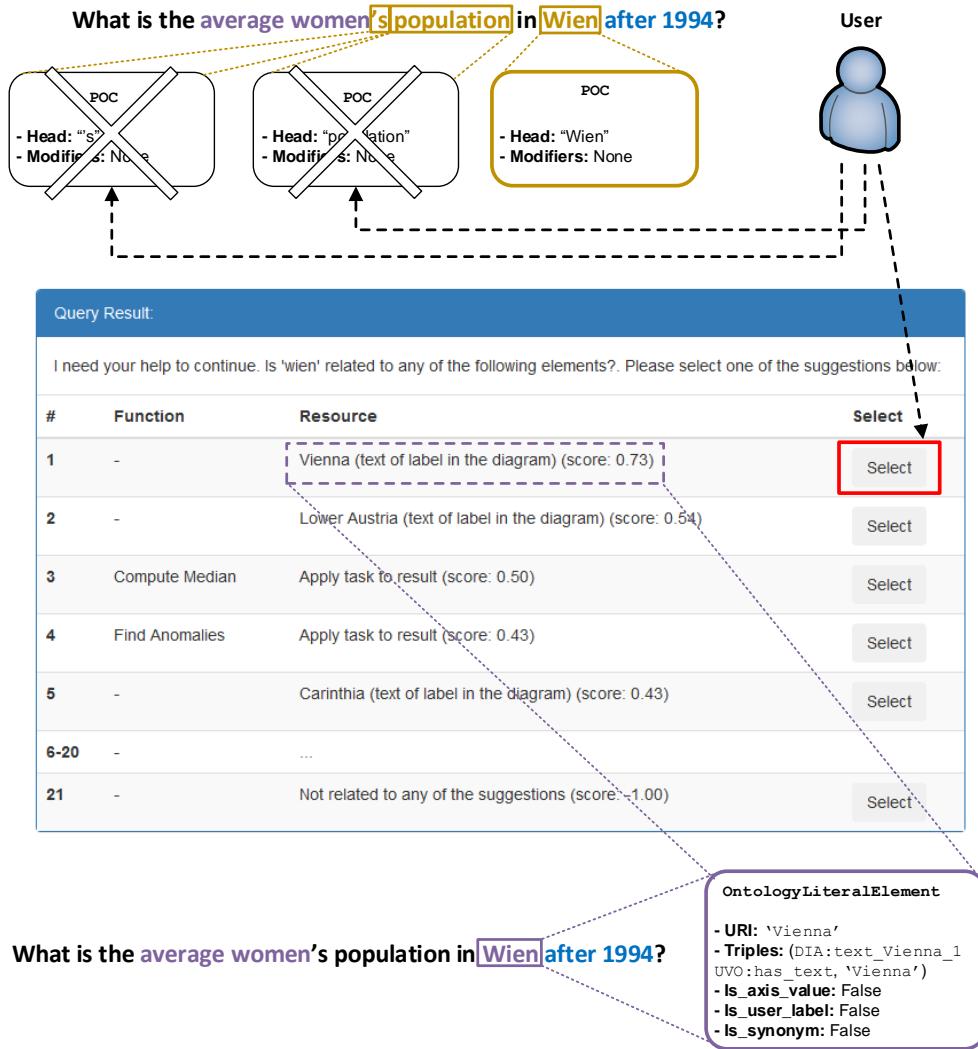


Figure 5.16: Example of an (adapted) mapping dialogue. The user should select the resource on the 1st row (the literal “Vienna”) as the resource that the POC corresponding to the phrase “Wien” should be mapped to, resulting in the creation of a new Literal OC. The remaining two POCs were previously discarded by mapping them to “None” elements (last row of mapping dialog). This results in a Parsed Query without POCs.

1. The POC instance that lays closest to the focus of the input query is selected, in a corresponding manner to how OCs are selected in disambiguation dialogs, see previous section.
2. The OC (or OCs) of the Parsed Query object that are closest to the given POC instance are retrieved. The distance between a POC and a OC is given by the shortest path

length between their parse sub-trees, see [Hir06] for a formal definition of distance between trees.

3. For each OC retrieved in the previous step, its generic OC is fetched from its corresponding annotation. Recall that generic OCs had been previously computed during the Annotation Enhancement step of the Phrase Mapping phase (Section 5.7.2). This generic OC, which we call the ‘context’ of the POC, will be later employed by the learning mechanism of AUDiaL, described in Section 5.8.4. If the Parsed Query object has no OCs, the context of the POC is set to none.
4. Formal resources are fetched from the KB that will be suggested to the user to be mapped to the unresolved POC. The resources fetched will vary depending on whether a domain ontology is present in the KB.
  - a) If no domain ontology is available, only certain resources from the closed-domain visualization ontologies described in Chapter 3 are considered. More specifically, priority is given to votes whose candidate resource is a subject of the `uvo:has_text` datatype property (i.e. all text present in the diagram) and to verbalizations of tasks in VTO, unless a task has been inferred from the input query already automatically or via a disambiguation dialog. Note that, for each task, only the most similar (according to similarity score, see next section) verbalization to the unresolved phrase is considered. An example of this case is shown in Figure 5.16. It can be seen how mapping suggestion resources are selected from diagram labels and visualization tasks according to their aggregated voting score. This score and how it is computed will be explained in next section. In addition, numerical POCs may be mapped to values in a metric axis of the diagram.
  - b) If one or more domain ontologies are present in the KB, suggestions are fetched from them following the ontology reasoning approach of FREyA [DACP13]. This is a domain-agnostic approach that allows AUDiaL to move from the closed-domain constraints of the visualization ontologies described in Chapter 3 to the open-domain scenario of querying domain ontologies from heterogeneous sources. This ontology reasoning is briefly described as follows [Dam11]: “[Mapping suggestions] are found [...] by looking at the closest Ontology Concepts to the POC. [...] For the closest OC  $X$ , we identify its neighboring concepts which are shown to the user as suggestions. Neighboring concepts include the defined properties for  $X$ , and also its neighboring classes. Neighboring classes of class  $X$  are those that are defined to be 1) the domain of the property  $P$  where  $\text{range}(P) = X$ , and 2) the range of the property  $P$  where  $\text{domain}(P) = X$ ”. At the moment no ontology matching mechanisms are employed to assign domain knowledge to the visualization resources of a semantically-enhanced graphic in AUDiaL.

After the user has selected one of the suggestions of a mapping dialog (including the ‘None’ option), its corresponding POC gets mapped to zero or one OC from the KB, and query resolution continues. Moreover, AUDiaL learns from this interaction with the user, and the

selected OC gets assigned higher priority in future similar mapping dialogs with users. In some cases, the system may learn to map certain POCs in context to a OC in an automatic fashion without involving the user (more information is given in Section 5.8.1).

### Filter Clarification Dialogs

Lastly in the Disambiguation Phase, the system will ask the user to resolve any remaining ambiguous Cardinal Query Filters (CQFs). Note that CQFs are not generated if the user is querying a domain ontology i.e. they are only considered by AUDiaL if the user is interacting with a semantically-enhanced graphic underpinned by the ontologies described in Chapter 3. For each ambiguous CQF, the following mapping possibilities are offered to the user in a so-called *filter clarification dialog* (Figure 5.17):

The screenshot shows a 'Query Result' dialog with a blue header. The main content area contains the following text: 'I need your help to continue. Is 'after 1994' related to any of the following elements?. Please select one of the suggestions below:'. Below this, there is a table with three rows:

#	Function	Resource	Select
1	return those greater than 1994	Diagram Labels (score: 0.24)	<input type="button" value="Select"/>
2	return those greater than 1994	Query result: 'number of people' (score: 0.00)	<input type="button" value="Select"/>
3	-	Not related to any of the suggestions (score: -1.10)	<input type="button" value="Select"/>

Figure 5.17: A filter clarification dialog for the input query “what is the average women’s population in Vienna after 1994?”. The user must choose the first suggestion so that the correct CQF is generated.

1. The CQF refers to a filtering of values with respect to a metric axis. In this case, the measurement unit of the corresponding axis is fetched from the KB according to its `ddo:expresses_cardinal_information` object property. If such information is not present in the KB, the most likely unit of measurement is determined through the head of the question’s focus (e.g. ‘women’s population’ in the query shown in Figure 5.17). The inferred measurement unit is then used as part of the suggestion’s label in the dialog, as also depicted in Figure 5.17.
2. The CQF is concerned with the values of some numerical label present in the diagram. In this case, all graphic objects in the diagram taking part in `vdo:is_labeled_by` property occurrences will be filtered so that triple objects match the filtering operation described by the CQF

3. The CQF has been incorrectly inferred, in which case the user must select the ‘Not related to any of the suggestions’ option so that the CQF will be discarded.

### 5.8.3 Ranking Suggestions

Any dialog between AUDiaL and the user consists of a list of suggested resources from the KB that the user may choose to map to unresolved or ambiguous phrases from the input query. In most cases, the total number of fetched suggestions would be too high to be displayed at once in a single dialog without overwhelming the user. For example, as described in previous section, in a mapping dialog all VTO tasks and text contained in a semantically-enhanced graphic are considered as potential suggestions for any POC in the user’s query. Therefore, a ranking mechanism is necessary so that only the resources that the user is most likely referring to are displayed in the dialog. AUDiaL’s ranking mechanism is based on a *similarity score (sim\_score)* between the normalized text of the POC, OC, or CQF and the RDF/PURL label (if available) or the name of a suggested resource from the KB as given by its IRI. The similarity score between two strings is calculated as a linear combination of three sub-scores, namely:

1. *Distance similarity* as given by the Monge-Elkan distance [JBGG09] between both strings. The generalized Monge-Elkan distance is a simple but effective method for measuring the similarity between two text strings based on an internal character-based similarity measure combined with a word-level similarity measure. We chose the Levenshtein distance [LL07] as the internal similarity measure i.e. the minimum number of editing operations (character insertion, deletion, and substitution) to transform one token into another. The distance similarity score *sim\_dist* between strings *a* and *b* is thus computed as follows:

$$sim\_dist(a, b) = \sqrt{\frac{1}{a} \sum_{i=1}^{|a|} \left( \max \{lev\_dist(a_i, b_j)\}_{j=1}^{|b|} \right)^2}$$

2. *Synonym similarity (syn\_dist)*. Up to three synonyms from the WordNet corpus are retrieved for both terms being compared. Then, the distance similarity between each term and the other term’s synonyms is computed and averaged. If no synonyms are available, the synonym similarity for that pair of terms is zero.
3. *Soundex similarity (soun\_dist)*. Soundex [OR18] is a phonetic algorithm that indexes words by generating alternative strings for the input words according to their English pronunciation. For example, homophones such as ‘leak’ and ‘leek’ yield the same or very similar strings (e.g. ‘L200’). Likewise, words with similar phonetics yield similar strings e.g. the Soundex string for ‘seek’ is ‘S200’. The Soundex distance between two phrases is computed by calculating a meaningful distance between the Soundex strings for both phrases. In AUDiaL, the Jaro-Winkler distance [Wilb] is employed.

Using these three similarity measures between both phrases, the final similarity score is given by:

$$\text{sim\_score}(a, b) = w_1 \cdot \text{sim\_dist}(a, b) + w_2 \cdot \text{syn\_dist}(a, b) + w_3 \cdot \text{soun\_dist}(a, b) \quad (5.1)$$

Where the weights  $w_1$ ,  $w_2$ , and  $w_3$  must sum up to 1 and can be manually adjusted in AUDiaL's settings. By default, we set  $w_1 = 0.45$ ,  $w_2 = 0.4$ , and  $w_3 = 0.15$ . The formula above gives the initial similarity score (also known as *numeric vote*) for a pair of phrases. However, this number may be altered in two different ways in order to give priority to certain suggestions: (1) In mapping dialogs, if no domain ontology is available, the scores for all textual resources of a semantically-enhanced graphic, i.e. subjects of `uvo:has_text` datatype property occurrences are given a small numeric boost as they are more likely to be chosen by the user compared to VTO task verbalizations; and (2) via a learning model that can learn an appropriate ranking of suggestions in context from the interaction between AUDiaL and its users. This learning mechanism will be explained next.

#### 5.8.4 Collective Learning

AUDiaL includes a learning subsystem (see Figure 5.18) that aims to improve the efficiency of interactions between untrained users and the NLI as the system acquires new knowledge by answering similar queries asked by the same or different users, this is, a mechanism that aims to improve the system's *habitability* in a collaborative, semi-automatic fashion.

First, simple Reinforcement Learning (RL) is employed to update the numeric votes for the suggested OCs in a dialog i.e. to learn a proper ranking of candidates in context. The selected candidate (an OC represented by the `Vote` class) by the user obtains a positive reward (+1.0 by default), whereas every other suggestion in the dialog receives a negative reward (-1.0 by default). Rewards are added to each `Vote` instance's numeric vote. The default values may be altered in AUDiaL's configuration files in order to fine-tune learning. For example, after resolving the mapping dialog shown in Figure 5.16, the 'Vienna' suggestion (i.e. its corresponding `Vote` instance which in turn is linked to a specific `OntologyLiteralElement` instance) would have a numeric vote of 1.73 (from the initial similarity score of 0.73 between the strings 'Vienna' and 'Wien' plus the default positive reward of 1), whereas every other suggestion would see its numeric vote reduced by 1 (negative reward). Note that the newly computed numeric votes would apply to future similar dialogs performed by any user, not only the user who initially resolved the dialog. For instance, if a different user inputted the exact same query again, the 'Vienna' suggestion would be retrieved with a default score of 1.73. If the user now voted for this option once more, the suggestion would be stored with a voting score of 2.73, and so on.

In addition to this, in order for the learning mechanism to function, the newly computed numeric votes need to be persisted, as follows. Each time any user resolves a disambiguation or mapping dialog, the ambiguous input phrase (`LearningKey`), its context (`GenericElement`), and the positively and negatively voted OCs (`LearningVote`) for the phrase are stored in AUDiaL's memory. A `LearningKey`'s context consists of a *generic*

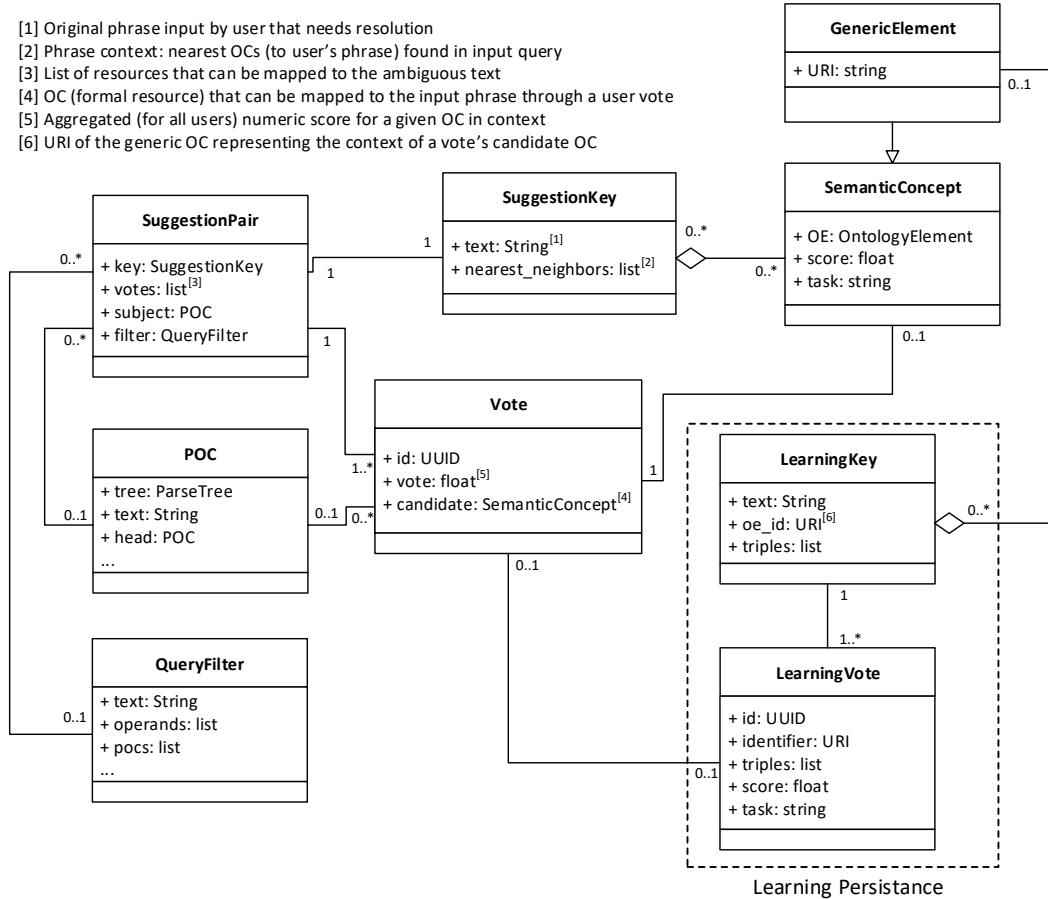


Figure 5.18: UML diagram of AUDiaL's disambiguation and learning model.

resource as given by the *annotation* of the *nearest OC* to the ambiguous input phrase. Let us unpack this sentence:

1. The closest OC to the input phrase being resolved (according to the distance between their sub-parse trees in the full input query [Hir06]) is retrieved. If the input query contains no OCs, then a special 'None' context element is created and the next step is skipped.
2. The generic resource for the context OC retrieved in (1) is retrieved from the OC's annotation. The process for creating generic resources for OCs was already described in Section 5.7.2.

The generic resource created in step (2) above is interpreted as *context* for the input phrase being resolved. In this manner, suggestions for ambiguous phrases are persisted

and fetched taking into consideration a broader semantic context, which enables unseen input queries to be treated in a similar fashion to previously resolved similar queries during the disambiguation phase of the natural language to KB pipeline. Figure 5.19 displays the workflow for learning a mapping of a phrase in context. Each time a dialog is settled, a LearningKey instance is created from the ambiguous input phrase and its query context. The LearningKey is then associated to a number of LearningVote instances, each one corresponding to a suggestion in the dialog with associated voting score after dialog resolution. These instances are lightweight versions (instance copies without foreign members) of their associated SuggestionKey and Vote instances that can be easily serialized to JSON format and persisted to a file system for later retrieval.

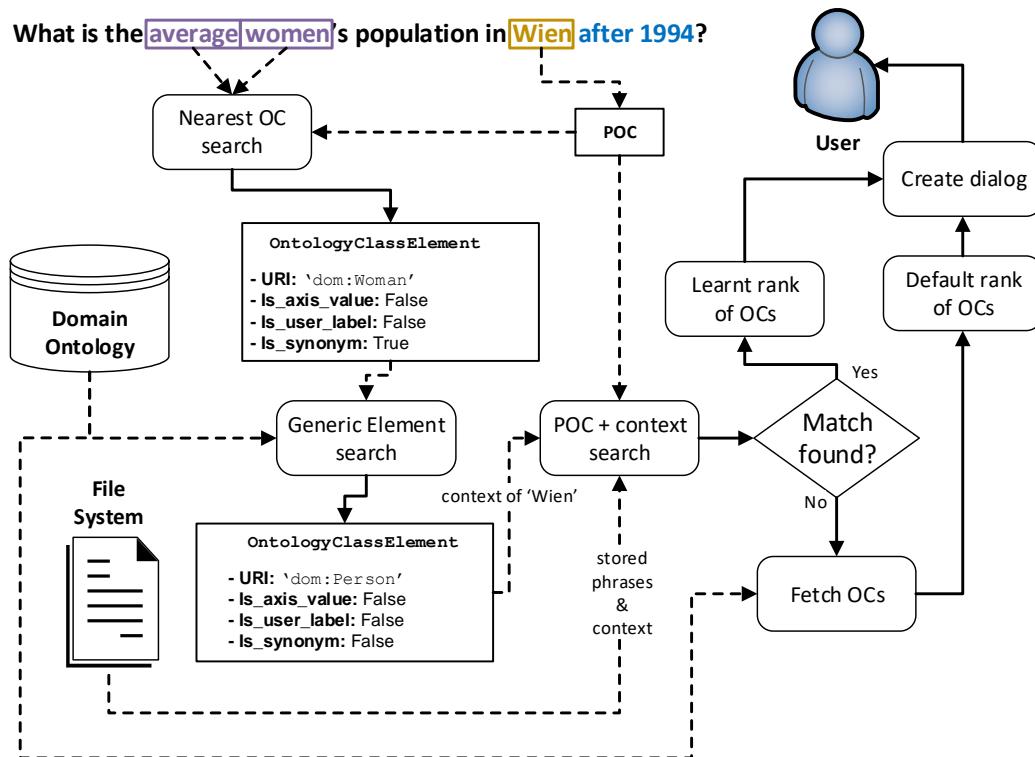


Figure 5.19: Workflow of the learning subsystem. An example query is shown: the system will eventually learn that ‘Wien’ in the context of `dom:Person` refers to the city of Vienna (if such knowledge appears in the domain ontology).

Importantly, if the voting score for a suggestion exceeds a certain configurable threshold, the disambiguation or mapping dialog is resolved automatically by the system without consulting the user. This is perhaps the most valuable learning mechanism of AUDiaL, because it exploits the collective knowledge stemming from its user base in order to extend the lexicon that the system knows about terms that are used frequently in context. The more common a

given set of similar queries is, the faster the system will learn about it and will not require explicit consolidation to resolve them. In addition, AUDiaL can be trained beforehand by expert users who are familiar with the semantic structure of the underlying KB and can map terms not present in the KB to existing ones more effectively in order to reduce the amount of dialogs that will be presented to end users. Therefore, the learning subsystem enables the collective intelligence of its users to improve the habitability and usability of the system, as prescribed by functional requirement FR5 (c.f. Section 5.4).

## 5.9 Task Execution Phase

The Disambiguation phase of the natural language to Knowledge Base (KB) pipeline ends when the input query has been fully consolidated i.e. no more ambiguous phrases remain in the Parsed Query instance. At this point, the task to be carried out and its parameters have been identified. The goal of the *Task Execution phase* (see Figure 5.20) is to perform the actual task being demanded by fetching resources from the KB and manipulating them accordingly. The end result of this phase is either a relevant set of KB triples or an aggregation thereof (textual or numerical) that is passed on to the next phase of the processing pipeline (Figure 5.2).

A consolidated query contains the following members which enable it to be resolved during the Task Execution phase:

- Zero or more Ontology Concepts (OCs) underpinned by a VTO task. If no other task has been found, AUDiaLs will default to performing a `vto:Filter_Task` analytical task, which would be executed regardless as the precursor to the main analytical task.
- A number of Cardinal Query Filters (CQFs) with associated information: operator, operand, and whether the filter has to be negated.
- The remaining OCs of the query, which are converted to Nominal Query Filters (NQFs), see Section 5.6.2 above, including metadata about whether the filter has to be negated and whether the filter refers to a user-defined label, to a value with respect to a graphic space, or to a text in the diagram.

With this information, a Parsed Query object is resolved as follows. First, the system detects whether the query is being executed against a loaded domain ontology or against a semantically-enhanced graphic. If all OCs in the Parsed Query object refer to resources from the visualization ontologies underpinning a semantically-enhanced graphic, a closed-domain procedure that allows for complex query resolution including high-level analytical tasks takes place. Otherwise, we follow the open-domain SPARQLs generation procedure of FREyA [DAC10b] to query domain ontologies. Let us now describe these two approaches, mostly focusing on the former one it the most likely to be encountered by AUDiaLs's users.

A VTO task performed on a semantically-enhanced graphic begins, as shown in Figure 5.20, by filtering its graphic objects according to the nominal and cardinal query filters of the

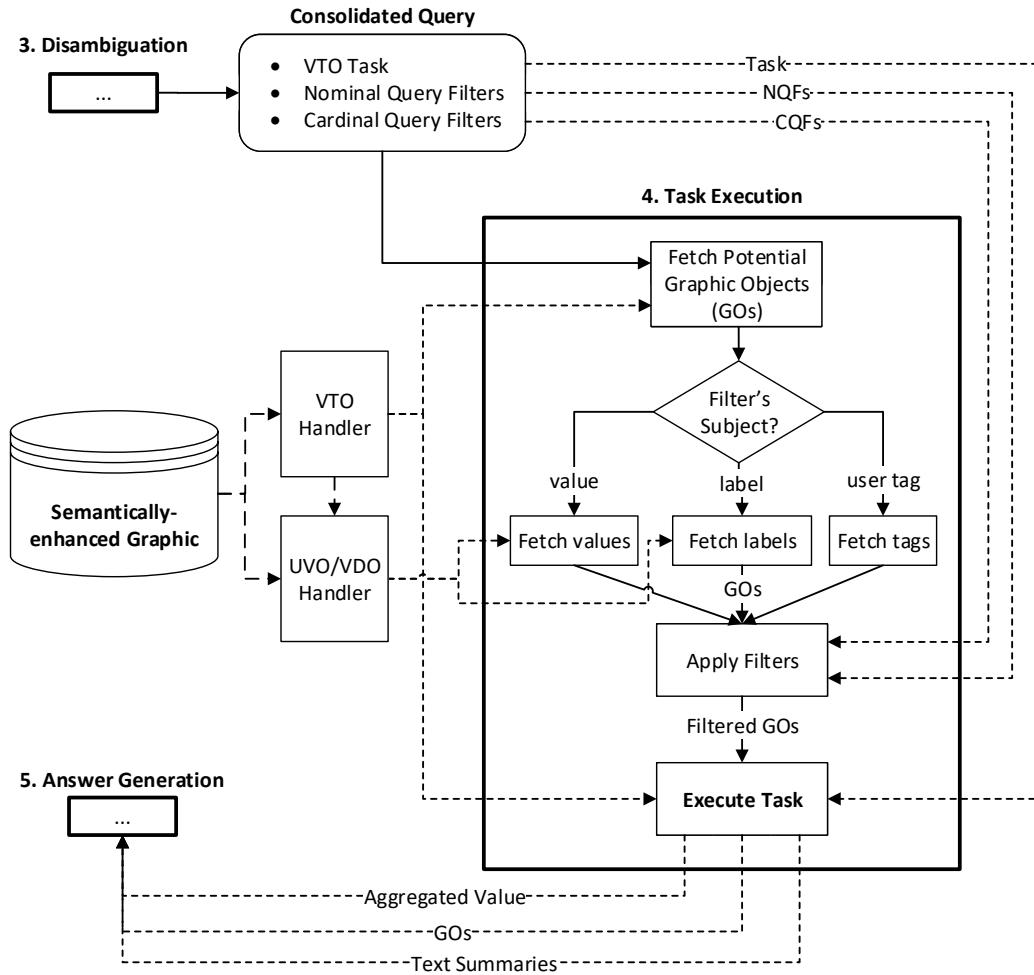


Figure 5.20: Task Execution Phase of AUDiaL carried out on a semantically-enhanced diagram. The inferred VTO task is executed against a filtered set of the diagram's graphic objects and its diverse outputs sent over to the Answer Generation phase for their final display to the user.

Parsed Query instance. Carrying out a `vto:Filter_Task` analytical task is a precursor to any other task and depends on the specific diagram type being queried. For instance, when interacting with a bar chart (or any other statistical chart containing metric bars), all graphic objects with associated syntactic role `vdo:Metric_Bar_SR` are considered as potentially relevant objects for the task. The first step to executing a filtering task thus is to single out those graphic object instances in the diagram that may carry important information.

Next, each Nominal Query Filter (NQF) in the Parsed Query is executed against the selected

graphic object instances. A NQF may be associated to three kinds of element:

- An Ontology Concept (OC) from the KB, more specifically, a string Literal object of a labelling graphic relationship. In this case, AUDiaLs fetches all graphic objects from the KB which are associated to labels matching the NQF (or not matching, if the NQF is negated) through `vdo:is_labeled_by` property occurrences.
- A numerical value with regard to a metric axis. A filtering operation of graphic objects according to their associated numerical value depends on the metric space(s) they inhabit. At the moment, AUDiaLs supports diagrams with only one-dimensional metric spaces. Therefore, the associated value with respect to the metric axis (if any) of each informational graphic object is retrieved. AUDiaLs can compute this value automatically from the position and values of the tick marks of the metric axis and the relative position between the axis and the graphic object being considered. Then, those graphic objects not matching the value of the NQF are ignored.
- A user-defined label. As it will be described in Section 6.4, users may introduce personalized textual labels to any informational graphic object in the diagram to aid them with navigating and analyzing its contents. Each NQF is compared against existing user-defined labels, and, if a match is found, all labelled graphic objects are considered and passed on to the task execution process.

After every NQF has been settled, it is the turn for any existing Cardinal Query Filter (CQF) of the Parsed Query object in a corresponding manner to how NQFs are processed. As described in Section 5.8, at this stage CQFs have been determined to refer to either (1) numerical labels of the diagram or (2) a filtering of values with respect to a metric axis. As such, graphic objects in the diagram are filtered so that they match all CQFs accordingly. For example, let us consider a CQF which has operator ‘less than’, operand ‘1994’, is not negated, and applies to diagram labels. This CQF will result in considering only those graphic objects labelled via `vdo:is_labeled_by` property occurrences to numerical labels with a value of less than 1994.

Once all relevant OCs from have been retrieved from the KB, the actual task execution procedure takes place. This process is entirely dependent on the VTO task being carried out, which was detected during the previous phases of the natural language to KB pipeline. The remaining of this section discusses the implementation of each analytical task described by VTO (c.f. Section 3.9). Each task is first formalized via its *pro-forma* abstract [New93] (when possible, adapted from their definition in [AES05]). We will then proceed to describe how the task is executed by AUDiaLs’s controller in order to communicate the end result to the user. Navigational tasks, having been included to improve the accessibility of the NLI system, are described in next Chapter (Section 6.3).

The following terms will be employed in task descriptions and *pro-forma* abstracts:

- A *data case* is the whole of inferred nominal, ordinal, and cardinal information carried by an informational graphic object in a diagram. It corresponds to an individual observation in a tabular data model.
- An *attribute* or *variable* corresponds to a ddo:Variable instance, see Section 3.8.3. In a tabular data model, it is akin to a table header.
- An *aggregation function* is a function that yields a numeric representation of a set of data cases, such as an average or count.
- A *descriptive function* is a function that yields a verbal description of a set of data cases, such as a listing or high-level analysis.

### 5.9.1 Reading Tasks

Three reading analytical tasks are covered by VTO (c.f. Section 3.9.1), namely: *retrieve value*, *filter*, and *retrieve adjacent*, of which the latter one does not cannot be realized in statistical charts are therefore has not been included yet in the AUDiaL prototype. These are, for the most part, basic tasks that do not require much computation or aggregation of data cases and can be easily answered. Note that AUDiaL expects its users to give an explicit, operating definition of any condition appearing in an input query. For example, the system would be unable to resolve queries such as “which states have a low population?”, as an explicit definition of what constitutes ‘low’ would be necessary for a Query Filter to be properly constructed e.g. “which states have a population lower than 500,000?”.

A **filtering task** adheres to the following *pro-forma* abstract: “Which data cases satisfy conditions  $C_1, C_2, \dots, C_n$ ?”. In AUDiaL, conditions are expressed by Nominal Query Filters (NQFs), which define equality and inequality conditions on variables, and Cardinal Query Filters (CQFs), which convey logical conditions on cardinal variables. As described in previous section, a filtering task is always performed before any other analytical task in AUDiaL in order to select the relevant data cases on which it will be carried out. The end result is a number of data cases (informational graphic objects). If no other task is present in the Parsed Query object, these data cases are passed directly to the Answer Generation phase of the processing pipeline.

A **retrieve value task** is a task of the form: “What are the values of variables  $X_1, X_2, \dots, X_n$  in the data cases  $d_1, d_2, \dots, d_m$ ?”. For example: “what was the population of Vienna in 2004?”. Since AUDiaL returns all labels and values of the graphic objects that belong to an answer, this task is akin to a filtering task containing only Nominal Query Filters. Therefore, after NQF evaluation, the corresponding data cases are passed on to the Answer Generation phase, where their variables and values will be given to the user.

### 5.9.2 Comparison Tasks

AUDiaL accounts for the six comparison tasks available in VTO (c.f. Section 3.9.2). Namely, *compare*, *find extreme* (maximum and minimum), *compute intersection*, *sort*, and *determine range*. However, some of these may only be invoked when navigating the diagram, see below.

A **compare task** follows the *pro-forma* abstract: “compare the variables of data case  $d_i$  and data case  $d_j$ ”. This task thus must follow the execution of a reading task in which the specific data cases  $d_i$  and  $d_j$  have been identified. At the moment, AUDiaL keeps a dialog context of navigational tasks only. Therefore, a compare task must follow a navigational task (Section 6.3.2) that has set a graphic object in the diagram,  $d_i$ , as the current node being visited. The user may, for example, first visit a data point with attributes *Vienna*, *Male*, *2004*, and *1,000,000 inhabitants* whereupon he or she may issue a query such as “compare this to *Vienna* and *Female*”. The system will then proceed to return all attributes of informational graphic objects with matching filters by performing a Retrieve Value task, and their comparison computed by the Answer Generation module. In this case, as shown in Figure 5.21, the differences in populations between male and female inhabitants of Vienna will be showcased to the user.

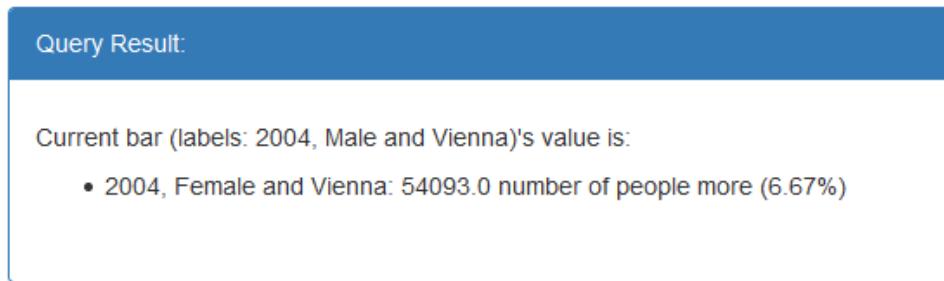


Figure 5.21: The result of a *Compare Task* being displayed to the user.

A **find extreme task** is formally defined as follows: “what are the top/bottom  $N$  data cases with respect to attribute  $A$ ?” For example, “which state had the highest population in 2014?”. As introduced in Section 3.9.2, VTO divides this task into a *find maximum task* and a *find minimum task*. Once again, this task is preceded by a reading task that filters data cases according to the attribute  $A$  as well as any other conditions appearing in the input query. Afterwards, the top/bottom graphic objects according to  $A$  are sent out to the Answer Generation module so their information can be given to the user.

A **determine range** task conforms to the abstract “what is the range of values of attribute  $A$  in a set  $S$  of data cases?”, such as “which years are included in the population data?”. At this time AUDiaL supports determine range questions enquiring about data values of graphic objects along a metric axis. Therefore, each time the system detects that such task is being asked, it returns the graphic objects with inferred maximum and minimum value analogously to the handling of a find extreme task. It is then the task of the Answer Generation phase to create a corresponding answer.

A **sort task** is enquired as “What is the sorted order of a set  $S$  of data cases according to their value of attribute  $A$ ?”. An example of this kind of task is the query “sort states according to their population in 2014”. Once more, sort tasks are assumed in AUDiaL to refer to attributes involving data values along a metric axis. Therefore, the filtered data cases

resulting from CQF and NQF resolution are sorted accordingly and sent over to the Answer Generation phase so that each of them, or a selection thereof, can be explained to the user.

A ***compute intersection task*** is an analytical task of the form “What is the result of the intersection (according to attribute  $A$ ) of data cases  $d_1, d_2, \dots, d_m$  ?”. For example, a query underpinned by a compute intersection task may be “what states have a population greater than 200,000 but smaller than 1,000,000 persons?”. AUDiaL implements compute intersection tasks as a special case of filtering task (c.f. Section 5.9.1). After filter resolution, data cases matching all inferred CQFs are ordinarily sent over to the Answering Generation phase for display. Note that other set algebraic operations, i.e. set union and complementation, are currently not supported in the prototype.

### 5.9.3 Derived Value Tasks

Derived value tasks of VTO are supported by AUDiaL, following the *pro-forma* abstract “What is the value of aggregation function  $F$  over a given set  $S$  of data cases?”. It must be however noted that there exists an undetermined number of possible aggregation functions that may be applied to a dataset. Therefore, only the most commonly employed aggregation functions have been regarded during the prototype’s implementation. For example, the arithmetic mean of values  $v_1, v_2, \dots, v_m$  for data cases  $d_1, d_2, \dots, d_m$  is easily computed as follows:

$$F(d_1, d_2, \dots, d_m) = \frac{1}{m} \sum_{i=1}^m v_i \quad (5.2)$$

Note that the aggregation function is once again executed against values of informational graphic objects with respect to some metric axis. As usual, these tasks are preceded by a filtering operation (Section 5.9.1) in which all Cardinal and Nominal Query Filters present in the Parsed Query object are inspected against a relevant set of graphic objects. The values of the resulting graphic objects with respect to the diagram’s axis are then passed on to the corresponding aggregation function underpinned by the inferred VTO task.

The three most commonly used measures of a dataset’s central tendency are considered by AUDiaL as aggregation functions. Namely, these are the ***arithmetic mean*** (e.g. “what has been the average population of Tirol?”), ***median*** (e.g. “what was the median population of all states in 1994?”), and ***mode*** (e.g. “what is the most common age of DiCaprio’s girlfriends?”). In addition, two more derived value tasks are implemented: the ***compute sum task*** (e.g. “what are the combined populations of Tirol and Vienna in 2014?”) employs a simple sum as aggregation function; and the ***count task*** returns the number of data cases matching the filtering operation (e.g. “how many regions had a population greater than 1 million in 2014?”). These aggregated numerical values are passed to the Answer Generation phase along with the graphic objects relevant to their calculation.

### 5.9.4 Distribution Tasks

Three distribution tasks, which occupy a higher conceptual level than the low-level analytical tasks discussed so far, as described in Section 3.9.4, are included in VTO and implemented by AUDiaL. These tasks have been characterized as follows.

A **characterize distribution task** enables users to determine which distribution is most likely to describe some data. Its *pro-forma* abstract is of the form “what (continuous) distribution best fits a set  $S$  of data cases according to their value of attribute  $A$ ?”. The chosen attribute  $A$  must evidently be a quantitative variable e.g. values of informational graphic objects with respect to some metric axis, as implemented in AUDiaL. At the moment, the prototype contemplates five continuous distributions. Namely, the filtered dataset is fit to (1) the *normal distribution*, (2) the *Gamma distribution*, (3) the *beta distribution*, (4) the *Rayleigh distribution*, and (5) the *Pareto distribution*. These distributions have been chosen because of their ubiquity in the natural and social sciences as means for describing observable phenomena.

This task begins by filtering a set of informational graphic objects,  $X = x_1, x_2, \dots, x_n$ , by evaluating the CQFs and NQFs present in the Parsed Query object. Then, their values with respect to the diagram’s metric axis,  $Y = y_1, y_2, \dots, y_n$ , are retrieved. Next, for a number of selected continuous probability distributions, their statistical parameters (shape, location, and scale) are estimated from  $Y$  by Maximum Likelihood Estimation (MLE). In short, given the parameter space  $\Theta$  for a given distribution, MLE finds the specific combination of parameters  $\theta \in \Theta$  that maximize the likelihood function i.e. the joint density at  $Y$  over the parameter space. This is, by employing MLE, we parametrize a probability distribution in such manner that we maximize the probability of the observed data being generated by it. The probability density function (PDF) for each parametrized distribution is then calculated. The error between each PDF and the data is computed afterwards by a sum of squared errors. Finally, the distribution that minimizes the error is returned as the most likely candidate and passed along  $X$  to the Answer Generation phase for display. This process is also outlined in Algorithm 5.5.

A **find anomalies task** is a high-level analytical task of the form “given a set  $S$  of data cases, which ones among them are statistical outliers according to their value of attributes  $a_1, a_2, \dots, a_n \in A$ ?”. An example query for this task could be: “which regions have exceptional populations?”. AUDiaL implements this task by first performing DBSCAN clustering [EKSX96] on the values with respect to a metric axis of a set of filtered graphic objects. A closer look at the clustering method employed by the prototype is given in the ‘cluster’ task description of Section 5.9.5. Next, all samples that do not belong to any of the computed clusters are considered *noisy samples* i.e. anomalies (see Figure 5.22 for a general example). The anomalous graphic objects are then sent over to the Answer Generation phase of the pipeline so they may be communicated to the user.

A **retrieve summary task** gives a general overview, in natural language, of the contents of a graphic as a whole. It corresponds to the first task of Shneiderman’s Visual Information-Seeking mantra: “overview first, zoom and filter, then details-on-demand” [Shn96], and is a basic, although very high-level, task that enables users to become familiar with a dataset

---

**Algorithm 5.5:** *Find distribution* analytical task.

---

```

1 function find_distribution (query, graphic);
  Input : A Parsed Query Object: query
  Input : A semantically-enhanced graphic: graphic
  Output: Most likely distribution that characterizes graphic's underlying data:
            best_f
  Output: Best parameter combination for parametric best_f: best_params
2 DISTR := ["GAMMA", "BETA", "RAYLEIGH", "NORM", "PARETO"];
3  $\mathbf{X} \leftarrow \text{apply\_filters}(query, graphic);$ 
4  $\mathbf{axis} \leftarrow \text{get\_metric\_axis}(graphic);$ 
5  $\mathbf{Y} \leftarrow \text{compute\_values}(\mathbf{X}, \mathbf{axis});$ 
6  $\text{max\_error} \leftarrow \infty;$ 
7 foreach  $f \in \text{DISTR}$  do
8    $\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta} \in \Theta} f(\mathbf{Y}; \boldsymbol{\theta})$ ; // MLE
9    $\hat{\mathbf{y}} = f(\mathbf{X}; \hat{\boldsymbol{\theta}})$ ; // PDF
10   $\text{sse} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ ; // error
11  if  $\text{sse} < \text{max\_error}$  then
12     $\text{best\_f} \leftarrow f;$ 
13     $\text{best\_params} \leftarrow \hat{\boldsymbol{\theta}};$ 
14     $\text{max\_error} \leftarrow \text{sse};$ 
15  end
16 end

```

---

and/or diagram before delving into specific aspects of interest. No *pro-forma* abstract can be given for this task, since it is dependent on the specific way data are illustrated in a diagram. Summaries are offered mainly as an accessibility feature of non-visual diagrams so that Visually Impaired Persons (VIPs) can familiarize themselves with the graphic before navigating or querying it. Therefore, this task will be explained in greater detail in Section 6.6 of next chapter.

### 5.9.5 Correlation Tasks

The last group of analytical tasks is that of correlation tasks which aim to find statistical associations between data attributes. Two correlation tasks are considered in VTO and implemented by AUDiaL. Namely, *correlate task* and *cluster task*, which are outlined in the following.

A **correlate task** aims to find a predictive dependency between attributes of data cases. Its *pro-forma* abstract is “what is the correlation between attributes  $a_i \in A$  and  $a_j \in A$  over a given set  $S$  of data cases?”. An example query could be “is there any correlation between population of Vienna and year?”. Correlate tasks are very commonly undertaken in diagrammatically displayed data in order to determine whether there is any relationship (and

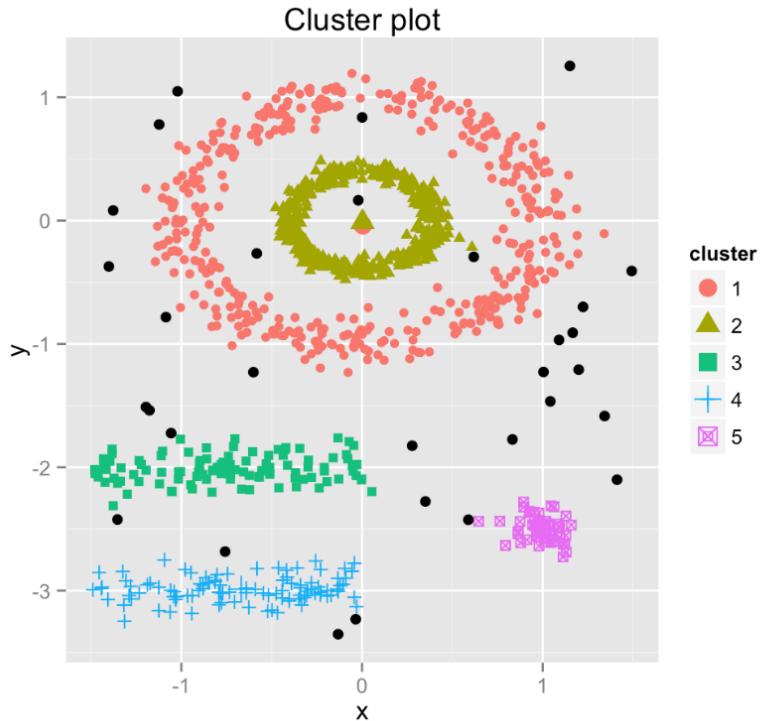


Figure 5.22: Finding anomalies in a 2-dimensional dataset via DBSCAN clustering [Kas19]. 5 high-density clusters are found as shown by the coloured markers. The remaining samples that do not belong to any cluster (black dots) are considered outliers or noisy samples.

if so, the magnitude and direction of such relationship) between two quantitative variables measured for the same individuals. We compute correlation via the widely employed Pearson product-moment correlation coefficient,  $r_{xy}$ , between attributes  $x$  and  $y$  for a given set of data cases  $S$ , which is defined as follows:

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (5.3)$$

Where  $n$  is the sample size (number of filtered graphic objects),  $x_i$  and  $y_i$  are individual samples indexed at position  $i$ , and  $\bar{x}$  and  $\bar{y}$  are the empirical means for  $x$  and  $y$ , respectively. The resulting correlation coefficient  $r_{xy}$  ranges between -1 and 1, where a value of 1 implies a perfect linear correlation between  $x$  and  $y$ , a value of -1 indicates a perfect negative correlation i.e. that all data points lay on a line where  $y$  decreases as  $x$  increases, and a value of zero implies that there is no correlation between the variables. Broadly speaking, a positive value of  $r_{xy}$  suggests that both variables change in the same direction, with a negative value

implying a change in opposite directions. The magnitude of the value indicates how strongly the two variables are related.

AUDiaL restricts this task to two circumstances; namely, it allows users to inspect the correlation between two variables expressed by metric spaces or between one variable expressed by a metric space and another one expressed by labelling relationships. A correlate task is carried out as follows: first, any NQFs are evaluated against the KB. Then, if two metric axes are present in the diagram, the correlation  $r_{xy}$  between their corresponding attributes for the filtered graphic objects are computed. Otherwise, the correlation between a metric axis and the graphic objects' labels is computed. This value is transmitted to the Answer Generation phase of the processing pipeline along with the filtered graphic objects. Lastly, it must be noted that this method is only concerned with finding linear relationships between variables. A correlation coefficient of zero does not imply that the two variables are independent, as there might exist a non-linear relationship between them. End users should be aware of this fact before enquiring correlation tasks in AUDiaL in order to avoid potential pitfalls.

A **cluster task** groups (*clusters*) a set of data cases according to how similar certain attributes thereof are. This task is formally described as “which data cases in a set  $S$  of data cases are similar in value for attributes  $a_1, \dots, a_n \in A$ ”? For example, a cluster query may be “group the states of Austria according to their population”. A clustering operation is carried out by AUDiaL as follows. First, all CQFs and NQFs of the parsed input query are resolved and the relevant informational graphic objects fetched from the knowledge base. Next, their values with respect to a metric axis are retrieved and clustered via DBSCAN [EKSX96], see Figure 5.22 for a two-dimensional example. For each cluster inferred, its constituent graphic objects are lastly grouped and sent over to the Answer Generation module for their eventual display. The main advantage of using DBSCAN as opposed to other clustering methods is that the number of clusters does not need to be known beforehand, and therefore users need not be asked about something that they are very unlikely to know. In the following, we proceed to described DBSCAN clustering in greater detail.

The clustering method is preceded by feature re-scaling in order to normalize different attributes so that each of them contributes proportionally to the clustering. Otherwise, attributes with a broad range of values would dominate other the rest when computing the distance between data cases. This is done via *z-score normalization*, in which the range of each attribute is re-scaled to zero mean and unit standard deviation. Note that in the basic uni-dimensional grouping performed by the prototype such scaling is not necessary; however, this step has been included to accommodate for eventual multidimensional clustering capabilities. Next, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering is performed on the normalized data values. This method groups together data points according to some distance measure, most commonly the Euclidean distance, between samples. Two hyperparameters need to be prescribed before clustering; namely,  $\epsilon$  (set to  $\epsilon = 0.3$  in AUDiaL as a general-purpose compromise) which specifies a radius around a sample's neighborhood; and  $m$  (that we set to  $m = 2$ ) which stipulates the minimum number of samples that can make up a cluster. The algorithm then begins with an unvisited random sample, for which the  $\epsilon$ -neighborhood of samples is retrieved. A cluster is created from the

neighborhood if it contains more than  $m$  individuals; else the initial sample is annotated as a noisy sample. For each point already in a cluster, its  $\epsilon$ -neighborhood is also added to the cluster. Once no more points can be added to the cluster, the process starts again with a new starting unvisited sample. For a more detailed and formal description of the method, the reader is directed to [EKSX96]. The end result of DBSCAN is a number of labelled samples according to the cluster they belong to as well as a number of noisy samples. Noisy samples are of use in other analytical tasks, especially the find anomalies task already described in this chapter.

## 5.10 Answer Generation Phase

The last phase of the natural language (NL) to semantically-enhanced graphics processing pipeline, known as the *Answer Generation Phase*, is in charge of collecting the final factoid results from the Task Execution phase and display them in natural language to the user. This phase is generally the easiest one to execute, as all relevant data is already given and the system just needs to decide in which order to print it, whether to truncate or summarize it, and which additional information needs to be included on a task-specific basis.

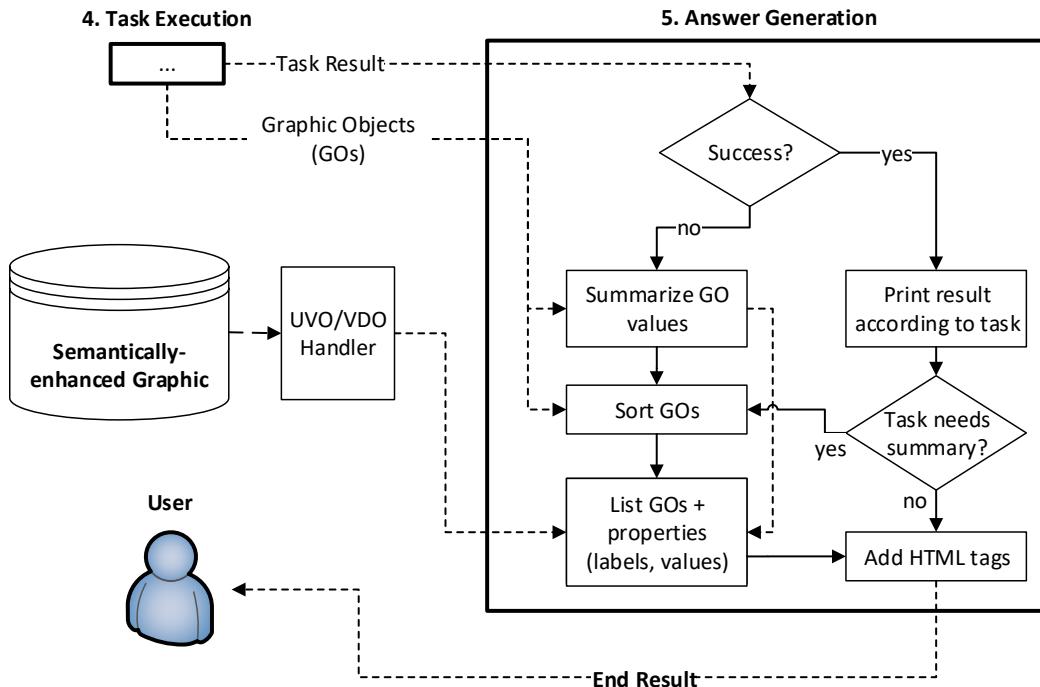


Figure 5.23: Answer Generation phase of AUDiaL. Final results after executing a task on a semantically-enhanced graphic are expressed in natural language and embedded on the Web user interface.

As shown in Figure 5.23, the last phase of the processing pipeline starts by checking whether the Task Execution phase returned with a successful state. Recall that the previous phase does not return a task answer if no task resources were found in the Parsed Query object. In such a case, the passed-in informational graphic objects (GOs) resulting from parsing the query's filters and their properties are listed to the user. This process is divided into three sub-steps, as follows:

1. The GOs' values according to their position with respect to the diagram's metric space (if it exists) are retrieved and summarized by common descriptive statistics, namely their mean and total sum.
2. A diagram-specific algorithm is executed that sorts the output graphic objects to be displayed. It takes into account the navigational order of the GOs as well as any domain-specific priority for certain GOs. The navigational order of graphic objects is an accessibility feature of AUDiaL discussed in Section 6.3.
3. For each GO, in order, their graphic syntactic role, textual labels, and values with respect to the metric axes of the diagram are retrieved.
4. The final answer is compiled as a HTML string with header "the following <n> <syntactic role of GOs> matched your query:", followed by the results of the 3rd and 1st steps, see Figure 5.24 for an example.

The screenshot shows a user interface titled "Query Result". The main content area displays a list of 9 bars that matched the query. Each bar entry consists of a blue bullet point followed by a text string indicating the location and population count. Below this list, two summary statistics are provided: "Total value: 4121730 number of people" and "Average value: 457970 number of people".

Query Result:	
The following 9 bars matched your query:	
• Simple bar with labels: 2014, Burgenland, Male (107640 number of people).	
• Simple bar with labels: 2014, Carinthia, Male (226878 number of people).	
• Simple bar with labels: 2014, Lower Austria, Male (719036 number of people).	
• Simple bar with labels: 2014, Male, Upper Austria (733813 number of people).	
• Simple bar with labels: 2014, Male, Salzburg (269165 number of people).	
• Simple bar with labels: 2014, Male, Styria (530745 number of people).	
• Simple bar with labels: 2014, Male, Tirol (350138 number of people).	
• Simple bar with labels: 2014, Male, Vorarlberg (196279 number of people).	
• Simple bar with labels: 2014, Male, Vienna (988036 number of people).	
Total value: 4121730 number of people	
Average value: 457970 number of people	

Figure 5.24: Answer to the query “what was the male population in 2014 like?”. Because no formal task is found in the input query, AUDiaL simply retrieves and lists in order all matching graphic objects and their properties. In addition, a short statistical overview of their values is given.

On the other hand, if a suitable task was found in the input query, the Answer Generation phase has to produce a task-specific answer in natural language. For instance, in the case of a cluster task (Section 5.9.5) this phase receives as input a number of grouped informational graphic objects. As shown in Figure 5.25, answer generation for this specific task consists of, first of all, listing the clusters identified by the DBSCAN algorithm along with their constituent GOs. Moreover, common labels and values to the GOs in a group are displayed in the cluster heading. In addition, each listed GO is complemented with their individual textual labels and values.

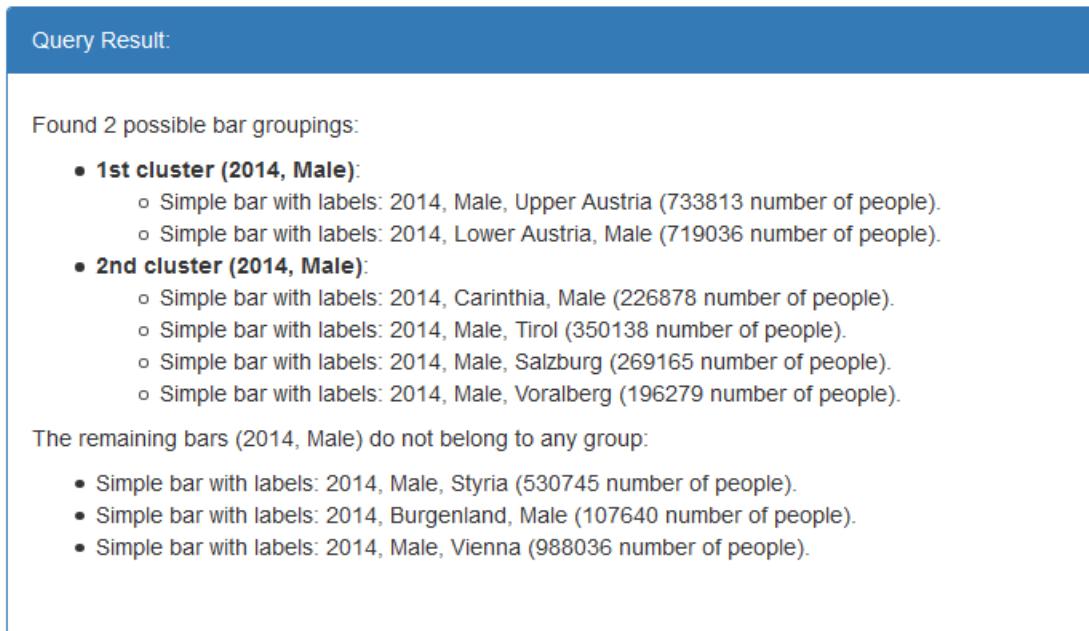


Figure 5.25: Answer to the query “please group the male population in 2014”. AUDiaL’s answers depend, for the most part, on the main task that has been identified in the input query.

Lastly, those GOs not belonging to any group (outliers) are listed individually. In the given example, the first cluster corresponds to regions with an average population of around 700,000 persons. The second cluster contains regions with small populations of between 300,000 and 400,000 persons approximately. The remaining regions cannot be grouped together and are thus listed as outliers individually. The example we have just seen shows how, depending on the task at hand, client applications should create an answer according to the needs of their user base. Being a heterogeneous, task-dependent process, we will therefore not cover the answer generation method of AUDiaL for all remaining tasks.

Some tasks are given, in addition to their standard answer, additional summary information in order to give users a high-level overview of the answer’s data. Tasks that require a summary in AUDiaL include all analytical tasks belonging to the Derived Value category (c.f. Section 5.9.3) as well as Characterize Distribution tasks, outlined in Section 5.9.4. When one of these tasks

is executed, a list of the graphical objects that were considered when computing the answer is given to the user. For example, if a Compute Average task is requested, not only the numeric aggregation is given to the user, but also which GOs, e.g. metric bars, make up the aggregation, as well as their attributes as values (see Figure 5.23).

After the final answer to an input query has been received, the system accepts a new prompt in natural language from the user, and the natural language to KB pipeline we have discussed in this chapter is executed again. This concludes the bulk on the specification of AUDiaL's design and implementation. Note however that some of the requirements listed in Section 5.4 have not been covered yet and will be taken into consideration in the next chapter, which tackles the usability and accessibility features of AUDiaL given the special access needs of its intended target users.

## 5.11 Summary

This chapter has illustrated the treatment design stage of a prototype Natural Language Interface (NLI) to semantically-enhanced diagrams (c.f. Chapter 4), named Accessible Universal Diagrams through Dialogue (AUDiaL). First, in Section 5.1, the conceptual framework upon which the prototype was built has been expanded by studying current NLIs (and, more specifically in Section 5.2, NLIs to conceptual models e.g. ontologies). The design process of AUDiaL following the design science methodology was introduced in Section 5.3. Next, a number of functional and quality requirements for the prototype were laid out in Section 5.4. Requirements are goals for the to-be-designed treatment (i.e. AUDiaL), which allow us to design a prototype meeting stakeholder's goals with the end aim of answering the knowledge questions set out in this thesis (c.f. Section 1.4).

The bulk of the treatment design cycle was described in the remaining sections of this chapter. Some initial architectural considerations, as well as the design of the main user interface of the Web prototype, are discussed in Section 5.5. State-of-the-art, end-to-end automatically trainable, Natural Language Processing (NLP) models based on deep learning and other machine learning methods do not fit our needs due to low precision when querying graphically-displayed numerical data, the need for high-level inference, and a lack of suitable training datasets. Therefore, AUDiaL's back-end design is grounded on a traditional NLP pipeline. Such a system must implement at least five stages (see Figure 5.2), namely:

1. **Question Analysis** (Section 5.6). This phase analyses the syntactic features of the input natural language query. The input query is first normalized. Next, a Parse Tree (PT) of the query is generated. The PT is then analyzed following several heuristics resulting in a number of Potential Ontology Concepts (POCs). POCs are phrases of the input query that have the potential of being mapped to resources in the KB. We generate POCs by extending the approach of FREyA [DAC12]. In addition, we introduced the concept of Cardinal Query Filters (CQFs), which enable more complex queries to be resolved.

2. **Phrase Mapping** (Section 5.7). This phase aims to map each of the query's phrases found during the previous phase to zero or more resources of the KB. For example, the phrase "People" could be mapped to a literal resource, for example "People^^xsd:string" underpinning a label's text in a semantically-enhanced diagram through a datatype property occurrence. This phase not only searches for query phrases as-is in the KB, but also lemmatized versions and synonymous phrases thereof.
3. **Disambiguation** (Section 5.8). The disambiguation phase resolves any remaining POCs, CQFs, and disambiguates between resources of the knowledge base that have been mapped to the same query phrase during the previous phase of the pipeline. If disambiguation cannot be performed automatically users are asked for explicit feedback. After this phase is completed, the input query has been fully resolved and can be executed against the KB.
4. **Task Execution** (Section 5.9). This phase is in charge of executing the corresponding task stemming from a consolidated user query. First, KB resources corresponding to the task(s) to be executed are identified. Next, a number of KB triples that match the filters of the consolidated user query are retrieved. Lastly, the chosen analytical task is executed on the matching triples. All analytical tasks defined by the Visualization Tasks Ontology (VTO) as characterized in Chapter 3, have been implemented in AUDiaL.
5. **Answer Generation** (Section 5.10). To conclude, the task result is expressed in a suitable manner, depending on the task, in natural language and dynamically embedded into AUDiaL's accessible Web interface.

AUDiaL realizes all stages of such a pipeline and is a functional NLI to semantically-enhanced diagrams. However, not all requirements defined in Section 5.4 are covered by the prototype as described so far. More specifically, this chapter has only dealt with the functional domain of the prototype; requirements related to the accessibility and usability of the artifact, such as FR2, are described in next chapter. In addition, the last cycle of the design cycle of this thesis, namely, the validation of AUDiaL in its problem context, is also explained in the following chapter.

# CHAPTER 6

## Accessibility and Validation of NLIs to Diagrams

After treatment design has been completed, *artifact validation* enables researchers to justify that the treatment the artifact models would contribute to stakeholder goals when implemented in the problem context. This can be attained by determining that the artifact satisfies its requirements [Wie14]. Given that not all artifact requirements outlined in Section 5.4 have been satisfied yet, this chapter sets out to adapt the AUDiaL Natural Language Interface (NLI) prototype designed in previous chapter to comply with the remaining requirements. The chapter is divided into two main sections. First, we will characterize a number of accessibility features included in AUDiaL in order to satisfy its unique requirements with regard to its usage by Visually Impaired Persons (VIPs). With an accessible artifact at hand, the empirical cycle of the greater design cycle (c.f. Section 1.5) may take place next, in which the scientific questions posed in Section 1.4 can be answered. Namely, the second half of this chapter details an empirical validation of the designed artifact with target users.

This chapter thus concludes a full iteration throughout the design cycle of the present research, from which answers to the initially posed research questions are derived. It must be noted that the design cycle is generally meant to be run several times until a satisfactory real-world treatment to the original problem is implemented. As it will be outlined in Section 6.13, the first cycle concluded in this thesis yielded a number of improvements to the treatment which, in turn, may trigger a new iteration through the design cycle.

### 6.1 Motivation

The artifact described in this document thus far consists of a Natural Language Interface (NLI) prototype to semantically-enhanced graphics. The artifact enables users to carry out common analytical tasks supported by visually displayed numerical data by means of free-formed

queries in natural language. However, recall that the problem being treated is specific to the needs of blind persons and not the general population. Therefore, the prototype has to be adapted so that it can be used in an effective and efficient manner by the intended user base. Such an adaptation is made up of two independent aspects: first, the user interface must be designed in such a manner that it can be accessed in a convenient manner by blind users employing standard assistive technology. This process is described in Section 6.8. Second, whereas enabling the execution of analytical tasks on diagrams (c.f. Chapter 5) is a vital aspect of accessing diagrammatically-displayed data, it only covers the Information Retrieval (IR) aspect of non-visual diagram reading. Just as important for blind consumers of diagrams is the ability to explore them spatially [Bro08]. Therefore, the NLI prototype designed in the previous chapter ought to be revised by enabling the execution of specific tasks that would be of little or no use to sighted persons, but which allow blind users to navigate and explore a diagram non-visually in an effective manner.

### 6.1.1 Cognitive Benefits of Diagrams

Some tasks underpinned by VTO (c.f. Section 3.5.2) were included with the needs of visually impaired users in mind. More specifically, the descendent navigational tasks of the `vto:Navigation_Task` super-class were included in the conceptual model with the intention of enabling blind persons to navigate diagrams in a non-visual manner. These tasks would not be of any use if AUDiAL's end users were sighted, since visually displayed content can be navigated effortlessly by means of sight. Navigational tasks were introduced to VTO after studying the specific cognitive benefits that diagrammatically represented information provide to sighted persons. In particular, research has shown that some of the reasons why sighted persons prefer diagrammatic representations to linear descriptions for problem solving are [LS87, GMA10]:

- **Resemblance preservation:** Diagrams possess the ability of directly resembling what they represent, as opposed to linear descriptions, in which the reader has to imagine the entities that are being described. This resemblance may take place in two different ways:
  - *Literal resemblance*: many diagrams literally preserve the topology and geometric relationships of what is being represented, including the relative size and position between elements. This is the case in pictures and most maps.
  - *Homomorphic resemblance*: diagrams using this type of resemblance use topology and geometric relationships metaphorically to represent abstract relationships with similar abstract properties. For instance, in family trees a horizontal line connecting two people indicates that they are married.
- **Spatial indexing and navigation:** diagrams avoid large amounts of search for the elements needed to make a problem-solving inference by grouping all information that is used together, which avoids the need to use and match symbolic elements. Besides *locational indexing*, they also allow *two-dimensional indexing*, this is, the ability of

locating an object by its coordinates on the two-dimensional plane. For example, two-dimensional indexing allows a sighted reader to find the values of a given point in a scatterplot by matching its location to horizontal and vertical positions on the plot's axes. In other words, relevant nodes can be grouped much more efficiently on a two-dimensional space than in a one-dimensional line of text.

The cognitive advantage of indexing allows for the innate capability of efficiently navigating the diagram by means of sight. When we talk about *navigating* a diagram we refer to the methods by which readers find their way around the diagram's elements with some goal in mind. *Wayfinding* is not a single task but rather a general category of navigational tasks performed on diagrammatically represented data. According to the functional framework for wayfinding of Allen [All99], there exist three general types of spatial wayfinding tasks based on their purpose, namely:

- *Commute* is the “travel between two places known to the traveler along a route that is familiar” [All99]. This is an activity with a low level of uncertainty due to the familiarity of the route being undertaken, and its success is characterized by the time it takes to complete the travel i.e. its efficiency. Commutes often “involve not only a primary route but a number of alternatives that can be taken when circumstances demand” [All99]. In addition, commutes may contain a number of intermediary destinations before reaching the final one.
- *Exploration*, i.e. foraging for spatial information, involves “traveling into unfamiliar territory for the purpose of learning about the surrounding environment” [All99]. The aim of this kind of task is to discover new places and new routes that link them to one another in order to ascertain useful information. These tasks are very uncertain for two reasons. Firstly, the usefulness of the information acquired from exploration cannot be ensured ahead of time; and secondly, the explorer may become lost (i.e. unable to return to a known location) during the process. The success of an exploration is determined by “the number and value of new places and routes that are added to the existing environmental knowledge while the traveler remains oriented to a familiar place” [All99].
- *Quest* consists of a “travel from a familiar place of origin to an unfamiliar destination [...] which is known to exist but which the traveler has not visited previously” [All99]. Therefore, for it to be found, knowledge of the destination needs to be conveyed symbolically e.g. by means of a map or verbal description. Success in a quest task is measured first by whether the destination could be reached, followed by the ratio between distance traveled to time spent.

Several means are posited to enable the undertaking of these navigational tasks, including *piloting*, *navigation by cognitive map*, and *path integration*; these means for accomplishing wayfinding tasks are in turn made up of a number of constituent cognitive abilities such as *object recognition*, *paired-associate learning*, and *sequential learning* [All99]. These cognitive abilities are visual in nature. For example, the means of piloting involves landmark-based navigation, for which it is required that visual

landmarks are recognized, selected, monitored and kept in memory [All99]. Therefore, in order to enable non-visual undertaking of wayfinding tasks, the navigational means constituted by visual cognitive abilities ought to be compensated by other mechanisms.

- **Breadth-then-depth search:** when visualizing data, lots of information can be acquired in parallel from a wide area. The perceptual abilities of the human sight allow persons to scan, recognize, and recall images rapidly. This leads to the ability of intentionally focusing the attention into a narrower, more detailed part of the whole image that the reader finds especially interesting. The capacity to gain a quick overview first, followed by the possibility of effortlessly filtering out uninteresting elements and obtaining details of the relevant ones make visual data display a much more efficient means of retrieving information than linear representations. This cognitive aspect of visualization is very well known in data visualization research, from where the so-called Visual Information-Seeking mantra, that aims to replicate it, stems: *overview first, zoom and filter, then details on demand* [Shn96]. The theoretical framework of Palmer [Pal77] is a classical model formalizing this phenomenon, where it is proposed that the visual form is analyzed hierarchically starting from an overall configuration moving down towards the atomic constituent features of the diagram. Components at lower conceptual level are clustered to form structural units in a selective manner so that connections between units which have important relationships are maximized [Bro08]. For example, spatial proximity is a very common means of determining importance in a domain-dependent manner. Studies performed by Palmer and others indicate that low level cognitive processes deal with information in a manner consistent with his model, and that, by extension, the process of visual perception might organize diagrams into a hierarchical form [Bro08].
- **Computational off-loading:** Diagrams automatically support a large number of perceptual inferences by means of sight, which are extremely easy for sighted humans. *Perceptual inferencing* depends on the visual and spatial properties of the diagram encoding the underlying information in a way that capitalizes on automatic processing to convey the intended meaning. For example, the relationship between the two variables in a line chart (i.e. the trend) can be inferred immediately by means of sight. Diagrams also allow *mental animation*, i.e. mentally activating components in the representation of the system in a serial manner [Heg92]. For instance, someone studying a diagram showing several interlaced gears can mentally ‘move’ the gears, perceiving the direction in which each one of them would spin in real life. In addition, work done by Bauer and Johnson-Laird [BJL93] established that diagrams act as an *external memory* that help persons solve certain kinds of problems by representing the problem states and solution more explicitly thereby making it less likely to overlook possible configurations.

In short, the main cognitive benefits of diagrammatic representations include (1)  that makes otherwise implicit features explicit to the sight; (2) *two-dimensional grouping* of related elements and *hierarchical representations* that aid with search; and (3) facilitation of external cognition by *lowering memory demands*. These cognitive advantages make proper

data display a critical aspect to data analysis. Properly displaying information can make a difference by enabling people to understand complex matters and find creative solutions [Cuk10]. For example, it has been proven that students learn better when textual materials are enhanced with the inclusion of didactic graphics [LAC87]. This applies not only to information retention, but also to deep comprehension, which finally turns into better solving skills [HSR<sup>+</sup>13]. A computer system aiming to make graphics accessible to the visually impaired population therefore must be able to compensate, where possible, for the lack of these benefits that its user base suffers. This is the rationale behind the inclusion of some VTO tasks, whose specific implementation is discussed in the following sections.

### 6.1.2 Accessibility of Natural Language Interfaces to Diagrams

The following sections tackle the problem of adapting the Natural Language Interface (NLI) to semantically-enhanced diagrams prototype (AUDiaL) to the specific needs of visually impaired users in order to satisfy requirement FR2, described in Section 5.4. This problem is threefold: firstly, accessibility mechanisms that enable the NLI to diagrams to be employed more efficiently by blind users need to be included to the prototype in order to prevent user confusion, lostness, and subsequent frustration. Secondly, new tasks ought to be included to the taxonomy of low-level tasks (Visualization Tasks Ontology (VTO), see Section 3.9) that can be performed in diagrams to comprise navigational operations that are carried out effortlessly by sighted readers but which need to be included explicitly in an accessible NLI to diagrams. In the following sections we proceed to describe them in detail. Lastly, Section 6.8 discusses the Web accessibility guidelines that have been taken into consideration in order to make the prototype Web application itself accessible using standard assistive technology.

Each of the cognitive benefits of diagrammatic representations outlined in Section 6.1.1 ought to be compensated, to the greatest extent possible, in a non-visually accessible Natural Language Interface (NLI) to diagrams. Noting that AUDiaL is a pure NLI, this is, that no other means other than natural language (e.g. tactile or sonification mechanisms) may be employed, the accessibility features devised in this section must be accessed and actioned by means of natural language<sup>1</sup>. The greatest drawback of natural language is its sequential nature, as opposed to the multidimensional capabilities of sight. This enforced sequentially entails that users are likely to forget information read long ago, and not much information can be presented at once to the user in order to prevent overloading the user's working memory. With these challenges in mind, we set out to develop interaction mechanisms that still allow blind persons to employ a NLI to diagrams in an effective and efficient manner.

---

<sup>1</sup>Parts of the following sections are published in: Tomás Murillo-Morales and Klaus Miesenberger. Techniques for improved speech-based access to diagrammatic representations. In Klaus Miesenberger and Georgios Kouroupetroglou, editors, *Computers Helping People with Special Needs. 16th International Conference*, Lecture Notes in Computer Science, pages 636–643. Springer, 2018

## 6.2 Resemblance preservation

Most visual attributes with the ability of preserving information resemblance in diagrams, such as shape, size, and color, cannot be accurately represented by textual means. This is perhaps the most limiting disadvantage of NLIs, since tactile diagrams, on the other hand, can carry over some of the original resemblance of the elements they depict, namely shape, size, texture, and position [GMA10]. The most a NLI can realize in this regard is to describe visual attributes by discretizing and categorizing them. All visual attributes underpinned by the Upper Visualization Ontology (UVO) (c.f. Section 3.5.2), have an associated class extension which exhaustively enumerates discrete members of the visual attribute. For instance, the `uvo:Shape_VA` class, which underpins shapes of graphic objects, is made up of 15 individuals characterizing specific shape names such as `uvo:5_point_star_shape` or `uvo:Triangular_Shape`.

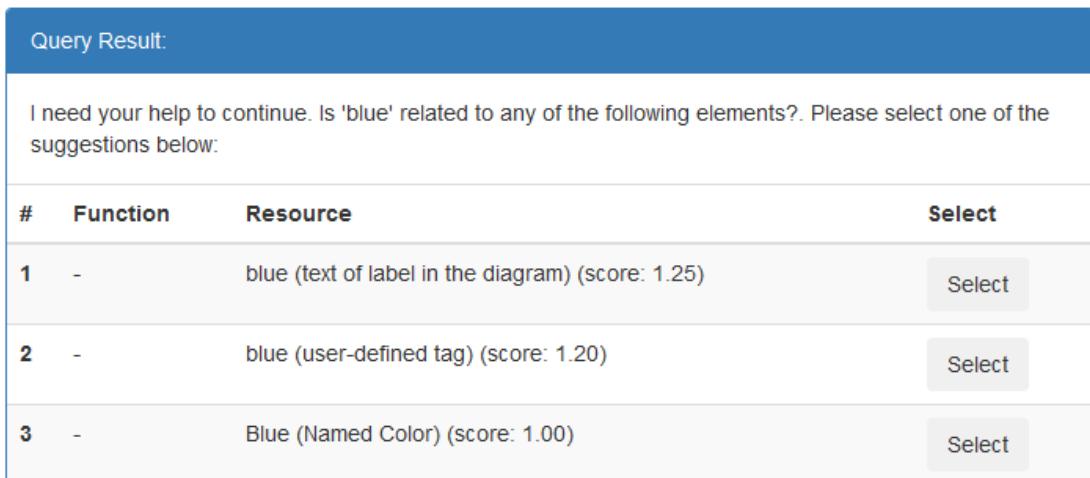


Figure 6.1: A disambiguation dialog (for the input phrase ‘blue’) containing separate suggestions for a diagram label, a user-defined tag on a Graphic Object, and a UVO visual attribute (a member of the `uvo:Color_Named_VA` class). Each suggestion is handled differently by AUDiaL.

The answering of a query referring to visual attributes of graphic objects may be detected by the system and its execution adapted accordingly by selecting a standard low-level filtering task (`vto:Filter_Task`) with either (1) the visual attribute as the operand and ‘equals’ as its implicit operator or, more commonly, (2) `uvo:has_visual_attribute` as the datatype property whose occurrences to retrieve having the visual attribute as triple object. In this manner, even if the literal resemblance of graphic objects is lost in the speech exchange, their visual attributes can be communicated to the user and employed to deduct higher level inferences. Users may, for instance, inquire about graphic objects sharing a certain named color underpinned in UVO (e.g. “blue bars” in a bar chart), size (“small bars”), or shape (“rectangular bars”), given that they have been labelled accordingly in the

semantically-enhanced diagram.

As Figure 6.1 shows, in the situation that there are labels or other diagram elements sharing the same textual representation as the visual attribute, a disambiguation dialog (c.f. Section 5.8.2) may have to be solved by the user. For instance, choosing the third suggestion in Figure 6.1 would lead to fetching all information objects of the diagram having blue color. Every remaining filter present in the Parsed Query object would be resolved afterwards as usual as described in the previous chapter.

It must be noted that during the validation of AUDiaL with blind participants (Section 6.9), visual attributes of graphic objects in the test diagrams were not asked a single time by any participant. It can be safely assumed that blind users are not accustomed to seek visual attributes of diagrammatically-represented data when inquiring about it unless they are explicitly instructed to do so. Therefore, the issue of communicating the literal resemblance of object does not seem to hinder the usability and accessibility of the NLI for its user base in a significant manner.

Regarding making up for the lack of direct homomorphic resemblance representing abstract relationships and properties, these are most commonly and conveniently underpinned, in a domain-specific manner, with the inclusion of a domain ontology that extends the semantics of the Graphic Objects and Graphic Relationships appearing in a diagram. For example, a marriage between two persons, depicted in a family tree link diagram by a horizontal link connecting them, may be represented by the Family Tree ontology with an object property occurrence: (`dom:Person_A`, `dom:married_to`, `dom:Person_B`). The domain ontology handler included in AUDiaL (c.f. Section 5.8.2) would then be in charge of answering input queries such as “to whom is Person A married?”. It would first map the phrase “married” to the object property `dom:married_to`, and the phrase “Person A” to the instance `dom:Person_A`. The system would then search on the domain ontology, via an automatically generated SPARQL query, for all objects  $X$  of triples having the form (`dom:Person_A`, `dom:married_to`,  $X$ ), which would result on the input factoid being correctly answered as `dom:Person_B`. For a detailed description of the SPARQL generation process the reader is directed to [DAC12], which AUDiaL’s domain ontology handler extends.

## 6.3 Spatial Indexing and Navigation

Given that the consumption of diagrams is mostly concerned with extracting new information from it, we will consider means for enabling blind persons to perform exploration and quest tasks on semantically-enhanced diagrams (c.f. Section 6.1.1). The wayfinding framework of Allen describes three main means for accomplishing these tasks. Namely [All99]:

1. *Piloting* involves landmark-based navigation with a final destination in mind, in which progress is measured by the position of the traveler in the ordered sequence of landmarks between the start and end locations. Once situated in a known landmark, travel emphasis is placed on achieving the next one, with a certain degree of variability. Piloting is the most commonly employed means for spatial navigation.

2. *Path integration* consists in “updating one’s current location with reference to a point of origin, relying on information about movement away from that point” [All99]. The path undertaken in a path integration is a novel one, in which the recollected information enables the traveler to return to the point of origin when desired. Progress towards the destination must therefore be defined according to heuristics and thus it is difficult to assess.
3. *Cognitive map-based navigation* is “a knowledge-based means of wayfinding in which the traveler refers to an internal representation of a set of systematically related places in planning and navigating a route” [All99]. The internal representation of the environment i.e. the cognitive map is developed over the course of many encounters with the environment. This allows the traveler to infer the distance and location from the current location to a desired one. The main characteristic of this means of wayfinding is its flexibility, but it requires an already established deep knowledge of the environment.

The accessibility features of AUDiaL that enable these wayfinding means to be tackled in a non-visual manner are grounded on three general mechanisms included in the Natural Language Interface (NLI) prototype. Namely, *node indexing*, *object annotation*, and *home nodes*. Let us now expand on these elements.

### 6.3.1 Indexing of Graphic Objects

Locational and two-dimensional indexing (c.f. Section 6.1.1) is bolstered in AUDiaL by automatically adding a *navigational order* to the graphic objects that make up a semantically-enhanced diagram. The NLI prototype employs a datatype property at the user/system level (c.f. Section 3.10) to keep track of the ordering of its elements, namely `adl:hasOrder`. Having a navigational order for Graphic Objects enables the wayfinding means previously discussed by facilitating users to sequentially guide along the informational elements of a diagram. The automatic ordering of elements depends on the specific diagram at hand, although it follows a similar recipe in all statistical charts, as follows. Firstly, non-reference information objects are singled out. Then, the topmost Composite Graphic Objects among them are chosen and temporarily sorted according to their *x* or *y* coordinate on the SVG canvas. The chosen coordinate depends on the orientation of their corresponding metric axis. For a vertically-oriented axis, objects are sorted according to their *x* coordinate. Conversely, elements relating to a horizontally-oriented axis are sorted vertically. If both coordinates are meaningful, the vertical coordinate takes precedence. Then, for each object in order, its constituent elements are retrieved, spatially ordered in a similar fashion, and added to the sorting rank right after their parent. This process is repeated until the bottom-most Elementary Graphic Objects are reached.

Algorithm 6.1 shows simple, domain-specific pseudo-code for computing the navigational order of the bars of a stacked bar chart following the method we have just described. The method shown thus corresponds to how AUDiaL would process a `vto:Reset_Task` task

(VTO tasks are described in Section 3.9) on a stacked bar chart with a single metric axis. The navigational order of a Graphic Object additionally doubles as its *unique identifier (UID)*, since two Graphic Objects may not share the same navigational order. This moreover parallels the identification of Graphic Objects in visually displayed diagrams, since sighted persons differentiate between them by their location on the two-dimensional display.

---

**Algorithm 6.1:** Compute the navigational order of bars in a semantically-enhanced stacked bar chart

---

```

1 function compute_navigation (graphic);
  Input : A semantically-enhanced graphic (stacked bar chart): graphic
2 sorted  $\leftarrow$  array();
3 axis  $\leftarrow$  get_metric_axis(graphic);
4 stacked  $\leftarrow$  get_stacked_bars(graphic);
5 ax_or  $\leftarrow$  get_orientation(axis);
6 coor  $\leftarrow$  uvo:hasYCoordinate;
7 if ax_or == uvo:Vertical_Orientation then
8   | coor  $\leftarrow$  uvo:hasXCoordinate;
9 end
10 sorted_stacked  $\leftarrow$  sort(stacked, key_property = coor);
11 foreach stacked_bar  $\in$  sorted_stacked do
12   | sorted.add(stacked_bar);
13   | metric_bars  $\leftarrow$  get_constituent_bars(stacked_bar);
14   | metric_sorted  $\leftarrow$  sort(metric_bars, key_property = coor);
15   | foreach metric_bar  $\in$  metric_sorted do
16     |   | sorted.add(metric_bar);
17   | end
18 end
19 i  $\leftarrow$  1;
20 foreach bar  $\in$  sorted do
21   | graphic.add_triple(bar, adl:hasOrder, i);
22   | i  $\leftarrow$  i + 1;
23 end
```

---

AUDiaL does not explicitly communicate the navigational order (or, for that matter, any other location information) to its users, since such information by itself has been found to be of little use for blind persons accessing diagram e.g. when spoken out as coordinate positions [Ben02]. However, given that position information may be a significant factor in combination with other factors in easing the learning of a non-visual diagram [Ben02], users may explicitly ask for the coordinates (as given by the uvo:hasXCoordinate and uvo:hasYCoordinate datatype properties of UVO) or navigational order (adl:hasOrder) associated with a given Graphical Object. However, during validation of the prototype (Section 6.9) such inquiries did not appear once, which reinforces the ineffectuality of explicitly displaying location information in an accessible NLI to diagrams.

Nodes are automatically indexed by AUDiaL the first time a semantically-enhanced diagram is loaded onto its backend. In addition to calculating the navigational order of elements, AUDiaL will reset its *navigational pointer* to the first element of the sequence. This pointer is an structural descriptor at the system level, namely a `adl:isCurrentNode` datatype property occurrence, that keeps track of which Graphic Object the user is currently visiting. Alternatively, AUDiaL may keep this information in the user's Web session instead of adding it to the knowledge base in order to enable more than one user to access a single semantically-enhanced graphic simultaneously.

### 6.3.2 Non-visual Wayfinding

Wayfinding (i.e. navigational) tasks that enable users to move from one node to another non-visually may then be undertaken by issuing simple commands into the input prompt of the user interface. In addition, users may employ the functionality of AUDiaL's navigation modal panel, which has been included as an integral part of its GUI, shown in Figure 6.2, in order to assist them in remembering the wayfinding tasks available. The most basic of these tasks consists in the sequential exploration either by executing `vto:Move_Right_Task` tasks one after another (forward direction) or by executing `vto:Move_Left_Task` tasks (backwards direction). Users may also move up (`vto:Move_Up_Task`) or down (`vto:Move_Down_Task`) in the node hierarchy, and perform quick jumps to the first (`vto:Go_to_First_Task`) and last nodes (`vto:Go_to_Last_Task`) of the sequence. Specifically, these tasks are processed by AUDiaL in the following manner:

- Go to next/previous node: the user goes from the current Graphic Object to the next/previous object (of the same syntactic role e.g. `vdo:Stacked_Bar_SR`), according to the navigational order of the diagram's elements. For example, moving next from a Stacked Bar would land the user in the Stacked Bar that follows the current one in the navigational order of the diagram's elements, as shown in Figure 6.4. Going to the previous node from the first one, or, likewise, moving to the next node from the last one, has no effect.
- Go to first/last node: the user jumps from the current Graphic Object to the first/last object (of the same syntactic role) according to the navigational order of the diagram's elements.
- Move up/down: the user moves from the current Graphic Object to the Composite Graphic Object wherein it is contained (when moving up) or to the first (according to navigational order) Graphic Object contained therein (when moving down). For example, moving down a simple Stacked Bar object would land the user in its first constituent Metric Bar object. Moving up a topmost Graphic Object or down a bottom-most Graphic Object yields no effects, and the user is informed that a navigational end has been reached.

Given the frequency with which blind users will employ navigational commands for performing wayfinding tasks on diagrams, as well as other directives commonly employed during diagram

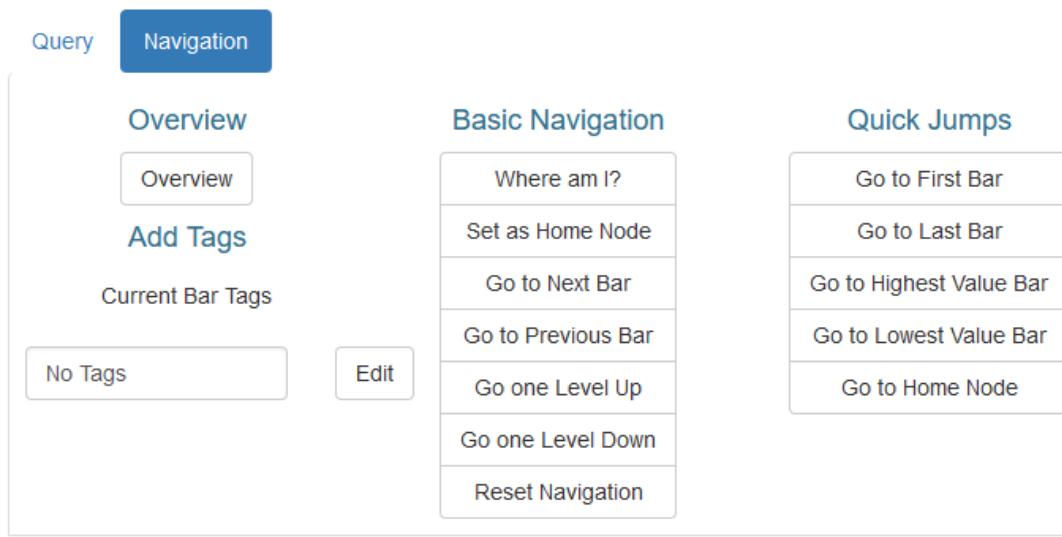


Figure 6.2: AUDiaL’s navigation modal panel. Available functions may be also executed by means of natural language commands in the query modal panel of the GUI.

exploration that are discussed later in this chapter (such as requesting a summary or annotating a graphic object), AUDiaL implements a `CommandParser` module that aims to speed up their execution in order to improve user satisfaction and task efficiency (c.f. quality requirement QR3 outlined in Section 5.4). This module, portrayed in Figure 6.3, maintains a list of common verbalizations for available commands characterized as regular expressions that may match a user’s input query in natural language. In addition, quick commands expressed as a single word or abbreviation (e.g. “next”, “up”, “overview”), are immediately recognized. If the user’s input matches any of these commands or expressions, its corresponding task is immediately executed against the semantically-enhanced graphic, thereby bypassing the Natural Language Processing (NLP) pipeline described in the previous chapter, which results in much faster query resolution times.

The mechanisms for sequential diagram wayfinding we have described are especially useful for non-visual exploration and quest tasks (c.f. Section 6.1.1) on diagrams. However, simply moving from one node to another and displaying information about the current node is not a sufficient means for achieving the wayfinding means described earlier in this section. Important requirements for these tasks to be achievable is that diagram nodes can be uniquely identified, easily told apart from each other, and important landmarks must be recognizable [Bro08]. These requirements arise from the need of supporting computational off-loading that lowers the memory demands of users reading diagrams, as otherwise the high mental demands for the user would likely lead to disorientation and frustration. AUDiaL tackles these necessities by employing a number of techniques based on system- and user-initiated annotation of elements with useful metadata. These methods are discussed in the following section.

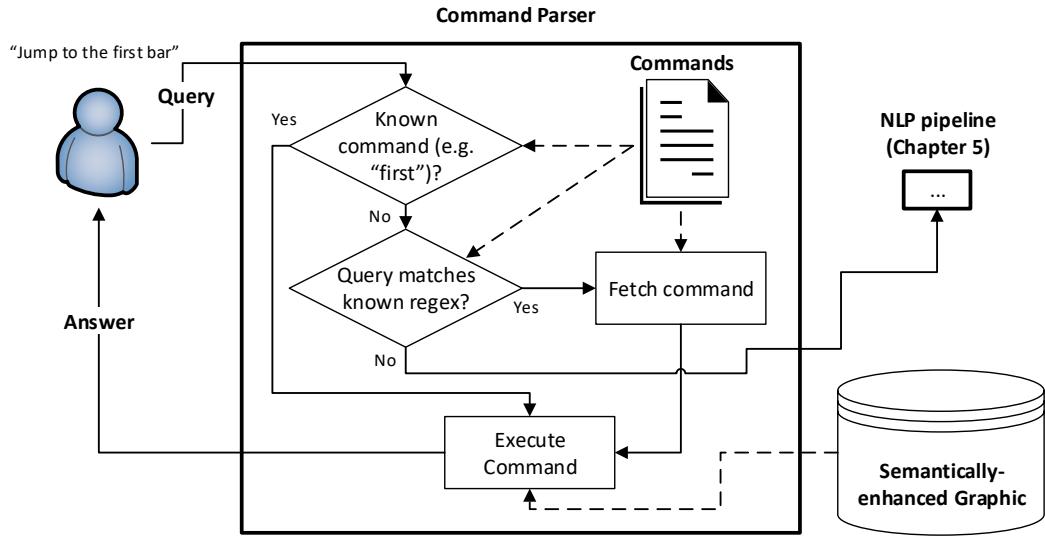


Figure 6.3: Command Parser module of AUDiaL. A known (navigational) command issued by the user will bypass the standard NLP pipeline.

## 6.4 Object Annotation

Previous studies by Brown and Stevens have shown that, in the context of computer-based access to mathematical graphs:

"Annotations can be designed to replace some of the benefits imparted by visual presentation of graph-based diagrams, including making implicit information explicit, grouping related items, interactivity and acting as an external memory, and to reduce disorientation while moving around the graph. A graph which is annotated in such a way requires less mental effort for a visually impaired user to explore than one which is not. Tasks can be achieved more effectively, efficiently and with more satisfaction through use of annotation to replace features of a visual presentation" [BSP13].

Stemming from this and other research in annotation of diagrams, annotation of informational Graphic Objects with metadata is the main means employed by AUDiaL for overcoming the lack of cognitive computational off-loading of its (blind) users.

In the context of this thesis, *object annotation* consists in enabling users to add personalized text notes to certain elements of a diagram. In AUDiaL, users may add an annotation to the current graphic object being visited during a travel task performed on a semantically-enhanced diagram. For instance, a user may choose, in the course of an exploration activity, to add a personal note to a bar in a bar chart in order to use it as a landmark for piloting later on. Annotations in AUDiaL may be added to an element by means of a verbal command e.g. "add tag 'hometown'" or by employing the mechanism provided by the navigation mode of

the user interface, see Figure 6.2. From that moment onward, each time a user lands on said element during wayfinding, or when the element is part of a more complex query's answer (Chapter 5), its personal annotations will be displayed along with the rest of the answer.

A straightforward benefit of allowing users to annotate individual graphic objects with personal text is that this mechanism mimics the readily available external memory that diagrams provide to sighted readers. As showcased by the example answers of Figure 6.4 and Figure 6.6, object annotations input by the user for a given element are included to that element's description each time it appears in an answer or navigation output. In this manner, the short-term memory of the user can be freed up with annotations that help him or her recall or retrieve important landmarks during piloting or cognitive map-based navigation activities. If a user becomes lost during exploration or wishes to return to a known landmark for some other reason, he or she may issue a "go to <annotation>" command or similar in order to immediately jump to the graphic object assigned to that annotation. The ability to quick jump to annotated elements mimics the instant accessibility of the visual form by reducing short-term memory demands, facilitating obtaining an overview of the data, and facilitating external cognition for AUDiaL's users. AUDiaL also implements VTO's vto:Where\_Task task, which gives the user information about his or her current position during a wayfinding task in order to help with keeping orientation. This command is generally executed by issuing the prompt "where am I?", or a similar query.

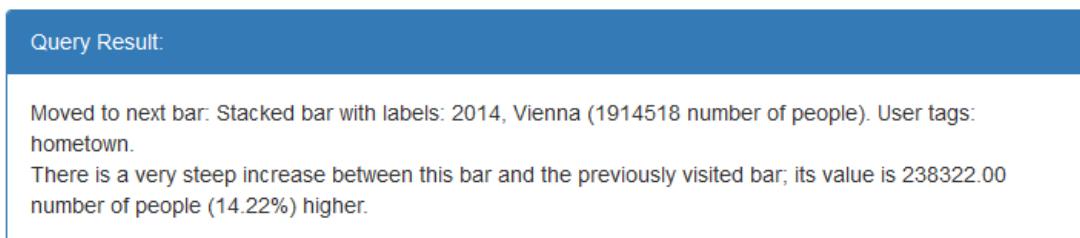


Figure 6.4: Example output of a sequential navigation between objects in AUDiaL. The attributes of the object the user has landed on are given first, followed by any personalized annotations. Lastly, a comparison with the previously visited object is presented.

Object annotation may also be used to prevent confusion caused by poor differentiation of similar Graphic Objects. It has been shown that improving element differentiation in graphs helps overcome the general problems of disorientation and makes some elements more suitable as wayfinding landmarks [Bro08]. For instance, a travelling user stumbling upon a bar with the same value as a previously visited one may choose to label the second bar with a unique annotation in order to tell it apart should he or she encounter it again in the future. Even if most informational Graphic Objects will be associated to their unique set of labels and values, thereby uniquely identifying them when output as part of a system's answer, enabling user to add personal notes speeds up element differentiation significantly, which in turn results in more efficient wayfinding and performance of analytical activities on diagrams.

According to Larkin and Simon, the single-most important benefit of diagrams is that they

make explicit features that would otherwise be implicit [LS87]. Object annotation can help preserve this benefit to some extent for blind persons accessing diagrammatic representations of data. For example, the benefits of clustering in graphs can be preserved by enabling hierarchical browsing of nodes and connection-based browsing [BPS03]. As described in Section 6.3.2, AUDiaL supports hierarchical-based navigation of Composite Graphic Objects by means of natural language-enabled wayfinding commands. Likewise, connection-based browsing is built upon the concept of home nodes, which will be discussed next.

## 6.5 Home Nodes

A *home node* is a distinctive node (i.e. Graphic Object), analogous to a landmark used in real-world spatial navigation, that is used as base for exploration and that can be easily returned to when lost [Bro08]. Home nodes are the most significant feature of AUDiaL for enabling users to perform wayfinding tasks supported by path integration (c.f. Section 6.3).

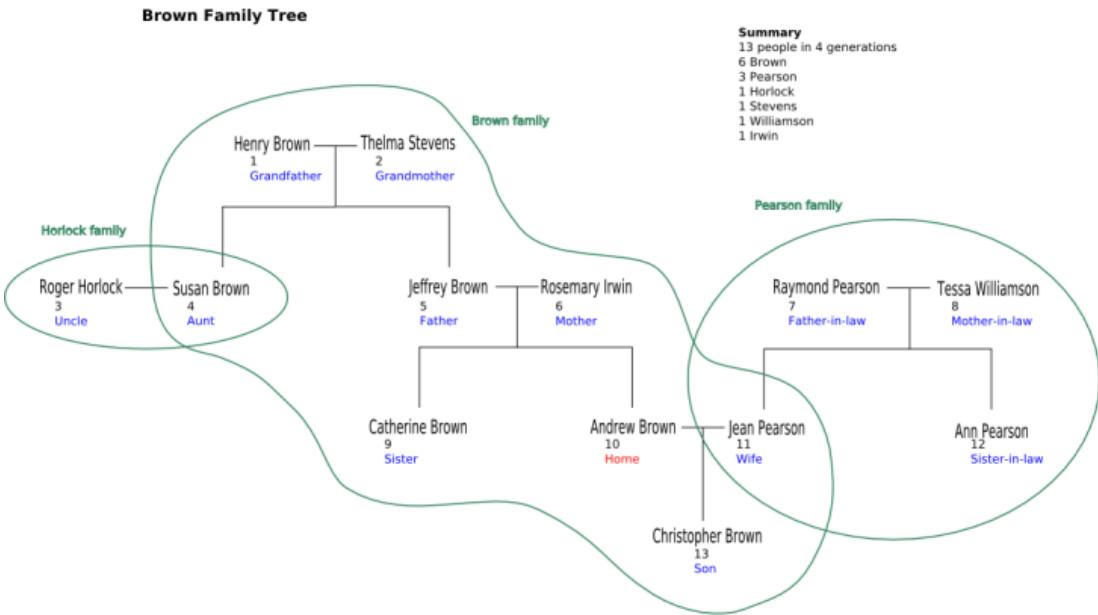


Figure 6.5: Semantic annotation of a family tree (a special case of link diagram) from Brown et al. [BSP13]. Nodes belonging to certain clusters ('families') are given special annotations. In addition, a *home node* ('Andrew Brown') is selected, and the implicit relationships of each node with respect to the home node are made explicit via annotations.

Home nodes in accessibility have been mainly investigated in the context of non-visual navigation of mathematical graphs [BSP13, BPS03]. In this particular scenario, home nodes allow distant relationships between nodes to be made explicit by annotating the relationship of all nodes of the graph with respect to the home node, see Figure 6.5. By enabling users

to discover how the current node they are visiting relate to their home node, and to travel back to their home node at any time, their memory demands are lowered their orientation is more easily retained, thereby making graph exploration more efficient [Bro08]. The relation between nodes in a graph is dependent on its domain of knowledge; for example, in family trees, this relationship can be expressed as a natural relationship (e.g. 'grandson', 'cousin') between the persons each node symbolizes. A more domain-agnostic approach to express node relationship is to describe it in terms of number of arcs distance [BSP13]. In the following, we will lay our focus on establishing how such an approach can be paralleled in the context of non-visual access to statistical charts for its inclusion in AUDiaL.

Query Result:

The average is 1762008.00 number of people  
 The result is 5.12% bigger than the home bar's value  
 The following 3 bars have been considered:

- Stacked bar with labels: 1994, Vienna (1695310 number of people). This is the current bar.
- Stacked bar with labels: 2004, Vienna (1676196 number of people). This is the home bar.
- Stacked bar with labels: 2014, Vienna (1914518 number of people). User tags: hometown last year.

Query
Navigation

Retrieve: what is the average population of vienna?

Submit Query

Figure 6.6: The answer to an input query showing several annotation techniques in use. First, the answer to the question itself is given. Numeric answers are then compared to the associated value of the home node, if set. For each individual Graphic Object that has been considered as part of the answer, their individual user/system annotations are additionally given as part of the answer.

Recall from Section 6.3 that path integration has two main components: (1) a 'reference' point of origin chosen by the user i.e. the home node, and (2) updating one's current location with reference to the reference point. In AUDiaL, a user may choose at any time to select the current Graphic Object as his or her home node, which from that moment onward will act as the main reference point for subsequent path integration. The user's current home node is then stored into the system via a session variable (for simultaneous, user-specific home node integration) or by employing `adl:isHomeNode` datatype property occurrences at the system/user semantic level (c.f. Section 3.10). Any future system answer will then be related to the home node's associated value, as exemplified in Figure 6.6. This includes not only the results from navigational tasks, but also answers to analytical tasks resulting on a single

numerical value such as computing the average value from a set of informational Graphic Objects.

The comparison between the current Graphic Object's value and the home node's value is given in relative terms, since the aim of the home node is to serve as a reference point against which every other element of the graphic can be contrasted. Home nodes make explicit vital information that would otherwise be lost by simply analyzing the raw tabular counterpart of a diagram. For example, comparing the sizes (as a proxy of value) of two bars in their visual form is done instinctively and almost instantly by sighted diagram readers. This comparison is explicitly recovered in AUDiaL by enabling users to choose their 'home bar' in a semantically-enhanced bar chart.

Given that in statistical charts the actual distance (or *node-arc distance*) between elements does not carry any additional semantics that cannot be inferred from the elements' labels or values with respect some axis, such information is not given as part of the answer as doing otherwise would only clutter it with irrelevant information for its users. In addition to selecting their home node, users may choose to jump back to it node at any time by issuing a "go to home" or similar command, an essential feature to enable any kind of high-level path integration task. Lastly, each time a user encounters his or her home node during navigation or as part of a system answer, this fact is highlighted, as seen in Figure 6.6, in order to aid wayfinding and preventing lostness (by e.g. obviously entering a navigational loop).

## 6.6 High-level Summaries

A quick glance at a diagram is the first stage in many everyday graphic reading tasks; by doing so, sighted readers obtain a broader understanding of the nature of the diagram and, consequently, the nature of the tasks that can be undertaken [Bro08]. Such a general peek at the diagram as a whole, not particularly aimed at any of its components, gives readers an idea about the complexity of the problem so they may start developing strategies for reaching a deeper understanding. In addition, readers may return to consulting a general picture of a diagram in order to orientate themselves while navigating it. Likewise, persons reading geographical maps (e.g. city maps) often get an idea of how areas are organized by a fast exploration of the map which enables them to get an abstract overview of the location, size, and shape of important geographical objects before exploring an unknown area [HWB06]. In Human-Computer Interaction (HCI) research, the importance of data summaries is made evident by Shneiderman's *Visual Information-Seeking mantra*, which, as previously introduced in this chapter, states: "overview first, zoom and filter, then details-on-demand" [Shn96]. This well-known and widely adopted principle is, according to Shneiderman, the most fundamental convention for designing effective computer interfaces for information exploration. It is not a coincidence that the first thing that should be offered to users according to the VIS mantra is an overview of the information space about to be explored.

### 6.6.1 Summaries in Accessibility

In the field of accessibility of visual information, overviews and summaries are considered a vital part of many, if not most, existing approaches. For example, the Visual Information-Seeking mantra has been adopted in sonification research as the *Auditory Information Seeking Principle (AISP)*. This guideline proposes dividing auditory data querying into four phases, the first of which (named *gist*) consists in a short auditory message describing an overview of a data series [WB06]. High-level summaries of real-world photographs are also offered by Communicative Images [KO11] to their users. Such an overview consists of a brief description of the objects in the image's foreground, followed by a short narration of the image's background objects and their positions [KO08a]. Offering summaries of a tactile diagram has been shown to be of benefit by Yu et al., who state that "an effective way of presenting the overview of [a] line graph will [...] give blind users a better understanding about the graph" [YRB01]. The lack of a readily available overview is noted by Asakawa and Leporini as "the greatest problem encountered when explaining how to use an application and a screen reader", which forces blind users to "memorize and mentally track more information than a sighted user" [AL09].

Compensating for the lack of the cognitive aids stemming from quick glancing at a diagram is a significant challenge in designing an accessible a Natural Language Interface (NLI), since high-level overviews of visualization expressed in text will certainly lose some important information that can only be observed by visual means. In addition, the sequential nature of text makes it hard for a human reader to understand the concepts being explained due to likely confusion and short-term memory overload when diagrams are described verbally in detail [BBR<sup>+</sup>03]. More specifically, static high-level descriptions of statistical charts in natural language have been shown to be very useful albeit not sufficient means for communicating diagram contents to blind users, with inference being a requirement to facilitate exploration of the information space [FLST13]. Nevertheless, given the extensive amount of evidence on the benefits of high-level summaries for disparate graphic domains, non-visual hierarchical access to diagrammatically displayed information in AUDiaL ought to start with a general overview of the data and how it is visually represented.

In designing an effective summary, the following attributes must be observed [BBR<sup>+</sup>03]:

- Summaries must be kept *short* both in absolute and relative terms. The goal is to summarize, not to re-phrase the contents.
- Users may have access to the summary at any time by requesting it. The benefits of having readily *accessible* summaries are twofold. First, they serve to refresh the user's working memory when needed; second, summaries may aid with orientation during wayfinding activities.
- Summaries must be *predictable*, this is, they should be given in a consistent order every time as to not confuse users and to enable them to build a 'skeleton' mental representation into which specific diagram data later fits [FLST13]. Likewise, similar diagrams must be summarized in similar terms.

Lastly, evaluation by Ferres et al. resulted in the following recommendation for items of information to be reported when summarizing statistical charts for both blind and sighted persons [FLST13]:

1. Diagram title.
2. Type of diagram e.g. bar chart or pie chart.
3. Purpose of the diagram
4. Title of  $x$  and  $y$  axes.
5. Units of measurement used in axes.
6. In bar charts, total number of bars and whether stacked bars are present.
7. General shape of the diagram e.g. “rising from left to right” or “almost horizontal”.
8. General trend of the diagram.
9. Highest and lowest points of the diagram along with relevant context such as units of measurement.
10. Color of the axes, bars, and other relevant graphic objects of the diagram.

### 6.6.2 AUDiaL Summaries

The statistical chart summarization method included in AUDiaL has been developed accordingly with the previous considerations in mind. Some further guidelines for non-visual description of charts we have considered can be consulted in [CL99]. The implemented algorithm is outlined the pseudo-code shown in Algorithm 6.2 (with three of its methods additionally described in Algorithm 6.3, Algorithm 6.4, and Algorithm 6.5). Note that printing every item listed above would result in very long summaries, something that would contradict the required attribute of brevity previously discussed. Therefore, only the most relevant items above, concluded from informal empirical evaluations with two blind participants (one of them a mathematician) were selected for inclusion in AUDiaL’s diagram summaries.

In addition, the following considerations were carried out when implementing AUDiaL’s diagram summarizer. Diagram type and title are given together on the first line of the overview. Next, a description of the axes of the diagram (if any) is generated. Studies have shown that blind persons are confused by terms such as ‘ $x$  axis’ and ‘ $y$  axis’, and that it is preferable to use more concrete terms such as ‘vertical’ and ‘horizontal’ [FLST13]. AUDiaL therefore describes a metric axis by first indicating what it measures (if the axis is properly labeled) and its orientation, as shown in Algorithm 6.3. If one or more legends are available in the semantically-enhanced graphic, a short description thereof is included afterwards (see lines 2-15 of Algorithm 6.4). This is followed by a list of all remaining labels not appearing in the diagram’s legend (Algorithm 6.4, lines 16-22).

---

**Algorithm 6.2:** Computing a natural language overview for a bar chart with a single metric axis.

---

```

1 function print_summary (graphic);
  Input : A semantically-enhanced graphic: graphic
2 type  $\leftarrow$  get_triple_object(graphic, vdo : is_graphic_representation);
3 print("This is a " + type + ".");
4 title_object  $\leftarrow$  get_triple_object(graphic, vdo : is_labeled_by);
5 title  $\leftarrow$  get_triple_object(title_object, uvo : has_text);
6 print("Its title is: " + title + "\n");
7 axis  $\leftarrow$  get_metric_axis(graphic);
8 print_axis_summary(axis);
9 print_labels_summary(graphic);
10 print_bars_summary(graphic);
11 print_shape_and_trend(graphic);
12 print_extrema(graphic);
13 print("End of summary");

```

---

**Algorithm 6.3:** Computing a natural language overview for a chart's metric axis.

---

```

1 function print_axis_summary (axis);
  Input : An axis graphic object instance: axis
2 if axis is NULL then
3   | print("There are no axes in this chart \n");
4 end
5 else
6   | units  $\leftarrow$  get_triple_object(axis, ddo : expresses_information);
7   | print("It main axis describes " + units + ".");
8   | ax_title_object  $\leftarrow$  get_triple_object(axis, vdo : is_labeled_by);
9   | ax_title  $\leftarrow$  get_triple_object(ax_title_object, uvo : has_text);
10  | print ("It has title: " + ax_title + ".");
11  | orientation  $\leftarrow$  get_triple_object(axis, uvo : has_orientation_named);
12  | print("It has " + orientation + "\n");
13 end

```

---

The next section of the summary is specific to the diagram type at hand. For example, bar charts are summarized by first listing the number of bars in the chart, divided into stacked bars, metric bars, and total number of bars. Each lineup of informational graphic objects (e.g. grouped bars) is listed next. The labels common to each lineup are included to the description of each item of the lineup (see Figure 6.7 for an example). A scatterplot can be described by the number of clusters and their attributes. More generally, informational graphic object groupings (through graphic relations such as lineup, clustering, or separation

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

by separator, see Section 3.7.1) ought to be individually described by their shared labels and meaningful attributes.

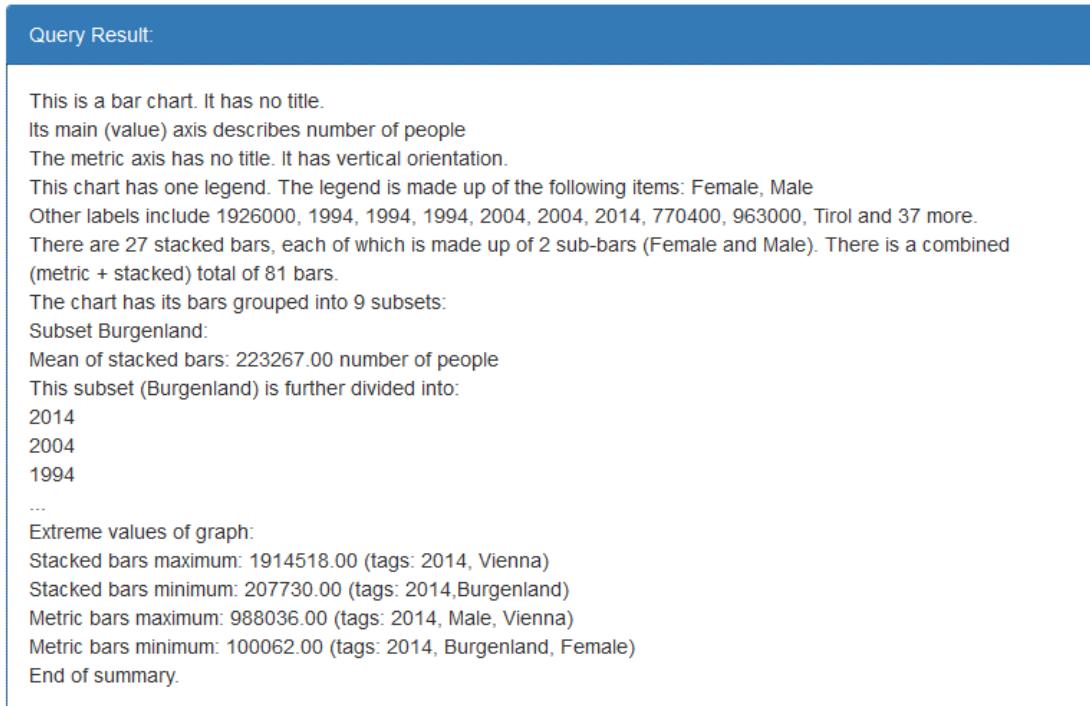


Figure 6.7: Adapted high-level overview (summary) of a stacked bar chart generated by AUDiaL.

The general shape and trend of the diagram are described next. This consists in a simple one-liner that gives readers a rudimentary idea of the overall trend (*increasing*, *decreasing*, *flat*) and geometrical shape of the diagram as a whole. Giving explicit shape names of trend lines (e.g. “like a W”) is avoided altogether, as this has been shown to confuse congenitally blind persons who rely on non-visual cues [FLST13]. The resulting shape summarized employed in AUDiaL may return one of the following five descriptions (examples depicted in Figure 6.8):

1. If all diagram values are within 10% of every other value, the diagram is said to have a *flat* shape. Otherwise:
2. If the first (according to navigational order, c.f. Section 6.3.1) element of the diagram, that we will name  $s$ , has a greater (10% tolerance) value than the last one, named  $e$ ; and no intermediate point has a value greater than  $s$  or smaller than  $e$ , then we say the diagram has a *falling* trend.
3. If element  $s$  has a smaller (10% tolerance) value than element  $e$ , and no intermediate point has a value smaller than  $s$  or greater than  $e$ , then the diagram is described as having a *raising* trend.

---

**Algorithm 6.4:** Computing a natural language summary of labels present in a diagram.

---

```

1 function print_labels_summary (graphic);
  Input : A semantically-enhanced graphic: graphic
2 legend_role  $\leftarrow$  vdo:Legend_Reference_Object_IR;
3 lg  $\leftarrow$  graphic.get_elements_with_role(legend_role);
4 if lg is not NULL then
5   | print("This chart has one legend. ");
6   | print("It is made up of the following elements:\n");
7   | lg_labels  $\leftarrow$  lg.get_elements_with_role(vdo:Label_SR);
8   | foreach label  $\in$  lg_labels do
9     |   | elm_text  $\leftarrow$  get_triple_object(label, uvo:has_text);
10    |   | print(elm_text + "\n");
11   | end
12 end
13 else
14   | print("This chart has no legends. \n");
15 end
16 all_labels  $\leftarrow$  graphic.get_elements_with_role(vdo:Label_SR);
17 remaining_labels  $\leftarrow$  all_labels \ lg_labels;
18 print("Other labels include: ");
19 foreach label  $\in$  remaining_labels do
20   |   | elm_text  $\leftarrow$  get_triple_object(label, uvo:has_text);
21   |   | print(elm_text + "\n");
22 end

```

---

4. If there is at least one element with value greater than both *s* and *e*, but no element with value smaller than both *s* and *e*, the diagram is said to have a *pyramid-like* shape.
5. If there is at least one element with value smaller than both *s* and *e*, but no element with value greater than both *s* and *e*, the diagram is said to have a *valley-like* shape
6. If the diagram does not adhere to any of the five shapes previously described, then it is said to have a *zig-zag* shape.

The description of pyramid-like, valley-like, and zig-zag shapes is further enlarged with an overall *trend description* of the depicted data. This trend is obtained by simple linear regression. The resulting regression slope is then interpreted as the ‘steepness’ of the visualized data and communicated to the user. For example, shape 6 of Figure 6.8 has a positive slope not close to zero; therefore, its shape and trend would be reported as follows: “the data follows a zig-zag shape with an overall steep increase”. Further details about the natural language description of the different degrees of steepness are given later in this section. If

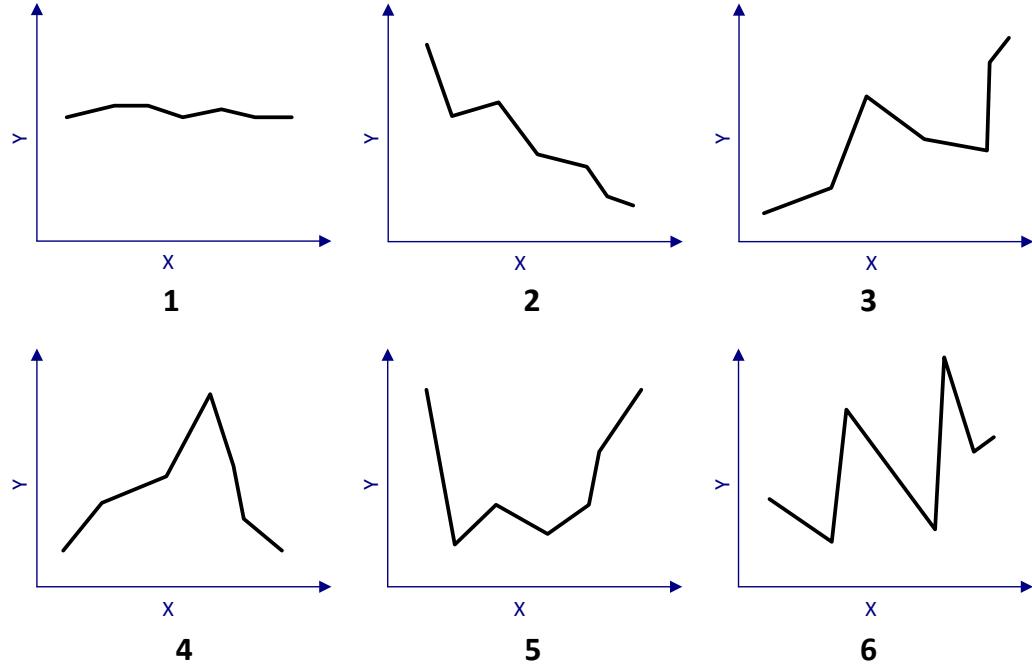


Figure 6.8: The six general diagram shapes considered in AUDiaL’s summaries: (1) flat, (2) falling, (3) raising, (4) pyramid, (5) valley, (6) zig-zag.

only one scalar variable is present in the diagram, the spatial coordinate along which the corresponding informational graphic objects are laid out is employed as the independent variable for linear regression. The regression slope is employed, as opposed to a dimensionless correlation coefficient, in order to preserve the visualization of its slope, which may change depending on how data are chosen to be depicted in a specific diagram.

Lastly, information about extreme values of the diagram is included in the summary. A varying number of extrema may be communicated depending on the diagram at hand, following a method, outlined in pseudo-code in Algorithm 6.5, derived from the diagram description guidelines of Corio and Lapalme [CL99]. Extreme values (along with the labels of their corresponding graphic objects) are divulged to the user when they carry useful information due to being distinctive in relation to their neighboring values.

These considerations result in informative and predictable overviews of a manageable size. With regard to the overview’s accessibility, users may invoke it at any time by issuing a “give me an overview of the graphic” command (or similar) in natural language, or by employing the ‘Overview’ button of AUDiaL’s navigation modal panel, shown in Figure 6.2.

A second kind of overview that AUDiaL offers to its users consists in short verbal descriptions of the visual shape of the (inferred) line segments between two selected data points during a

---

**Algorithm 6.5:** Computing a natural language description of a bar chart's extrema.

---

```

1 function print_extrema (graphic);
  Input : A semantically-enhanced graphic: graphic
2   bars  $\leftarrow$  graphic.get_elements_with_role(vdo:Stacked_Bar_SR);
3   if empty(bars) then
4     | bars  $\leftarrow$  graphic.get_elements_with_role(vdo:Metric_Bar_SR);
5   end
6   n_bars  $\leftarrow$  count(bars);
7   values  $\leftarrow$  sort_descending_according_to_value(bars);
8   printed  $\leftarrow$  FALSE;
9   if n_bars > 6 AND abs(values[2]/values[3]) > 1.2 then
10    | print("The three largest values are: \n");
11    | print_labels_and_values(bars[0], bars[1], bars[2]);
12    | printed  $\leftarrow$  TRUE;
13  end
14  else if n_bars > 4 AND abs(values[1]/values[2]) > 1.15 then
15    | print("The two largest values are: \n");
16    | print_labels_and_values(bars[0], bars[1]);
17    | printed  $\leftarrow$  TRUE;
18  end
19  else if n_bars > 2 AND abs(values[0]/values[1]) > 1.1 then
20    | print("The largest value is: \n");
21    | print_labels_and_values(bars[0]);
22    | printed  $\leftarrow$  TRUE;
23  end
24  if n_bars > 2 AND abs(values[-2]/values[-1]) > 1.3 then
25    | print("The smallest value is: \n");
26    | print_labels_and_values(bars[-1]);
27    | printed  $\leftarrow$  TRUE;
28  end
29  if printed is FALSE then
30    | print("The largest value is: \n");
31    | print_labels_and_values(bars[0]);
32  end

```

---

wayfinding activity, which contributes to preserving the diagram's literal resemblance of its slopes and trends to a finer granularity than the overall data trend described in its high-level summary. Two examples are given in Figure 6.4 and Figure 6.9. This partial trend overview is more detailed than the overall diagram trend, and corresponds to some extent to the 'zoom' aspect of the Visual Information-Seeking mantra.

This kind of overview is offered to users in bar charts and line charts when navigating between two data points. First, the positions of the start and end data points in the canvas (e.g. the topmost middle point of their corresponding metric bars) are computed, as well as the positions of any intermediate points, according to navigational order, between start and end positions. Then, for each intermediate line segment between points (whether it explicitly exists or not), its slope in degrees is computed. Changes of less than 5% in slope between line segments are considered as a continuity of the previous trend. Otherwise, slope changes between intermediate segments are communicated to the user. Slope degrees (in absolute value) are converted to brief descriptions as follows:

- $0^\circ \leq \text{slope} < 2^\circ$ : *flat course*.
- $2^\circ \leq \text{slope} < 5^\circ$ : *almost flat course*.
- $5^\circ \leq \text{slope} < 15^\circ$ : *slow increase or slow decrease*.
- $15^\circ \leq \text{slope} < 30^\circ$ : *moderate increase or moderate decrease*.
- $30^\circ \leq \text{slope} < 45^\circ$ : *sharp increase or sharp decrease*.
- $45^\circ \leq \text{slope} < 75^\circ$ : *very steep increase or very steep decrease*.
- $\text{slope} \geq 75^\circ$ : *extremely steep increase or extremely steep decrease*.

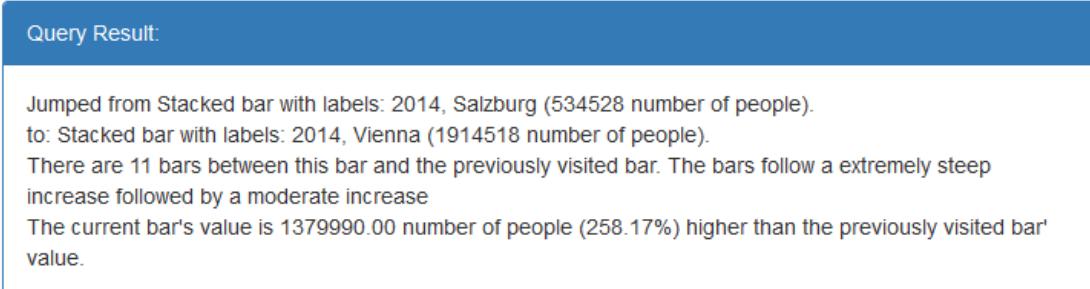


Figure 6.9: Example output of a ‘jump’ navigational command between distant objects in AUDiaL. A high-level summary of the visual graphic shape between intermediate data points is offered, followed by a direct comparison between start and end data points.

A maximum total of 10 trend changes are displayed in a summary. If there are more than 10 trend changes between starting and end points, only the ten most significant ones (by magnitude of change) are shown. Moreover, monotonically increasing or decreasing trend changes are combined and marked as “accelerating” or “decelerating”, respectively (e.g. “a very steep accelerating increase”). These trend changes are concatenated and displayed after the main answer to the navigational command issued by the user. In addition, the values of the starting and end data points are compared in absolute and relative terms. Figure 6.9 shows an example of a trend summary between distant data points.

## 6.7 Review of HCI Techniques for Accessibility

The combination of high-level summaries, navigation summaries, and low-level analytical and wayfinding tasks that have been described in the previous sections ensures that the three aspects of the Visual Information-Seeking mantra; namely (1) overview, (2) zoom-and-filter, and (3) details-on-demand, are covered by AUDiaL's functionality while being fully actionable by means of natural language. In addition, we have covered a number of annotation and interaction techniques included in AUDiaL with the aim to compensate for the lack of cognitive benefits related to visualization that blind persons endure (c.f. Section 6.1.1). These methods are briefly summarized in Table 6.1 according to the cognitive benefits they help to compensate for.

		Hierarchical and Sequential Navigation	Quick Jumps	Object Annotation	Home Node	High-level summary/ Trend description
Wayfinding	Piloting Path Integration	✓	✓	✓		
	Cognitive map-based navigation	✓	✓		✓	
				✓		
Resemblance preservation	Literal			✓		
	Homomorphic	✓				✓
Computational off-loading		✓		✓	✓	✓
Breadth-then-depth search		✓	✓		✓	✓

Table 6.1: Summary of techniques included in AUDiaL that aim to compensate for the lack of certain cognitive benefits in non-visual access to diagrams.

## 6.8 Web Interface Accessibility Evaluation

In this section the main Web Interface of AUDiaL will be evaluated for accessibility according to the success criteria of the Web Content Accessibility Guidelines (WCAG) version 2.1 (see Section 2.7.1 for an overview of Web accessibility guidelines). These guidelines lay their focus on ensuring that Web developers following them create web pages that can be accessed by persons with disabilities using standard assistive technology such as screen readers.

WCAG 2.1 consists of twelve guidelines grouped into four principles; namely, principles state that the content must be (1) *Perceivable*, (2) *Operable*, (3) *Understandable*, and (4) *Robust*.

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

Each guideline has several *success criteria* which address important issues for people with disabilities beyond the usability problems that might be faced by all users. In turn, each success criterion has a *conformance level*: A, AA, or AAA. Level A sets a minimum level of essential accessibility without achieving broad accessibility of the content. At the other end of the scale, level AAA is generally too strict, which means that it is not possible to satisfy all of its success criteria for some content without setting unreasonable expectations on the content creators. Higher conformance levels cover conformance at the levels below it. For this reason, AA conformance is generally the recommended choice for Web-based information. We therefore decided upon complying to the latest WCAG criteria at the AA level of conformance for AUDiaL.

### 6.8.1 Success Criteria Overview

Table 6.2 outlines the criteria taken into account during AUDiaL's design, their conformance level, and whether the current version of the prototype meets said criteria. Additional information on how guidelines are tackled by the prototype is given in Section 6.8.2.

Criteria	Level	Evaluation
<b>Principle 1: Perceivable - Information and user interface components must be presentable to users in ways they can perceive</b>		
<b>Guideline 1.1 – Text Alternatives:</b> Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language		
1.1.1 Non-text Content. All non-text content that is presented to the user has a text alternative that serves the equivalent purpose.	A	Passed
<b>Guideline 1.2 – Time-based Media:</b> Provide alternatives for time-based media.		
Not applicable: no time-based media (video, audio) is provided by AUDiaL.		
<b>Guideline 1.3 – Adaptable:</b> Create content that can be presented in different ways (for example simpler layout) without losing information or structure.		
1.3.1 Info and Relationships. Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.	A	Passed
1.3.2 Meaningful Sequence. When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined.	A	Passed
1.3.3 Sensory Characteristics. Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.	A	Passed

Table 6.2: Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1

Criteria	Level	Evaluation
1.3.4 Orientation. Content does not restrict its view and operation to a single display orientation, such as portrait or landscape.	AA	Passed
1.3.5 Identify Input Purpose. The purpose of each input field collecting information about the user can be programmatically determined [...]	AA	Passed
<b>Guideline 1.4 – Distinguishable:</b> Make it easier for users to see and hear content including separating foreground from background.		
1.4.1 Use of Color. Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.	A	Passed
1.4.3 Contrast (Minimum). The visual presentation of text and images of text has a contrast ratio of at least 4.5:1 [...]	AA	Passed
1.4.4 Resize text. Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.	AA	Passed
1.4.5 Images of Text. If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text [...]	AA	Passed
1.4.10 Reflow. Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions [for vertical and horizontal scrolling at a given width].	AA	Passed
1.4.11 Non-text Contrast. The visual presentation of [user interface components and graphical objects] have a contrast ratio of at least 3:1 against adjacent color(s).	AA	Passed
1.4.12 Text Spacing. In content implemented using markup languages that support [...] text style properties, no loss of content or functionality occurs by [changing text spacing] and by changing no other style property.	AA	Passed
1.4.13 Content on Hover or Focus. Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true: [...]	AA	Passed
<b>Principle 2: Operable - User interface components and navigation must be operable.</b>		
<b>Guideline 2.1 – Keyboard Accessible:</b> Make all functionality available from a keyboard.		

Table 6.2: Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

Criteria	Level	Evaluation
2.1.1 Keyboard. All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.	A	Passed
2.1.2 No Keyboard Trap. If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.	A	Passed
2.1.4 Character Key Shortcuts. If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true: [...]	A	Passed
<b>Guideline 2.2 – Enough Time:</b> Provide users enough time to read and use content.		
Not applicable: no time-based media (video, audio) is provided in AUDiaL. Users may take as long as they wish to navigate the front-end, read system prompts and input queries.		
<b>Guideline 2.3 – Seizures and Physical Reactions:</b> Do not design content in a way that is known to cause seizures or physical reactions.		
2.3.1 Three Flashes or Below Threshold. Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds.	A	Passed
<b>Guideline 2.4 – Navigable:</b> Provide ways to help users navigate, find content, and determine where they are.		
2.4.1 Bypass Blocks. A mechanism is available to bypass blocks of content that are repeated on multiple Web pages.	A	Passed
2.4.2 Page Titled. Web pages have titles that describe topic or purpose.	A	Passed
2.4.3 Focus Order. If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.	A	Passed

Table 6.2: Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1

Criteria	Level	Evaluation
2.4.4 Link Purpose (In Context). The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.	A	Passed
2.4.5 Multiple Ways. More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process.	AA	Passed
2.4.6 Headings and Labels. Headings and labels describe topic or purpose.	AA	Passed
2.4.7 Focus Visible. Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.	AA	Passed
<b>Guideline 2.5 – Input Modalities:</b> Make it easier for users to operate functionality through various inputs beyond keyboard.		
2.5.1 Pointer Gestures. All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.	A	Passed
2.5.2 Pointer Cancellation. For functionality that can be operated using a single pointer, at least one of the following is true: (1) no down-event, (2) the function can be aborted or undone before completion, (3) up-reversal, and (4) completing the function on the down-event is essential.	A	Passed
2.5.3 Label in Name. For user interface components with labels that include text or images of text, the name contains the text that is presented visually.	A	Passed
2.5.4 Motion Actuation. Functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation [...]	A	Passed
<b>Principle 3: Understandable - Information and the operation of the user interface must be understandable.</b>		
<b>Guideline 3.1 – Readable:</b> Make text content readable and understandable.		
3.1.1 Language of Page. The default human language of each Web page can be programmatically determined.	A	Passed
3.1.2 Language of Parts. The human language of each passage or phrase in the content can be programmatically determined [...]	AA	Passed

Table 6.2: Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

Criteria	Level	Evaluation
<b>Guideline 3.2 – Predictable:</b> Make Web pages appear and operate in predictable ways.		
3.2.1 On Focus. When any user interface component receives focus, it does not initiate a change of context.	A	Passed
3.2.2 On Input. Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component.	A	Passed
3.2.3 Consistent Navigation. Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user.	AA	Passed
3.2.4 Consistent Identification. Components that have the same functionality within a set of Web pages are identified consistently.	AA	Passed
<b>Guideline 3.3 – Input Assistance:</b> Help users avoid and correct mistakes.		
3.3.1 Error Identification. If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.	A	Passed
3.3.2 Labels or Instructions. Labels or instructions are provided when content requires user input.	A	Passed
3.3.3 Error Suggestion. If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user [...]	AA	Passed
<b>Principle 4: Robust – Content must be robust enough that it can be interpreted by a wide variety of user agents, including assistive technologies.</b>		
<b>Guideline 4.1 – Compatible:</b> Maximize compatibility with current and future user agents, including assistive technologies.		
4.1.1 Parsing. In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.	A	Passed

Table 6.2: Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1

Criteria	Level	Evaluation
4.1.2 Name, Role, Value. For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.	A	Passed
4.1.3 Status Messages. In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.	AA	Passed

Table 6.2: Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1

### 6.8.2 Techniques for Meeting WCAG Criteria

In this section, the specific techniques employed by AUDiaL in order to meet the guidelines prescribed by the WCAG 2.1 Standard, outlined in the previous section, are described in detail. Recall that each guideline is divided into success criteria at various conformance levels, and that AUDiaL was designed with the goal of meeting all guidelines up to AA level. Therefore, success criteria at AAA level are ignored in this section.

#### Principle 1: Perceivable

Regarding guideline 1.3, the content provided by AUDiaL is adaptable according to the standard: proper HTML elements are used when necessary (see Section 5.5.2) and no presentation elements are used for providing information; moreover, Accessible Rich Internet Applications (ARIA) landmarks and roles are used extensively throughout the user interface to ensure that it is accessible using standard assistive technologies. The order of content presented to assistive technologies allows the user to make sense of the content. No items of the artifact's interface (buttons, textual instructions, images) rely on their visual attributes to be used or identified. The web interface has been created following a responsive design so it can be accessed using a wide array of disparate client screen resolutions and orientations. Lastly, fine-grained definitions are included, where possible, for form inputs via the HTML autocomplete element so that their purpose can be programmatically determined by assistive technology.

Concerning guideline 1.4, all content in AUDiaL is easily accessible by persons using a screen reader and/or a standard Web browser. No color is used as the only means of conveying information either in the user interface or the accessible diagrams, and contrast ratios have

been ensured to meet the required standard<sup>2</sup>. Besides logotypes of contributing organizations, which are considered essential and thus not affected by the accessibility guidelines, no images of text are present in the prototype's front-end. Users may zoom in the user interface as much as they need to read its contents comfortably. Once the zoom exceeds a certain level, its layout is adapted to a single column so that no horizontal scrolling is required. No content is made temporarily visible on hover or focus.

### Principle 2: Operable

The full functionality of the prototype may be accessed by means of the keyboard in order to comply with guideline 2.1 to the maximum level. Users may tab to any functional element of the user interface, and may click on any text field, link, or button in order to trigger the system's functionality. Proper tabbing order is ensured by means of the HTML tabindex attribute. Moreover, actionable elements are highlighted when hovering or tabbing to them, and no specific timings for individual keystrokes are required to use the prototype. Content can be tabbed through from start to finish without trapping the keyboard focus at any time. No keyboard shortcuts consisting of only one letter or symbol have been included in the system.

```
<div role="tabpanel" class="tab-pane active query-box" id="dialog">
  <div class="form-inline">
    <div id="form-group-retrieve-label" class="form-group">
      <p class="form-control-static">Retrieve: </p>
    </div>
    <div id="form-group-retrieve-input" class="form-group">
      <label for="inputRetrieve" class="sr-only">Retrieve:</label>
      <input type="text" class="form-control" name="inputRetrieve"
             id="inputRetrieve" placeholder="e.g. What is the average
                                         population of Vienna?">
    </div>
  </div>
  <button id="submit-retrieve" class="btn btn-default clr-result">Submit
    Query
  </button>
</div>
```

Figure 6.10: Partial source code of AUDiaL's dialog tab panel area. ARIA-roles, screen reader-only labels and semantic HTML are employed, among other methods, to enhance the system's accessibility.

The prototype is fully navigable following the WCAG 2.4 guideline up to AA level. ARIA landmarks are used throughout the user interface to identify page regions. The main and navigation areas are therefore easily identifiable and can be jumped to and bypassed easily

---

<sup>2</sup>Contrast levels of interface elements were checked up using the WAVE Web Accessibility Evaluation Tool extension for Firefox, available at <https://wave.webaim.org/>

by users of assistive technologies. Page titles are descriptive, listing the most important identifying information first. All interactive elements of the prototype, such as form input fields, are ensured to follow the same navigational order as their intended logical order. Link purpose is made clear for all links by employing descriptive link text. Both a navigation menu and a table of contents (on the main page) are available to users so that individual Web pages can be located in more than one way. All focusable user interface components, e.g. text input fields and links, are highlighted when they receive focus to ensure that it can be visually identified by sighted users.

Input modalities are kept simple in AUDiaL by using only standard HTML buttons, links, and input fields that react to up-events on keypresses or click actions to complete their function, none of which are limited to touch-only input on touchscreen devices. In addition, accessible names of controls are matched to their visible labels. Lastly, no functionality in AUDiaL requires activation by device motion or user motion. In this manner, AUDiaL fulfills all requirements of guideline 2.5 up to AA level, ensuring its operability by the widest array of inputs beyond the use of a keyboard.

### **Principle 3: Understandable**

The artifact is ensured to be readable according to the WCAG 2.1 guidelines. The `lang` attribute is used on the HTML element to identify the default language (English) of the Web application. Since no other language is present in the prototype, this method already ensures that the prototype is readable with regard to accessibility up to AA level. Regarding its predictability, context changes in AUDiaL (for example, displaying the answer to a user's query) must always be initiated by the user by employing some element of the user interface such as a button. The presentation and layout of the interface are consistent in all pages that make up the prototype. The order and location of interface elements, including navigation menus and dialog components, do not change in any page. Labels, names, and text alternatives are used consistently for content that has the same functionality. With respect to input assistance, all system errors are shown in text to the user in the same `aria-live` region where query answers are given, so that users are aware of system errors without leaving the query area. Text labels are associated to all form controls by means of label elements and corresponding ARIA properties. Corrections to input errors or ambiguities are suggested to users via accessible mapping and disambiguation dialogs (see Section 5.8) so that they may understand the nature of the error and how to correct it.

### **Principle 4: Robust**

Lastly, system robustness has been scrutinized against WCAG 2.1 guidelines. The system's HTML source was validated to ensure that it complies with the specifications set out by the W3C<sup>3</sup>. Screen readers and other assistive technology are ensured to gather information about, activate, and keep up to date on the status of user interface controls in the content by setting

---

<sup>3</sup>AUDiaL's source HTML code was validated with the W3C Markup Validation Service: <https://validator.w3.org/>

programmatically determinable name and role for all user interface elements in the prototype. This is achieved by employing suitable ARIA elements (`aria-label`, `aria-labelledby`) and semantic HTML for all form controls, links, and labels. As previously stated, status and error messages are displayed in an ARIA live region, thereby ensuring that screen reader users are aware of them.

## 6.9 Prototype Validation

At this stage, the Natural Language Interface (NLI) to semantically-enhanced statistical charts prototype has been fully designed in agreement with its requirements (c.f. Section 5.4), thereby completing the treatment design task of an initial design cycle outlined in Section 1.5. Next, the designed prototype (AUDiaL) needs to be validated by its utility with respect to the stakeholder goals. The goal of treatment validation is “to develop a design theory of an artifact in context that allows us to predict what would happen if the artifact were transferred to its intended problem context” [Wie14]. In this context, *treatment* refers to the interaction between AUDiaL and the problem context (this is, blind persons accessing statistical charts).

Note that treatment validation does not call for actually observing an implemented artifact in a real-world context; but rather to predict how an artifact will interact with its context via a simpler to handle surrogate simulated context. Therefore, real-world implementations are simulated through the use of validation models [Wie14]. A classic example of a validation model consists in studying a prototype of an artifact, interacting with a model of the intended problem context, to develop a design theory about the interaction between the artifact and a context. That is precisely what the present section sets out to explore. In our case, we expose the artifact prototype designed in this thesis (AUDiaL) to a simulated context in order to learn about its performance and to find out what the effects of the treatment are, this is, to answer the knowledge questions posed in Section 1.4.

### 6.9.1 The Empirical Cycle

The main activity of this section is to answer the central knowledge question of this thesis, which, we recall from Section 1.4, states as follows:

“Are Natural Language Interfaces (NLIs) a suitable means for accessing statistical charts non-visually?”

This activity asks for specific knowledge about a (designed) artifact i.e. AUDiaL while interacting with a specific (simulated) context. More specifically, it is concerned about the suitability of employing the NLI prototype we have designed in the context of blind persons accessing statistical charts. In turn, the definition of suitability in this context is threefold; namely, we will study whether the designed artifact is an *effective*, *efficient*, and *user-friendly* means for accessing statistical charts non-visually. These concepts will be expanded upon in the remaining of this chapter.

In design science, the empirical cycle is adopted in order to answer scientific knowledge questions such as this one in a rational, systematic manner. The empirical cycle has the structure of a rational decision cycle, made up of five tasks, as follows [Wie14]:

1. **Research problem analysis.** This task frames the empirical research problem. The research problem is stated by defining a *conceptual framework*, *knowledge questions*, and a *population*. The conceptual framework of this research has been grouped in each chapter of this document according to the specific structures under study. In this section, we will expand on it by defining statistical structures and other predicates necessary for the artifact validation process. Knowledge questions have likewise been introduced as they emerged in the context of the main design problem; they were brought together already in Section 1.4. Lastly, knowledge questions are about a population, not only about the object of study [Wie14]. In the present experimental research, as stated in Chapter 1, the population consists of functionally blind persons who are experienced Web users. The selection process of population elements, which is part of next task (research design), will be described in greater detail in Section 6.10.3.
2. **Research design and inference design.** This phase, also known as *research setup*, is concerned with the specifics of how the knowledge problem will be solved. The Object of Study (OoS), treatment, measurements, and inferences are specified so that research can take place and conclusions from the data generated from it can be drawn. Research and inference design are described in Section 6.10.
3. **Validation of research and inference design.** This step ensures that the research setup and inferences match each other and are sufficient to answer the knowledge questions. The validation of the research and inferences carried out during validation of AUDiaL are pointed out in the sections dedicated to research and inference design. Notably, potential difficulties stemming from a small participant sample are described in Section 6.10.3. Research repeatability is ensured by making SAI and AUDiaL free software, and describing the undertaken research in its totality. Validation session participants were treated following ethical guidelines of informed consent, independence, data confidentiality, and beneficence [SV02].
4. **Research execution.** Research is executed according to the research design. This procedure, that took part in a series of laboratory sessions, is described in Section 6.11.
5. **Data analysis.** Lastly, the data generated by the research is analyzed according to the inference design, a process described in Section 6.12. Individual answers to the research questions posed in Chapter 1 are ascertained from data analysis and reported in Chapter 7.

Given that the research problem analysis has been thoroughly established in previous chapters, the following section proceeds to describe the research setup task. The remaining sections of this chapter accordingly discuss the remaining tasks.

## 6.10 Research Setup

In experimental research, an experimental treatment is applied to one or more *Object of Study* (*OoS*) and any relevant observable results are measured and analyzed in order to attain useful inferences that help us answer one or more research questions. In the present empirical cycle, conclusions will be drawn from data gathered during laboratory sessions in order to perform abductive inference that allows us to answer the main research question of this thesis. In this section, the main aspects of the research setup phase are laid out.

### 6.10.1 Objects of Study

An *Object of Study* (*OoS*) is “the part of the world that the researcher interacts with in order to learn something about population elements” [Wie14]. An *OoS* is the entity where the interesting phenomena occur and from which measurements are taken. The remaining of this section will discuss in detail the devised *OoS* for the empirical cycle and will expand upon our conceptual framework by defining necessary statistical structures for their investigation.

In our specific research setup, the objects of study consist in validation models of methods for non-visual access to diagrammatic representations. Namely, on each *OoS*, individuals from the population (blind persons) interact with a model of an artifact and context. More specifically, the following *OoS* are to be scrutinized:

1. A model of an NLI to diagrams artifact i.e. AUDiaL, and the model of the context, which consists in three semantically-enhanced graphics authored for this specific purpose, along with laboratory sessions in which blind participants are asked to solve a number of pre-established tasks on the data they depict and their visualization.
2. Models of traditional means for non-visual access to diagrams, including models of artifacts (e.g. a tactile graphic) and a model of the context, which, like in the previous *OoS*, also consists in laboratory sessions with population elements (blind persons) in which they have to solve tasks on the information being diagrammatically represented.

Both kinds of *OoS* will be studied and compared against each other in laboratory sessions in order to rationally answer the main knowledge question of this thesis. A statistical difference-making experiment, discussed in Section 6.10.3, was devised in order to compare the average outcomes of both treatments (AUDiaL and traditional diagram counterparts) when used by blind persons. Qualitative evaluation regarding the usability and accessibility of AUDiaL was additionally carried out, and is also discussed in detail in Section 6.10.3. Let us next proceed to defining the statistical structures that support these experiments.

### 6.10.2 Statistical Structures

A *statistical structure* consists of a *population* and one or more *random variables* defined over those populations [Wie14]. The necessary similarity between population elements is expressed

via a *population predicate* which allows experimenters and readers to recognize population elements. In our case, the population predicate can be expressed as follows: the population consists in functionally blind persons who are proficient Web users i.e. who have more than 5 years of casual Web browsing experience using standard Web browsers and assistive technology on personal computers. Once a population predicate has been defined, during experimental setup we may additionally need a *sampling frame* i.e. a way of selecting population elements in order to acquire knowledge about the study population through statistical inference.

We now move on to the second aspect of a statistical structure: in descriptive and statistical inference, variables must be observable properties of the objects of study (i.e. AUDiaL, traditional accessible diagram alternatives, and their problem context) [Wie14]. As we outlined before, we are concerned about three variables that comprise the greater notion of the suitability (or usability) of a specific approach as a non-visual method for accessing diagrammatic representations. ISO 9241-11 defines *usability* as the “extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [Int18]. Accordingly, three variables will be under study during treatment validation:

1. *Effectiveness* is the “accuracy and completeness with which users achieve specified goals” [Int18]. In this research, effectiveness will be measured as the amount of successfully completed tasks under a given OoS, along with their perceived ease of solving from the point of view of the researcher. This will be detailed in Section 6.10.3. Other ways of measuring effectiveness in NLI systems include precision and recall measures:
  - *Precision* is defined as “the number of questions for which the system returned a correct answer divided by the number of questions for which it returned an answer at all” [CHH07].
  - The definition of *recall* varies from source to source. In evaluation of NLIs, it is most commonly defined as “the number of questions answered correctly by the system divided by the total number of questions” [CHH07]. Note that this is a different definition from that employed in most data science/statistics scenarios.
2. *Efficiency* refers to “resources used in relation to the results achieved” [Int18], such as time, human effort, and materials. This variable will be measured as the time that takes users to solve tasks under a given OoS in an at least partially successful manner.
3. *User satisfaction*, which is the “extent to which the user’s physical, cognitive and emotional responses that result from the use of a system, product or service meet the user’s needs and expectations” [Int18]. User satisfaction will be measured qualitatively by means of surveys related to the tasks being solved and artifact design as a whole.

All these variables are observable properties of the process of interacting with AUDiaL and other diagram counterparts. Efficiency by means of logging time, effectiveness by task completion scores, and user satisfaction by surveying users. Logging of task completion times,

which includes the time it takes users to resolve any dialogs necessary for the resolution of their input query, is a feature of AUDiaL which was specified as part of requirement FR3, outlined in section 5.4. Each time a user submits a query in AUDiaL, the time it takes for the complete natural language to KB pipeline (c.f. Figure 5.2) to complete execution is logged, in seconds, to a text file along with the user's name and the start and end timestamps. These logs may then be analyzed in order to study the efficiency of solving (or failing) a task by a user sample.

Effectiveness has to be inferred by the researchers according to each answer returned by the system to the users' queries while performing tasks. The logging mechanism of AUDiaL also includes information about input queries, resolved dialogs, and output answers for each user. The way in which this information is gathered and analyzed is discussed in Section 6.10.3. User satisfaction will be determined in two dimensions. The first one is concerned with inferring the system usability as a whole of the AUDiaL prototype. The second one consists in assessing the perceived workload of performing cognitive tasks in AUDiaL as opposed to traditional means of accessing diagrams non-visually. These methods are detailed in Section 6.10.3.

### 6.10.3 Research Design

In experimental research, an experimental treatment is applied to the OoS and the relevant observable results are measured and analyzed in order to attain useful inferences that help us answer one or more research questions. In the present empirical cycle, conclusions will be drawn from data gathered during laboratory sessions in order to perform abductive inference that allows us to answer the main research question of this thesis. In this section, the main aspects of the research design phase will be laid out.

#### Experimental Setup

The experimental setup was originally intended to accommodate both sample-based and case-based research, as follows:

- The main inferences stemming from effectiveness data were to be deduced via statistical tests for independence between two random variables; namely, effectiveness scores for tasks performed with the AUDiaL prototype and a traditional non-visual diagram counterpart, respectively.
- Likewise, the main inferences obtained from efficiency data were to be concluded by means of statistical tests for independence between two random variables; namely, completion times of (at least partially) correct solutions for tasks performed with the AUDiaL prototype and a traditional non-visual diagram counterpart, respectively.
- The principal inferences derived from user satisfaction data were planned to be deduced qualitatively via two complementary user satisfaction surveys which users were asked to fill in after completing the more cognitively demanding tasks on both the AUDiaL

prototype and a traditional non-visual diagram counterpart. One of these two surveys is employed to measure perceived cognitive workload, whereas the other one is traditionally used to measure the perceived usability of software systems as a whole. These two tests are detailed in Section 6.10.3.

Quantitative validation, here planned to analyze the effectiveness and efficiency of AUDiaL, generally requires performing thorough, complex, and very time consuming experiments. Therefore, it is usually recommended to complement it or to even completely replace it (during early research stages) by more cost-effective qualitative methods [WM91]. This kind of approach foregoes the costly and time-consuming need of retrieving and investigating a large sample size of objects of study. In Human-Computer Interaction (HCI) research, two sorts of qualitative data are commonly employed [WM91]:

- Keystroke-level records of user inputs and system outputs. These are automatically gathered by AUDiaL's logging mechanism, which stores information about every interaction with users, including input queries, system output, and manually and automatically resolved dialogs. Namely, the username and start and end timestamps are stored for each individual interaction.
- Records of what the user says as he or she interacts with the system, for instance, during a think-aloud evaluation session. Instead of recording or videotaping full evaluation sessions researchers may focus on critical incidences and breakdowns that appear during a session [WM91]. A *critical incident* is “behavior from the user which is sub-optimal with respect to the intentions of the user and functionality of the system” [WM91] i.e. a situation in which a user performs a task goal in a much less efficient manner than the system designer had in mind. When a critical incident takes place, verbal protocols are needed to identify and interpret the user’s intentional context, and to detect subtle problems. *Breakdowns* refer to situations in which the user becomes conscious of the properties of a tool (as opposed to the task being undertaken) because the tool has failed [WM91]. AUDiaL, being a natural language interface, automatically processes and stores user input in natural language. In addition, participants were observed as they interacted with it during laboratory sessions and interesting comments noted. Lastly, willing participants were briefly interviewed at the end of the session about their personal opinions on each OoS and the validation session as a whole.

Qualitative evaluation data can be jointly gathered by experimenters during think-aloud evaluation sessions of a system in order to infer generalizations about the ways users learn to tackle problems using a new system [CM84]. The essential characteristics of this method, known as *cooperative evaluation*, are that (1) users complete set tasks, (2) users think aloud while working, and (3) the evaluator is the designer [WM91].

An experimental setup was designed where participants could interact with both semantically-enhanced statistical charts via natural language using the AUDiaL prototype and use a

traditional non-visual alternative of the same diagrams in a single validation session to act as a baseline access method that participants are already familiar with. This experimental setup is described in detail in the remaining of this section.

### Diagram Selection and Adaptation

The design of our experimental setup commenced with a selection of three statistical charts of varying complexity found on the Web. These diagrams were selected according to two main criteria: (1) they depict information from mainstream domains of knowledge that do not require specific domain comprehension for their understanding by the average person, and (2) we constrained our search to the most commonly employed diagram types; namely, bar charts, line chart, and area charts (e.g. pie charts or stacked area charts). From this search three diagrams with the following characteristics were singled out:

- *Diagram 1*: a simultaneous combination of a bar chart and a line chart sharing the same metric space representing a total of 24 data points.
- *Diagram 2*: A stacked bar chart with grouped bars sharing the same metric space depicting a total of 81 data points.
- *Diagram 3*: A lineup of stacked area charts sharing identical but distinct metric spaces depicting a total of 102 data points.

These diagrams are shown in Appendix B. The first statistical chart was selected for being an entertaining and easy to understand diagram in order to introduce the artifact to participants and engage them from the beginning of an experimental session. The remaining two diagrams were an integral part of the objects of study studied for treatment validation, since all inferences derived from the experimental sessions stemmed from the interaction between participants and a non-visual counterpart of these graphics.

For this purpose, the three chosen diagrams were converted to semantically-enhanced diagrams underpinned by the hierarchy of ontologies described in Chapter 3 and authored using Semantic Annotator for Inkscape (SAI), described in Section 4.5. The enhanced diagrams were then uploaded to an instance of AUDiaL running on an Apache server on Ubuntu 18.04. A number of user accounts were added to the AUDiaL application in advance in order to save time during laboratory sessions with participants. Moreover, user permissions were adjusted in order to hide raw graphic diagram data and other sensitive information from users. For example, if a given user was selected to employ the long description of a diagram to perform certain tasks, access to the NLI front-end of AUDiaL was disabled for that specific user. Figure 6.11 shows a screenshot of AUDiaL's administrative back-end for fine-grained user permission configuration.

In addition to the semantically-enhanced diagrams, two traditional non-visual counterparts were generated for each selected diagram to be compared against the novel approach:

## User Permissions

		Austrian Population			Leonardo DiCaprio			Power in Europe		
Username		Dialogue	Image	Longdesc	Dialogue	Image	Longdesc	Dialogue	Image	Longdesc
001		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
002		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
003		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 6.11: User permissions back-end of AUDiaL. Access to certain elements of the prototype (diagram images, NLI mode, long descriptions) can be enabled or disabled for each user in the system.

- Tactile versions of *Diagram 2* and *Diagram 3* were manually authored in Microsoft Word and printed to Braille paper with a specialized embosser (ViewPlus SpotDot<sup>4</sup>) after being automatically translated to tactile format via the VP Formatter from the ViewPlus Tiger Software Suite<sup>5</sup>. Appropriate guidelines for the production of tactile diagrams were followed. In addition, the resulting hard copies were proofread by a blind expert to ensure their readability. Several identical copies of each tactile graphic were produced to enable laboratory sessions in which several participants could access them simultaneously (c.f. Figure 6.12).
- A detailed long description was authored as a single HTML page for each diagram and a text link to the long description adjacent to each image included to the AUDiaL prototype. Each long description consisted of three main sections as prescribed by the W3C guidelines [Egg14]. Namely, a short textual overview of the diagram contents as a whole, followed by its numerical values in tabular form, concluded with an overview on how the diagram is visually presented. Appendix C shows these long descriptions as offered to experiment participants. As shown in Figure 6.11, access to the long description of each diagram could be restricted to specific users during validation sessions.

The objects of study outlined in Section 6.10.1 are now almost fully characterized. In the remaining of this section we will focus on the missing components; namely, the specifics of the population elements and their sampling procedure.

<sup>4</sup><https://viewplus.com/product/vp-spotdot/>

<sup>5</sup><https://viewplus.com/product/tiger-software-suite/>

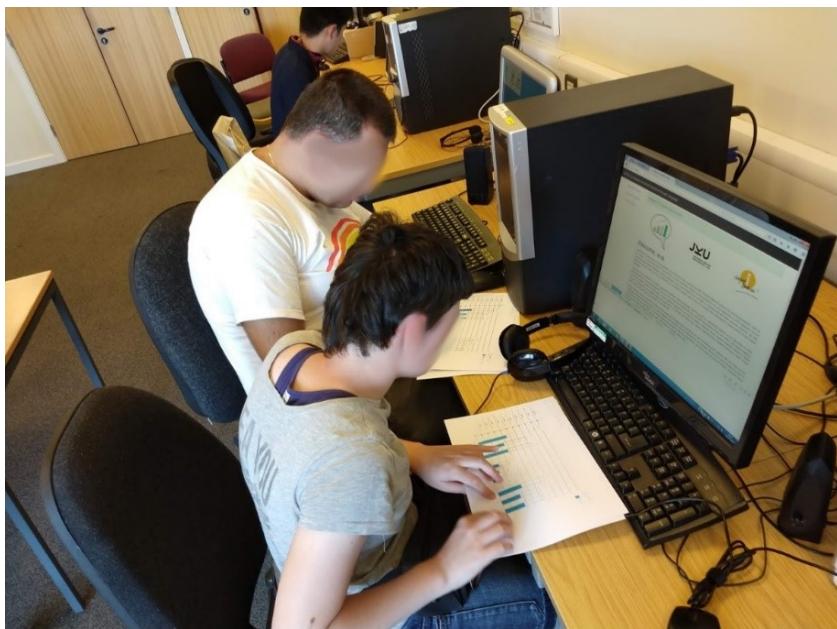


Figure 6.12: An edited photograph that shows three participants studying tactile diagrams during a laboratory validation session. One participant's computer screen shows the AUDiaL homepage. AUDiaL would be employed by these participants during the second half of the session to be compared against their chosen traditional non-visual diagram counterpart.

### Participant Sampling

Originally, sampling of participants was thought out to accommodate sample-based empirical research. Simple random sampling of blind persons was not possible due to the difficulty of contacting blind individuals among the general population. Most voluntary participants were recruited as part of a workshop on accessible graphics taking place at the International Camp on Communication and Computers (ICC) 2019 in Hereford, United Kingdom<sup>6</sup>. The workshop took part on two different days, with each session lasting around 3 hours. Out of the total allotted workshop time, a laboratory validation session lasted around 150 minutes, with two intermediate 15 minute breaks. More detail on the validation sessions is given in Section 6.11. In addition to ICC attendants, two participants were recruited among the employees of the Johannes Kepler University in Linz, Austria. These persons are blind experts in mathematics and teaching science, technology, engineering, and mathematics (STEM) subjects in graduate studies, and their opinions and contributions as blind STEM experts were extremely valuable during prototype validation.

In total, nine blind volunteers could be recruited for the validation of AUDiaL. This small sample size made carrying out sample-based research unfeasible. Nevertheless, meaningful inferences

---

<sup>6</sup><https://www.icc-camp.info/icc19>

may be derived from studying the objects of study described in Section 6.10.1, despite the small participant sample size, through case-based research [Wie14]. This undertaking will be outlined in Section 6.12. Note that a small number of persons that initially agreed to take part in the laboratory sessions ended up withdrawing them due to a general lack of interest in the tasks at hand from the early beginning or because of some sort of miscommunication that made them believe that the sessions concerned a different topic. These persons did not complete more than a few tasks with only one method of access, and are disregarded from the present research since no comparison between methods can be ascertained from the limited data collected from their sessions.

It must be noted that the resulting participant sample implies a relevant change in our theoretical population. Namely, these volunteers share many common traits, such as being highly educated, from developed countries, from middle-high socioeconomic backgrounds, and, for the most part, young. Therefore, the theoretical population was finally chosen to be confined to *blind persons from developed countries having completed, currently pursuing, or intending to pursue higher education*. Even if this population is not representative of the blind population as a whole, we believe it is nevertheless characteristic of the population most likely to encounter and be interested in accessing diagrams on the WWW in their daily lives. Further research ought to take place in order to generalize any conclusion gathered as part of this research to the general blind population.

The difficulty in obtaining a significant sample of participants is noted by the vast majority of user studies involving non-sighted persons due to common challenges such as heterogeneous participant profiles and limited availability of participants [BTM<sup>+</sup>20]. An exemplary consequence of this issue is that less than 10% of educational technologies for visually impaired children undergo some kind of quantitative empirical evaluation [KS11]. Therefore, most studies resort to including sighted participants in their experiments who are blindfolded or have limited access to any visual components of the prototype being validated e.g. [BBR<sup>+</sup>03, Bro08]. Some authors conclude that including sighted participant is a necessity due to the small sample size of the alternative validation process [BSP<sup>+</sup>19]. On the other hand, other authors doubt such practice in terms of its validity and express concern that disability simulation negatively affects the perception and representation of disabled people:

“While it may be acceptable to include sighted participants for testing interaction techniques at the level of ‘actions’, it is not for testing a device designed to do a more cognitive task. Indeed, visual impairments have consequences on education, knowledge and mental representations, which, in turn, have an impact on the execution of cognitive tasks. [...] validating the final design [of a user-centered design] with that same population [VIP users] is considered crucial. Single subject design for quantitative evaluations could be an interesting approach to address the current difficulties encountered by researchers” [BTM<sup>+</sup>20].

Given the cognitive nature of the tasks that can be carried out on diagrammatically displayed data, the validation process of AUDiaL was undertaken under this premise by taking only fully functionally blind persons into account for the pool of voluntary participants, as indicated previously in this section, even if it prevents us from performing a standard quantitative study. Section 6.12.1 gives an in-depth characterization of the participant sample.

### Measurements of Variables

The last aspect of the research design delves into the details of how the statistical variables of interest are assigned specific values during experiments. Recall from Section 6.10.2 that we are interested in measuring the effectiveness, efficiency, and user satisfaction of the OoS (c.f. Section 6.10.1). These measurements are to be employed in order to compare the proposed OoSs and to serve as the basis on which to perform abductive inference towards useful explanations.

#### Effectiveness

*Effectiveness* is measured by studying the success rate of the performed tasks with the assistance of a non-visual diagram counterpart. Each answer given by a participant to a task was manually graded with the following scale, ranging from zero to two points:

- Task completed with ease: 0
- Task partially completed or completed with difficulty: 1
- Task failed: 2

This scale enables task effectiveness to be expressed numerically and statistically summarized. Success or failure of tasks was determined by the experimenter, whereas the subjective difficulty of tasks was communicated by the participants by answering the following feedback question after answering each task:

- Were you able to complete the task?
  1. Yes, with ease
  2. Yes, but with difficulty
  3. I am not sure
  4. No, I failed to complete the task

If a participant failed to complete the task, a score of 2 was assigned to the answer regardless of their subjective impression of whether they had been able to solve it. If the participant answered the task in a partial manner, a score of 1 was also given regardless of the feedback given. If the task was answered correctly, then a score of 0 was assigned to the answer if the participant chose “Yes, with ease” to the feedback question; otherwise, the answer was given a score of 1. Answers were considered to be partial if at least half of the total correct data points or inferences were given, and if at least twice as many correct data points or inferences were provided as incorrect ones.

## Efficiency

*Efficiency* was measured by observing how long it took participants to complete those tasks that were given at least a partially correct answer. Time was kept via the implemented logging mechanism of AUDiaL, and by manually measuring it by the experimenter in the case of tasks performed on traditional non-visual diagram counterparts. Manual timekeeping is not as accurate as system log timestamps are, thus time durations were rounded to the nearest minute in both cases.

AUDiaL's time logger starts measuring time the moment a user submits a query (or navigational command) to be processed by the system. It keeps counting throughout any dialogs that may appear during the disambiguation phase of the NLP pipeline (c.f. Section 5.2), and stops measuring once a final answer, lack thereof, or a system error is communicated to the user. Start and end timestamps, total duration in seconds, and the username carrying out the task are then stored to a log file. On the other hand, time spent on tasks performed on traditional diagram alternatives started counting from the moment the task description was communicated by participants, likely a few seconds before they commenced its actual solving process. For this reason, task efficiency may be slightly overestimated for tasks carried out with AUDiaL. However, we believe that rounding time measurements to the nearest minute may compensate for these differences, since it took most participants only a few seconds after hearing a task description to begin operating on the given diagram.

## User Satisfaction

Subjective user satisfaction of each OoS was measured through two complementary means, namely:

- *System Usability Scale (SUS) scores*: SUS questionnaires are widely used as a measure of system usability satisfaction in industry. Therefore, they are the main measure of user satisfaction employed during the validation of AUDiaL.
- *NASA-TLX scores*: As an addition to SUS scores, which evaluate an application as a whole, NASA-TLX scores are employed to assess and compare the perceived workload in specific tasks.

Let us now expand on these two methods. The System Usability Scale (SUS) has been the *de facto* standard for measuring perceived software usability for more than 25 years, having provided its value over time [Bro13]. In the context of HCI with blind users, SUS is the most commonly employed measure of system usability [BTM<sup>+</sup>20]. SUS has been shown to produce the most reliable results among all sample sizes when compared to four other website usability questionnaire [TS04]. Moreover, it can be used on small sample sizes with reliable results, with a reported accuracy of 75% with a sample size of 8 participants [TS04]. It consists of a questionnaire with 10 simple and quick to answer questions, each of which has one possible answer out of five options ranging from “Strongly Agree” (numerical value: 5) to “Strongly Disagree” (numerical value: 1). The ten questions are as follows:

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

From the answers given to these questions, a final usability score (the SUS score) for each participant can be computed, ranging on a scale from 0 (extremely poor satisfaction) to 100 (extremely high satisfaction). The SUS score is computed by first summing up the score contributions from each item, where each answer is scored in the following manner:

- Odd items (1, 3, 5, 7, and 9) are scored by subtracting 1 to their numerical value
- Even items (2, 4, 6, 8, and 10) are scored by subtracting their numerical value to 5

Finally, the sum of these scores is multiplied by 2.5 to obtain the final SUS score for the given participant. SUS scores between 60 and 70 are considered average, with 100 being the ideal score.

Conversely, the NASA Task Load Index (NASA-TLX) is “a multi-dimensional scale designed to obtain workload estimates from one or more operators while they are performing a task or immediately afterwards” [Har06]. It is comprised of six sub-scales that represent “somewhat independent clusters of variables”, with the assumption that the combination of these variables are likely to represent the workload of the system’s user. Namely, these variables are *Mental*, *Physical*, and *Temporal Demands*, *Frustration*, *Effort*, and *Performance*. An overall workload score may then be computed from the weighted sum of these scores to account for individual differences [Har06]. After several decades of use, NASA-TLX is used as “a benchmark against which the efficacy of other measures, theories, or models are judged” being “recommended for use in situations as diverse as aircraft certification, operating rooms, nuclear power plant control rooms, simulated combat, and website design” [Har06].

NASA-TLX scores are employed in the validation of AUDiaL to assess the subjective perceived workload of participants when solving the two hardest tasks with either diagram modality.

After having completed (or withdrawn) each task, users were asked to answer the following questions (each question can be answered with a number from 1 to 20). Note that Physical Demand was removed from the variables under scrutiny, since AUDiaL requires virtually no physical effort to operate.

- *Mental*: How mentally demanding was the task? (1 means very low, 20 very high)
- *Temporal*: How hurried or rushed was the pace of the task? (1 means very low hurry, 20 very high hurry)
- *Performance*: How successful were you in accomplishing what you were asked to do? (1 means complete success; 20 means complete failure)
- *Effort*: How hard did you have to work to accomplish your level of performance? (1 means not hard, 20 means very hard)
- *Frustration*: How insecure, discouraged, irritated, stressed, and annoyed were you? (1 means very little, 20 means a lot)

In AUDiaL's validation these measurements we chosen to be kept separate instead of computing a weighted sum to obtain an overall perceived workload score for each task and participant. In this manner, we can take a closer look at each individual variable to infer more precise judgments on which aspects of the current prototype are users satisfied with and which ones ought to be improved in future iterations of the design cycle.

## 6.11 Research Execution

After having completed the experimental setup, this section proceeds to describe the execution of the laboratory sessions in which data about participants interacting with the OoS (Section 6.10.1) were collected for later analysis.

A Web application was created to carry out the experiments in a streamlined and straightforward manner requiring minimal external support when navigated by experimented Web users. On the landing page of this Web application instructions regarding the experiment were given to the participants. They include a brief description of AUDiaL, the overall aim of the research, example questions, and brief instructions in simple language. The instructions in full can be consulted in Appendix Appendix A. At the beginning of the laboratory sessions, each participant was given individual user credentials, and user permissions were adjusted according to the tasks being carried out at the moment. As described in Section 6.10.3, three diagrams of different complexity were selected, adapted into semantically annotated graphics and uploaded to a running instance of AUDiaL reachable from the landing page. These diagrams were also converted to traditional non-visual counterparts for comparison with AUDiaL (c.f. Section 5.3). The diagrams are shown in Appendix B.

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

A set of eight tasks to be solved by blind participants was devised for each of the diagrams. These tasks were given as one or two questions to be answered in the given text boxes of the Web application, regardless of diagram version employed. Tasks could be accompanied with a small hint or an example answer to help users comprehend what the task was about. Tasks were ordered in ascending order of difficulty, from simple questions that could be answered by reading the diagram's high-level summaries, to complex queries that required participants to build a mental model of the diagram almost its entirety. The prescribed tasks for each diagram can be consulted in Appendix D. In the following we give a brief reasoning for the inclusion these tasks in the context of prototype validation:

- Tasks 1 and 2 were simple questions whose answers could be readily found in the diagram's high-level summary in AUDiaL (c.f. Section 6.6.2) or by reading the summary of the diagram's long description. They were meant as warm-up questions that allowed participants to get acquainted with information provided by the diagram as a whole before delving into more specific tasks.
- Task 3 was a simple question that could be answered by AUDiaL directly by inputting it into the query text field. This question was meant to get participants accustomed with AUDiaL's user interface and its general operation.
- Tasks 4 and 5 asked participants for information that required to aggregate information from two different low-level input queries. For example, finding the intersection between two data series overlaid on the same diagram. These tasks were designed to influence participants into trying out different queries, keeping information in their working memory (or leverage annotation techniques see e.g. Section 6.4), and carrying out the first high-level cognitive task on a diagram.
- Task 6 trained participants in some of the accessibility features of AUDiaL. For example, users were asked to compare a single data point to a number other data points in the diagram, in order to encourage them to set the subject data point as their home node (c.f. Section 6.5), and employing navigational quick-jumps (c.f. Section 6.3.2) instead of having to navigate throughout the entirety of the data.
- Task 7 requested participants to perform several comparisons between data points, and summarize any findings. This task demanded users to retrieve, store, and compare a fairly large number of data points, more than most persons can keep in their working memory. Participants therefore had to employ some sort of annotation or summarization method to solve this task effectively. NASA-TLX scores were collected for this task after each participant completed (or withdrew) it.
- Task 8 was, for each diagram, the final and most mentally demanding assignment. It required participants to keep a large mental model of the diagram by asking them to perform some high-level inference task on its data. For example, participants were asked to figure out the general evolution of the population in Austria for the given

time period by taking into account all its constituent regions. NASA-TLX scores were gathered for this question as well.

All participants commenced by solving the eight tasks associated with *Diagram 1* using AUDiaL. No external support was given to the participants other than a brief introduction to the prototype, but they were encouraged to ask questions to the experimenter at any time. Next, participants had to try to solve the eight tasks associated with *Diagram 2*. Participants were randomly assigned to either remain employing AUDiaL or to switch to an alternative accessible diagram counterpart of their choice (tactile graphic or long description with tabular data) to act as the baseline against which AUDiaL would be measured up. This was followed by a 15-minute break. Participants were then asked to solve the eight tasks associated with *Diagram 3* employing the method they did not use for *Diagram 2*. Executing research in this manner, known as *complete counterbalanced repeated measures, task-based validation design*, minimizes sequence effects i.e. the pollution of validation results by practice effects such as participants becoming fatigued or familiar with the prototype, while halving the number of subjects needed for the experiments [Dam11].

The AUDiaL instance had been trained previous to the experimental sessions. Around 20 questions for each task were devised in advance and input into the system. These questions stemmed from both the experimenter himself and a blind colleague who did not take part in the validation process. Any disambiguation or mapping dialog (c.f. Section 5.8.2) was subsequently answered until the correct mappings from query phrases into ontology concepts had been learned by the system for all input queries. For example, phrases such as “people” or “city”, which do not appear in the semantically-enhanced diagrams at hand, were learned by the system to be mapped to “None” Ontology Concepts. AUDiaL works best when already trained by the collective intelligence of its users (c.f. Section 5.8.4), and by pre-training the prototype we intended to simulate this process.

## 6.12 Data Analysis

Once experimental sessions were completed, data was gathered from system and manual logs as well as short post-session interviews with some participants. This section discusses the analysis of this data and conclusions inferred from it with respect to the three dimensions, previously discussed, that encompass the overall usability of the research artifact; namely, effectiveness, efficiency, and user satisfaction<sup>7</sup>. Section 6.10.2 and Section 6.10.3 characterized these variables and their measurements in detail. Before delving into the analysis of the variables of interest, next section characterizes the participant sample by analyzing the collected data from demographic questionnaires given to participants at the beginning of laboratory sessions.

---

<sup>7</sup>This section was partially published in: Tomás Murillo-Morales and Klaus Miesenberger. Audial: A natural language interface to make statistical charts accessible to blind persons. In Klaus Miesenberger, Roberto Manduchi, Mario Covarrubias Rodriguez, and Petr Peñáz, editors, *Computers Helping People with Special Needs*, Lecture Notes in Computer Science, pages 373–384. Springer, Cham, 2020

### 6.12.1 Participant Demographics

Our specific sample of participants has  $N = 9$ , with an average age of 24.11 years and a standard deviation of 12.17 years, ranging from 17 to 58 years of age. This sample size, although small, is typical for HCI evaluation studies performed with blind users [BTM<sup>+</sup>20]. 44.4% of the participants identified as male, with the remaining ones identifying as female. 77.78% of participants described themselves as ‘totally blind’ with the remaining persons reporting having ‘near total blindness’. 88.9% percent of participants reported being visually impaired from birth, and the remaining person stated having been blind for more than 10 years. This implies that all participants were totally or functionally blind and had been so since birth or early childhood. In addition, participants were asked to answer three questions related to the ways they surf the Web in their daily lives. These questions were included in order to assess whether participants presented any hindrances that may obstruct their use of the Web tool employed during the validation sessions, such as lack of experience browsing web pages or being accustomed to assistive technology not available in the session rooms. All participants reported being accustomed to browsing the Web for more than 5 years, with 33.3% thereof reporting Web browsing experience longer than 10 years. Since all participants were proficient in navigating the Web, it can be assumed that any difficulties experienced during the evaluation of AUDiaL were not due their lack of skill or experience while using the Web prototype or the available AT.

Two additional questions were posed to participants with respect to the specific AT they employ in their daily lives. 44.4% of them noted using both TTS and Braille output to browse web pages, whereas 22.2% used only TTS, 22.2% employed only Braille. Therefore, all participants used common AT that were available in the experiment room. One participant reported using magnifier software for browsing the Web. However, her very low vision made it implausible for her to be able to correctly interpret diagrams by means of sight. Moreover, the diagrams used during the evaluation session were not displayed to the users in a visual manner at any time. Regarding specific screen reader (c.f. Section 2.1.5) usage, 88.89% of participants reported using NVDA, 55.56% were accustomed to using Jaws, and one participant reported being additionally skilled in Voiceover. The two main desktop screen readers reported by all participants (NVDA and Jaws) were pre-installed on the computers employed during the evaluation sessions.

The demographic questionnaire also included questions on the participants’ knowledge about diagram types, the frequency and type of websites they were likely to encounter diagrams, and whether accessible diagrams are a common occurrence on the WWW. The first question stated “Which of these diagram types do you know about?” with several common diagram types listed as possible answers. Figure 6.13 summarizes the answers to this questions. It can be observed how the majority of participants are familiar with three very common statistical diagrams, namely bar charts, pie charts, and line charts. On the other hand, less than half of the participants remembered dealing with link diagrams, group diagrams, histograms, and scatterplots in their lifetimes. Not a single participant was able to name other kinds of diagrams. This may be due to the fact that participants felt like they had to complete the questionnaire in a short time, even though they were not given any time constrain to fill

it in during the experiment sessions. The fact that participants are most familiar with bar and line charts is of great benefit for prototype validation, as the diagrams being employed are precisely of these two types (note that the stacked area chart used in the experimental sessions, namely *Diagram 3*, was converted to a multi-line time chart during the semantic annotation process). It was therefore assumed that, for most participants, answering the evaluation questions is not hindered by them failing to realize the purpose of using these diagrammatic representations of data in the first place.

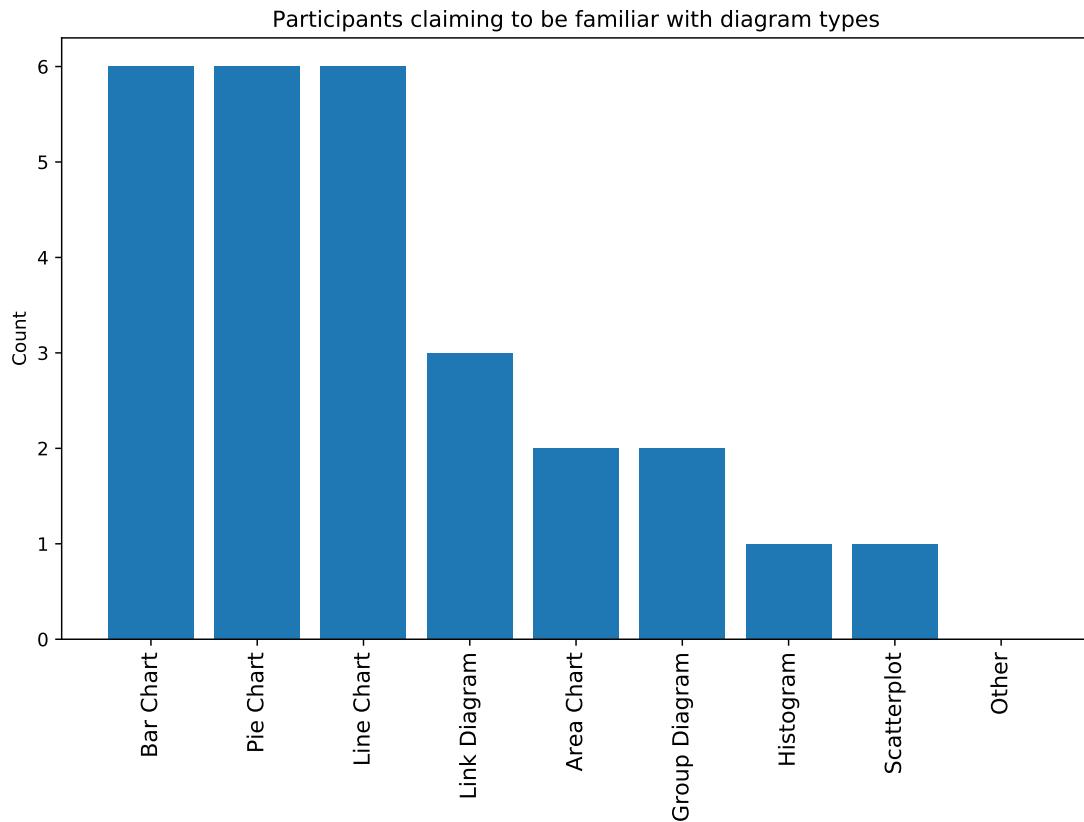


Figure 6.13: Answer counts to the question “Which of these diagram types do you know about?” by validation participants.

When inquired about the frequency of encountering diagrams on websites, 33.33% participants answered that they ‘almost never’ came across diagrams during typical Web browsing, whereas the remaining ones (11.11%) reported ‘never’ stumbling upon diagrams on the Web. On the other hand, most participants (55.56%) reported encountering diagrams on less than 15% of the websites they visit i.e. ‘not very often’. 15% of visited websites is nevertheless a high frequency, given that very often blind web readers completely skip diagrams during navigation and may not remember having encountered them.

Next question asked participants about the type of website they recall frequently encountering

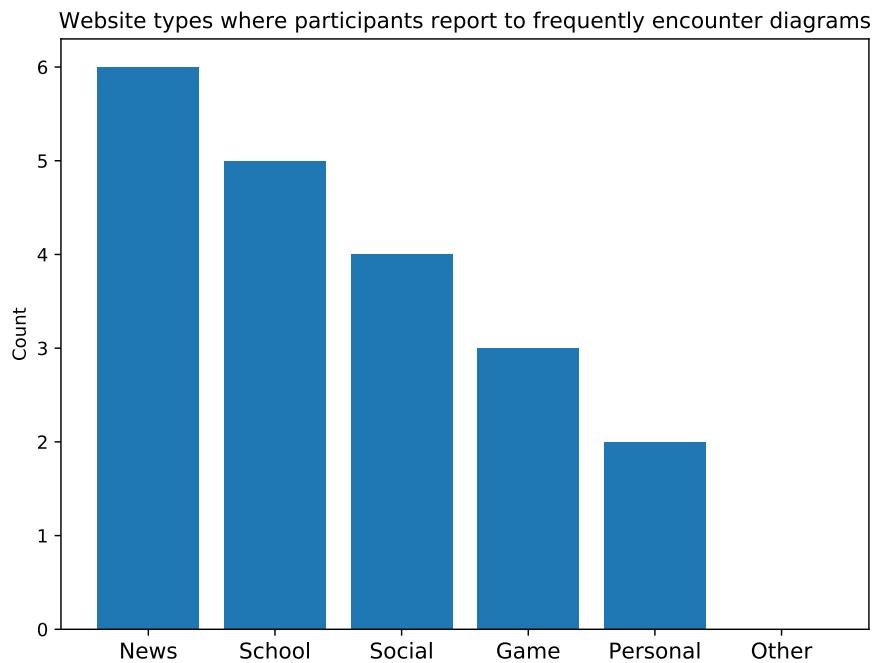


Figure 6.14: Answer counts to the question “Where do you find diagrams in Web pages?” by validation participants. Bar labels correspond, from left to right, to: “News sites”, “At my school/university course Web pages”, “Social Networks”, “Online Web-based games”, “Personal and Corporate Web pages”, and “Other web pages”.

diagrams on. The answers to this question are summarized in Figure 6.14. Not surprisingly, diagrams are most likely to appear in news and educational sites, with social networks, web games, and personal sites coming in next. The final questions of the demographic report concerned accessible diagrams. First, participants were asked the frequency with which the diagrams they encountered on the Web were properly labeled by means of an alt text description. Most (66.66%) participants reported that diagrams ‘never’ or ‘almost never’ had an associated alt attribute. 22.22% reported alt text to be present ‘sometimes’, and 11.11% reported ‘most times’. Regarding the more detailed alternative of adding a longdesc attribute to the diagram, all participants reported that diagrams on the Web ‘never’ or ‘almost never’ had an associated accessible long description. These results highlight the importance of finding automatic means for labelling and describing diagrams on the Web, given the scarcity of properly accessible diagrams on the WWW. When asked about their preferred means for accessing diagrammatic information, most users responded “Reading a textual description of its contents” (55.56%), whereas “reading an equivalent table with its contents” was chosen by 33.33% of participants, and a single participant reported a preference for tactile graphics.

Participants were lastly enquired about their personal interest in STEM subjects, being asked to rate their overall interest in a scale from zero to ten. Figure 6.15 shows the gathered answers. Most participants (66.6%) reported at least partial interest (score > 5) in STEM.

22% participants reported a very high interest in STEM (score >8). From these results we can appreciate a distribution of data congruent to the interest in STEM of the general population, if perhaps presenting a slight negative skew due to the fact that participants belong to medium-high socio-economic backgrounds.

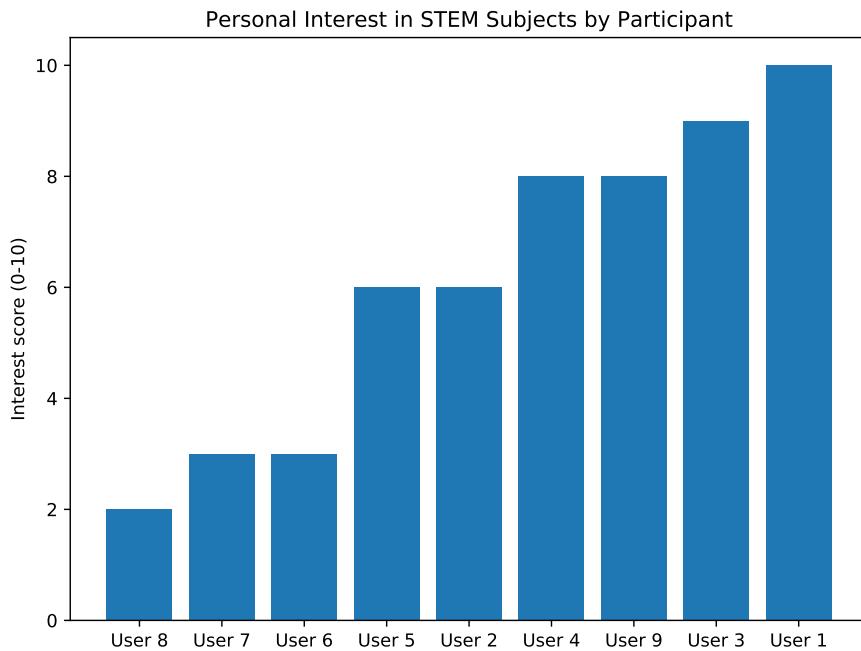


Figure 6.15: Individual subjective personal interest of validation participants in STEM subjects.

### 6.12.2 Effectiveness

A subjective assessment for *effectiveness* was requested from participants, adapted to objective measures of task difficulty, and analyzed for all tasks. Figure 6.16 displays the average collected task difficulties employing AUDiaL as a quantitative measure for effectiveness. In addition, Table 6.3 shows a statistical summary of the collected effectiveness scores. The following takeaways can be derived from their analysis:

- Most participants were able to solve all tasks at least partially; therefore, AUDiaL seems to be an efficient means for solving cognitive tasks on diagrams.
- All tasks could be solved by the majority of participants with ease, even the hardest ones. This reinforces the proposed efficiency of AUDiaL. As expected, participants faced more difficulties solving the hardest two tasks. Task 3 also yielded a relatively high difficulty score, due to the fact that it was the first task requiring users to input a natural language query.

- Task 2 was the easiest one of the lot, as its answer could be found within the summary that participants had likely already retrieved to solve task 1.
- Task difficulty lessens for tasks 4 and 5, probably due to participants becoming accustomed to operating AUDiaL. The perceived difficulty then steadily increases for the subsequent tasks, thereby confirming that the last tasks were the most difficult ones, as designed.

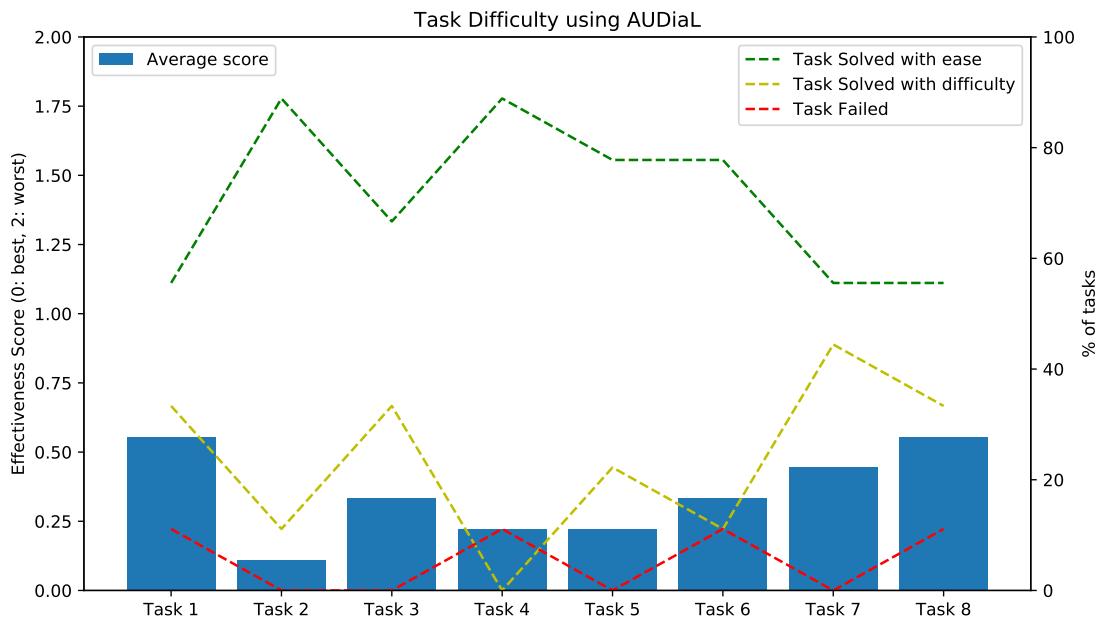


Figure 6.16: Analysis of task difficulties employing AUDiaL.

A comparison in effectiveness between AUDiaL and the alternative non-visual counterparts, measured by task difficulty, is displayed in Figure 6.17 and Table 6.3. Even if our sample size is too small to perform a statistically significant quantitative analysis of the data, a clear difference in effectiveness can be appreciated between AUDiaL and traditional accessible diagrams. There is a stark difference in the distribution of difficulty scores for Task 7; namely, most participants failed to solve these tasks using the traditional alternative; on the other hand, most persons were able to solve them with ease employing AUDiaL. There is also a clear discrepancy in the difficulty of solving Task 8 with either method; namely, most participants could solve the tasks easily with AUDiaL, whereas the majority of participants reported that solving them with a traditional alternative was difficult or the tasks were solved partially.

These results indicate that AUDiaL is a more effective means for accessing data represented in diagrams than using traditional tactile and long descriptions thereof, since complex cognitive tasks can generally be solved with less difficulty. When asked individually, participants had, for the most part, positive feedback regarding AUDiaL's effectiveness. An interesting aspect of the studied feedback is that, even when subjective impressions of AUDiaL were negative,

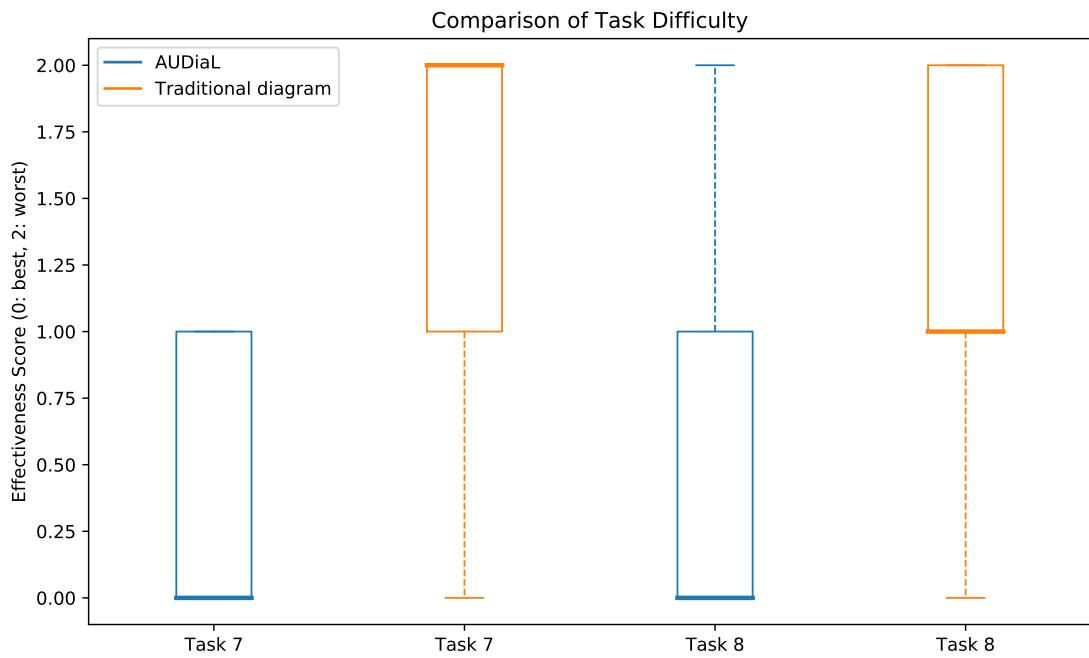


Figure 6.17: Comparison of task difficulty between AUDiaL and traditional counterparts.

the participants voicing these opinions could perform the tasks requested more effectively than with their traditional counterpart of choice. Individual feedback gathered after the experimental sessions will be analyzed in Section 6.13.

### 6.12.3 Efficiency

*Efficiency* was measured by observing how long it took participants to complete those tasks that were given at least a partially correct answer (details are given in Section 6.10.3). Times to complete all tasks, measured by the logging subsystem of AUDiaL, are summarized in Figure 6.18. Because both the answers for Task 1 and Task 2 could be fetched from the diagram's summary, most participants required a very short time to complete Task 2. For this reason, the times of Task 1 and Task 2 have been added up in the following analysis.

By observing Figure 6.18 the following interpretations can be discerned:

- Tasks 1 and 2 took participants significantly longer than other tasks to complete. This is to be expected, since participants had just encountered a new diagram and spent time getting a general idea of its contents, generally from its high-level overview (c.f. Section 6.6.2). Conversely, most participants found these tasks easy to solve, as described in previous section.
- All subsequent tasks require less time than the previous one up to Task 6, likely due to participants becoming accustomed to employing AUDiaL. Surprisingly, most

	<b>AUDiaL</b>	<b>Traditional Alternative</b>
<b>Task 7</b>	$\mu_A = 0.44$	$\mu_T = 1.33$
	$\sigma_A = 0.50$	$\sigma_T = 0.82$
	$\bar{A} = 0.00$	$\bar{T} = 2.00$
<b>Task 8</b>	$\mu_A = 0.56$	$\mu_T = 1.22$
	$\sigma_A = 0.68$	$\sigma_T = 0.79$
	$\bar{A} = 0.00$	$\bar{T} = 1.00$
<b>Combined</b>	$\mu_A = 0.50$	$\mu_T = 1.28$
	$\sigma_A = 0.60$	$\sigma_T = 0.80$
	$\bar{A} = 0.00$	$\bar{T} = 1.50$

Table 6.3: Summary of effectiveness scores. Mean, standard deviation, and mode are shown for each combination of task and diagram access method. Last row displays the combined statistics for both tasks.

users had already encountered a mapping dialog (c.f. Section 5.8.2) during Task 3 for all diagrams, even if these tasks were designed so that they could be answered without having to solve any dialog. Participants showed a tendency to inputting complex questions into AUDiaL’s input box at the beginning of the session, which in turn involved encountering more dialogs, thereby reducing both the effectiveness (see previous section) and efficiency of solving Task 3. Once participants adapted their language to what they expected AUDiaL to understand, an improvement in both effectiveness and efficiency can be clearly appreciated.

- Participants took longer to solve Task 6, since these tasks involved using some of the more complex accessibility features of AUDiaL, such as annotating nodes, setting a home node, or jumping between nodes (c.f. Section 6.1.2). Participants spent time investigating the navigation modal pane (Figure 6.2) or enquiring the experimenter about available commands. Mean time for solving Task 6 was around 10 minutes, more than double than for Task 5.
- Once participants got accustomed to the language understood by AUDiaL and gained confidence in using its accessibility features, even the most complex tasks could be solved in a notably short amount of time. Task 7 was solved in an average of 5.6 minutes, whereas Task 8 was answered on average after 4.34 minutes.

Next, a comparison between efficiency of AUDiaL and traditional diagram counterparts for solving the two most demanding tasks is displayed in Figure 6.19 and Table 6.4. The following can be appreciated:

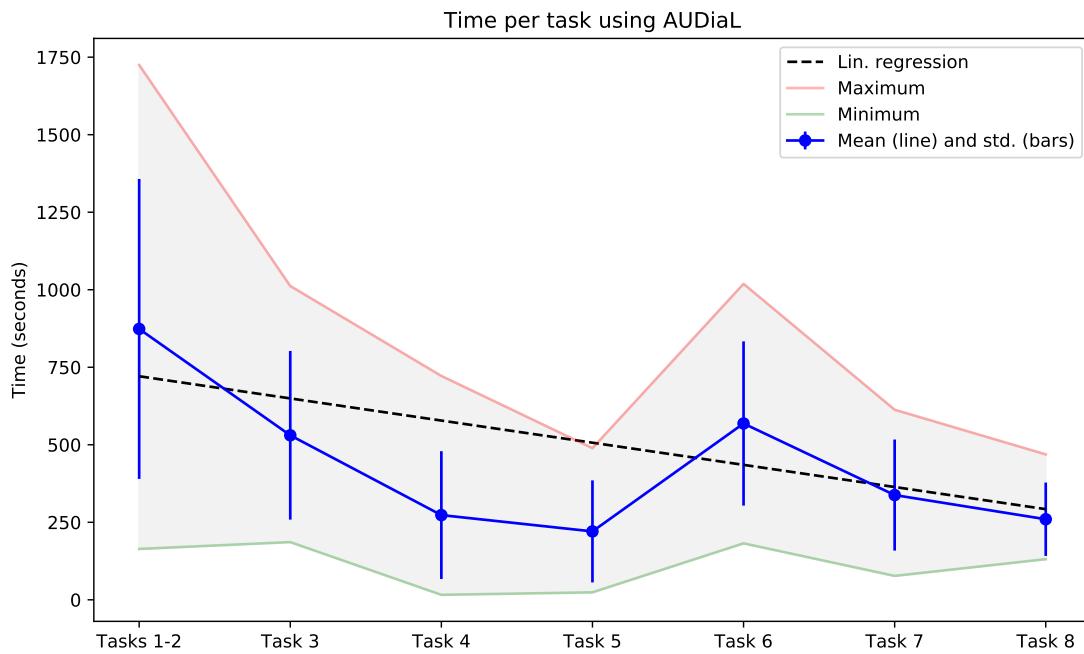


Figure 6.18: Duration of tasks completion employing AUDiaL.

- Regarding Task 7, participants required, on average, 90% less time for solving it with AUDiaL. There was however a large spread of individual times. In any case, the vast majority of participants could solve these tasks faster than the average for solving the same tasks employing the traditional counterpart.
- Task 8 yielded an even greater discrepancy in efficiency between both methods. Participants were able to solve this challenging task 3.5 times faster (on average) with AUDiaL than consulting the traditional alternative.
- Some participants required a very long time to solve these tasks using the traditional diagram counterpart. In general, participants spent a long time writing individual data points down (for example, on a MS Word document) for later retrieval and analysis, or they resorted to using online tools for computing data aggregation when necessary. These techniques slowed them down significantly as opposed to employing the native accessibility and analytical features of AUDiaL.

From this qualitative analysis it can be concluded that AUDiaL is a much more efficient means for solving complex cognitive tasks than traditional diagram counterparts. All in all, solving complex tasks with AUDiaL was 1.73 faster than with a traditional diagram alternative. Surprisingly, several participants had the subjective impression that AUDiaL was less efficient than the alternatives they were accustomed to employing in their daily lives. Subjective feedback is analyzed later in this chapter.

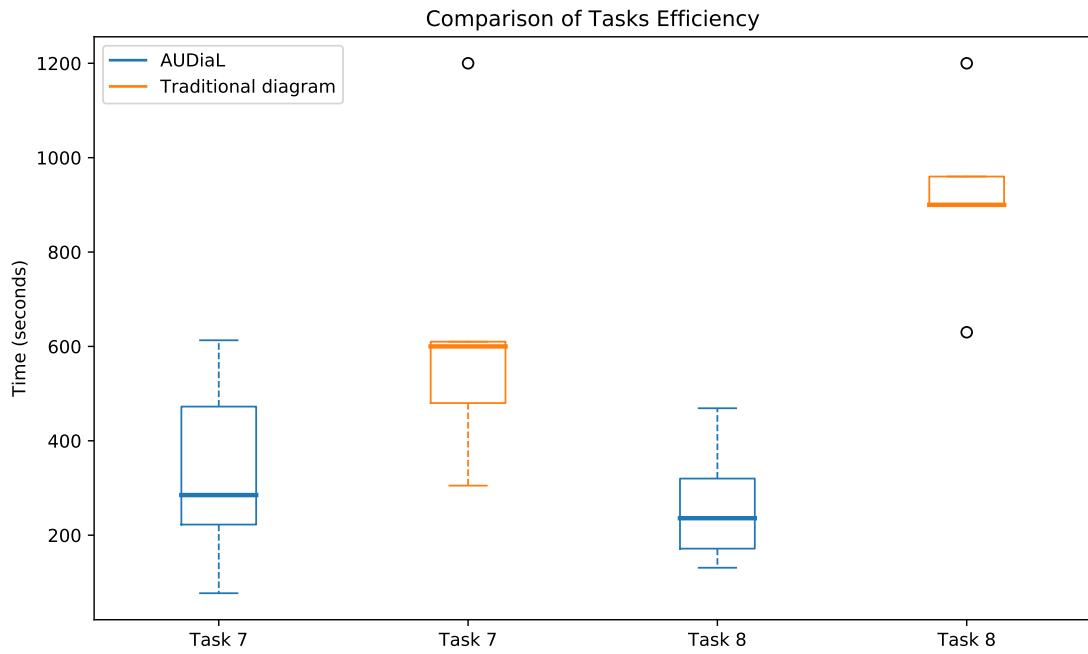


Figure 6.19: Comparison of task efficiency between AUDiaL and traditional counterparts.

	<b>AUDiaL</b>	<b>Traditional Alternative</b>
<b>Task 7</b>	$\mu_A = 337.86$ $\sigma_A = 179.10$ $\bar{A} = 285.0$	$\mu_T = 639.00$ $\sigma_T = 301.30$ $\bar{T} = 600.0$
<b>Task 8</b>	$\mu_A = 259.86$ $\sigma_A = 118.30$ $\bar{A} = 236.0$	$\mu_T = 918.00$ $\sigma_T = 181.59$ $\bar{T} = 900.0$
<b>Combined</b>	$\mu_A = 449.02$ $\sigma_A = 352.48$ $\bar{A} = 380.0$	$\mu_T = 778.50$ $\sigma_T = 285.20$ $\bar{T} = 765.0$

Table 6.4: Summary of efficiency scores (i.e. time spent to solve tasks, in seconds). Mean, standard deviation, and mode are shown for each combination of task and diagram access method. Last row displays the combined statistics for both tasks.

#### 6.12.4 User Satisfaction

The first dimension of user satisfaction that will be explored in this section is subjective perceived workload of the most cognitively demanding tasks. As previously described in Section 6.10.3, five aspects of perceived workload were enquired to participants after solving (or withdrawing) these tasks with either AUDiaL and a traditional diagram counterpart. Figure 6.20 displays the gathered averages for each NASA-TLX aspect, which can be interpreted as follows:

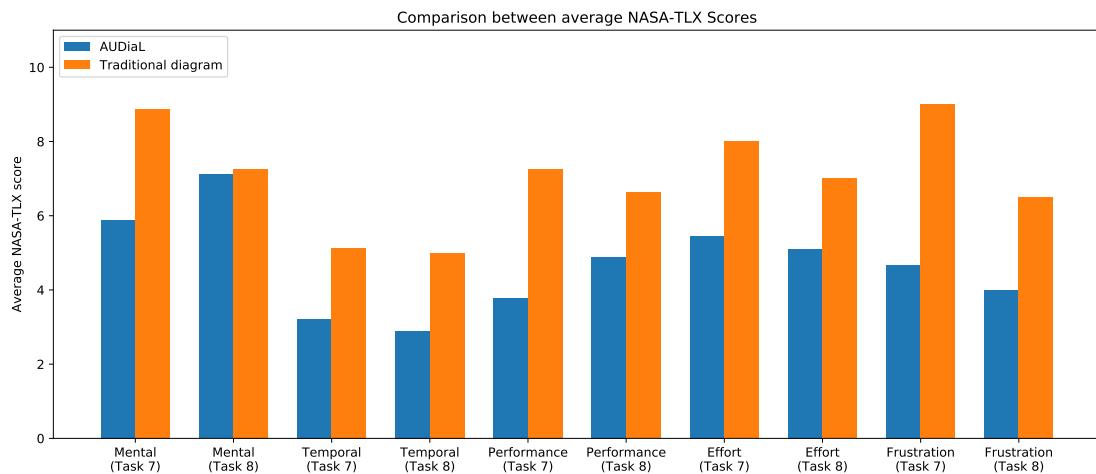


Figure 6.20: Comparison of average NASA-TLX scores between AUDiaL and traditional counterparts.

- Mental demand was perceived as lower by participants employing AUDiaL, especially for solving task 7. Surprisingly, mental demand was deemed as very similar when solving Task 8 with either method, even if, as Figure 6.19 shows, participants required significantly less time for solving it. This is because the analysis of user satisfaction also includes participants who withdrew tasks. Persons that gave up on Task 8 reported a higher mental load than those that were able to answer it, likely due to accumulated fatigue.
- Temporal demand was regarded as lower by participants in both tasks. This is consistent with the objective measures of efficiency discussed previously.
- Participants felt more successful when employing AUDiaL to solve complex tasks. These findings also conform with the objective measures of effectiveness analyzed previously.
- Participants reported having to work less hard when using AUDiaL. Participants found it especially hard to annotate and/or keep in memory intermediate data necessary for high-level analytical analysis of the diagram when employing traditional access methods such as a tabular description of the diagram.

- Frustration was reported as much higher when employing a traditional diagram counterpart. Participants often voiced irritation at the inconvenience of analyzing numeric data offered in raw tabular form, which frequently led to them giving up on the task at hand.

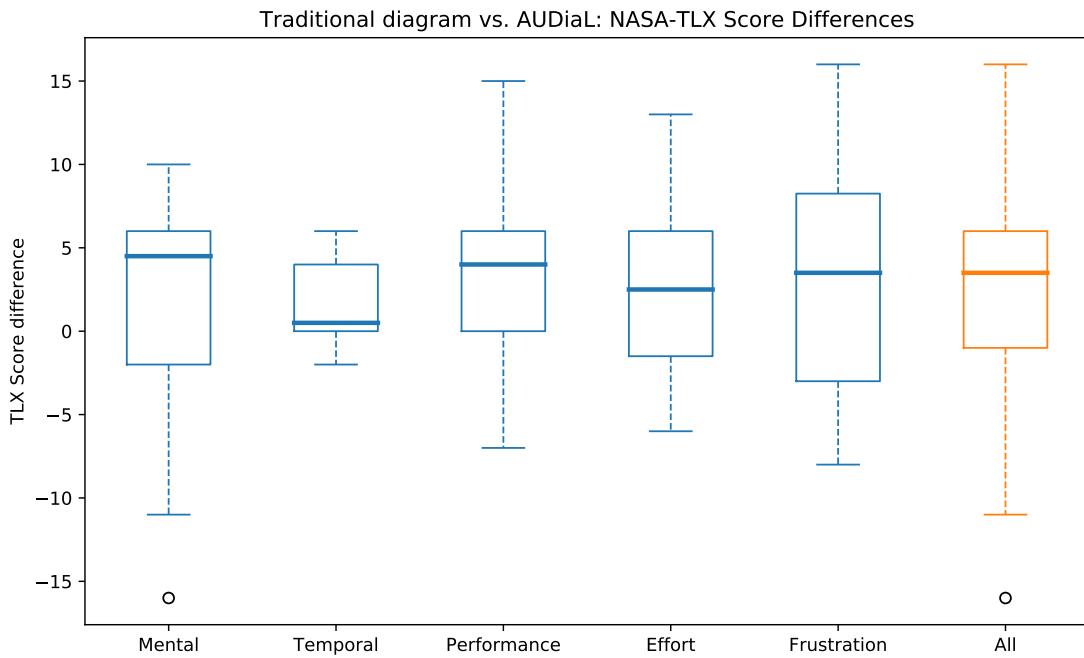


Figure 6.21: Distributions of individual and aggregated ('All') NASA-TLX score differences.

Figure 6.21 displays the distributions of NASA-TLX score differences between tasks solved using a traditional diagram counterpart and tasks solved with AUDiaL. It can be appreciated how, for all scores, the median lays above zero i.e. the median participant perceived AUDiaL as less mentally demanding, more efficient, better performing, easier to use, and less frustrating than a traditional non-visual alternative. The distribution of temporal scores is the least dispersed, since participants generally agreed that AUDiaL was the more efficient method to use, although not by a big margin. On the other hand, the remaining scores present a rather high spread. Specially, mental demand and frustration scores show a large inter-quartile range. Specifically, some participants found AUDiaL to be mentally demanding or frustrating to use due to the frequency of encountered mapping dialogs when inputting complex input queries. A detailed inspection at verbal feedback gathered from the participants is given later in this section.

At the end of laboratory sessions, participants answered the SUS questionnaire described in Section 6.10.3. Obtained individual SUS scores are displayed in Figure 6.22. The average score was a satisfactory 58.61. However, a rather large spread can be observed ( $\sigma = 16.64$ ), with

individual scores ranging from 32.5 to 80. Figure 6.23 offers a closer look at the distribution of each individual SUS score. In general terms, the usability of AUDiaL can be interpreted as adequate, although there is still room for improvement in several areas. Every gathered SUS score had a median of at least 2 out of 4, with four individual scores (40%) yielding a more favourable median value of 3. For the most part, participants found the system usable (i.e. they report a willingness to continue using it in the future), not too technical (i.e. usable in an independent manner without expert support), well integrated, and consistent. On the other hand, subjects disagreed on the complexity of the system, with some of them reporting that AUDiaL was complex to use due to the presence of mapping and filter clarification dialogs that were sometimes hard to interpret. In addition, this constrained habitability negatively affected the prototype's perceived ease of use, and ease of learning scores.

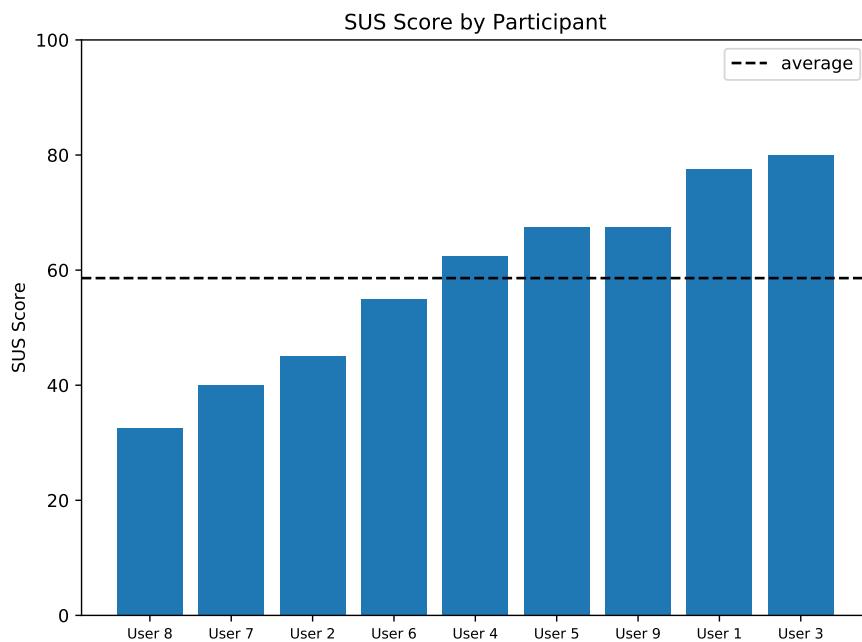


Figure 6.22: Final SUS scores per participant.

### 6.12.5 Feedback on Accessibility Features

In order to prevent participant fatigue that would have hindered the quality of the gathered data, the accessibility features of AUDiaL (c.f. Section 6.1.2) were validated through direct observation of the participants, analysis of the available logs from difference-making experiments, and voluntary interviews post-session. Ideally, additional difference-making experiments comparing the usability of AUDiaL with and without accessibility features would have been undertaken. Figure 6.24 displays the proportion of participants that employed each accessibility feature in at least one solved task. In the following, a brief analysis of each of these features in the context of AUDiaL's validation is given:

## 6. ACCESSIBILITY AND VALIDATION OF NLIS TO DIAGRAMS

---

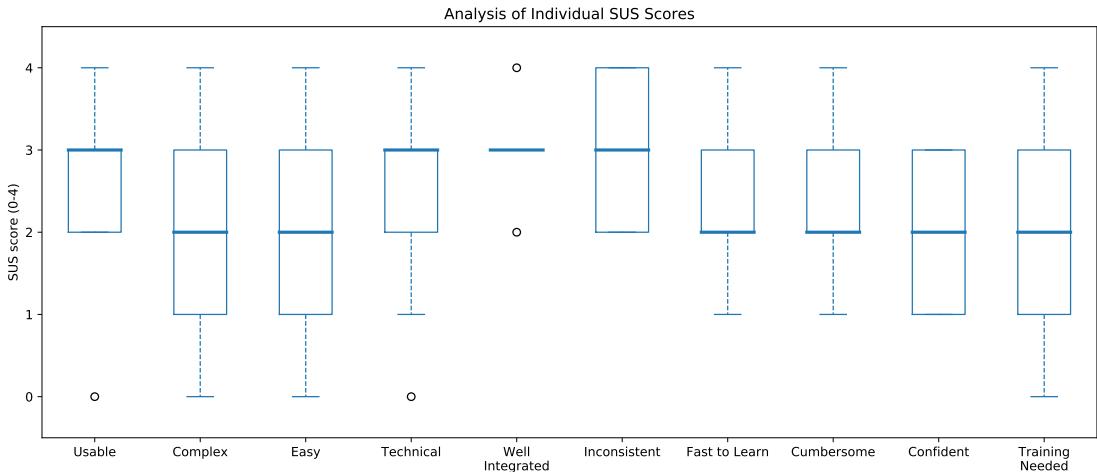


Figure 6.23: Distribution of individual SUS scores.

- Enquiring about a high-level summary of the diagram at hand is, by far, the most commonly employed accessibility feature of AUDiaL. Every single participant found this feature useful and reported that it helped him or her in gaining a better understanding of the diagram and the data it depicts. With regard to the information provided by the summary (c.f. Section 6.6.2), participants reported that it was useful and clear for the most part. However, dividing the stacked bar chart into subsets of grouped bars and displaying the list of all subsets in the diagram (see Figure 6.7 for an example) was reported as making the summary too long and hard to understand. The language employed in the summary was reported as clear and consistent for all diagrams.
- Quick jumps were used to solve at least one task by 80% of participants, most commonly to retrieve data extrema. Participants reported enjoying this feature for the most part, with one important caveat. Namely, participants reported that the offered trend summaries between the data points navigated in the jump (c.f. Section 6.6.2) were not useful; additionally, complains about terms such as 'bar' or 'steep' being confusing were frequently voiced.
- Sequentially navigating between data points was often employed by participants, sometimes even exclusively, during problem solving. Namely, 80% of participants used it effectively to solve at least one task. All participants found this feature at least somewhat useful, but some wished that it was possible to jump to distant nodes with a single command. The possibility of allowing users to jump to the  $n$ th object according to their navigational order (in addition to adjacent ones) should therefore be included in future iterations of the prototype.
- Most participants did not make use of AUDiaL's personal annotation capabilities, even if annotation was theorized to lead to an increase in efficiency, effectiveness, and user satisfaction in non-visual task solving supported by diagrams (c.f. Section 6.4). Around

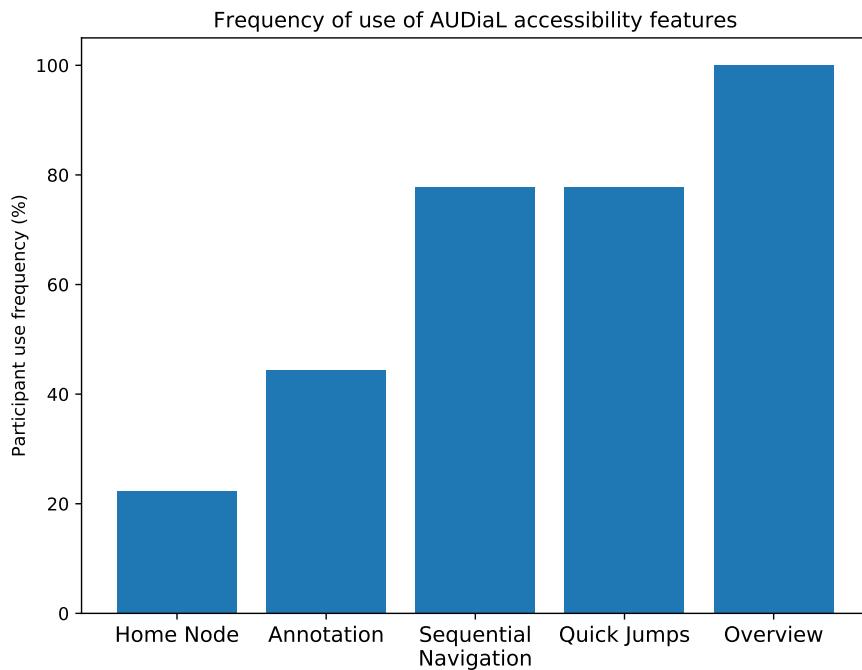


Figure 6.24: Frequency with which selected accessibility features of AUDiaL were employed in at least one task solving activity by participants.

40% of participants made some use of personal annotations to solve at least one task. Some positive comments included “I think adding my own text notes makes it easier to navigate the diagram” and “I like that I can jump to parts that I marked”. However, most participants seemed not to be aware of this feature in the first place. Only when verbally encouraged by the experimenter to try it out they seemed to become aware of its potential usefulness. Most participants reported an interest in annotation, and they assumed that it may be a convenient feature once they would get accustomed to it. However, most subjects ended up not experimenting with AUDiaL’s annotation feature, likely due to time constraints during the experimental sessions.

- Only two participants (~20%) made use of home nodes during problem solving after being recommended to give them a try. Once again it is likely that most participants did not employ this feature due to not being accustomed to it and because it is the least intuitive accessibility feature of AUDiaL. When enquired individually, some participants reported not recognizing the potential value of this feature: “I don’t understand what the home node does, I prefer to just type questions” or “maybe this is useful, but I can’t see how right now. Maybe I need more practise”. From this feedback it can be presumed that home node is a useful feature in navigation of link diagrams (c.f. Section 6.5), but probably not as much in the context of accessing statistical diagrams non-visually. However, further research is necessary to confirm this hypothesis.

### 6.13 Conclusions from the Empirical Cycle

All in all, the validation process described in this chapter has revealed that AUDiaL is a more efficient and effective means for solving complex analytical tasks on diagrams non-visually as opposed to traditional accessible diagram counterparts. The proposed tasks could generally be solved faster and with more accuracy by employing the Natural Language Interface (NLI) prototype than by consulting a tabular and textual description of the diagram or a tactile version thereof. Regarding user satisfaction, AUDiaL obtained an overall SUS score (c.f. Section 6.12.4) of 58.61, which could be interpreted as being on the low end of an adequate score in terms of the system's perceived usability by target users. Individual opinions, however, varied greatly between participants.

On the one hand, criticisms of AUDiaL included comments such as “the table [mapping dialog] is confusing, I tried to avoid it as much as possible”, “I always get stuck [on the dialog]”, or “the system is not easy, but it is usable. It feels like solving a puzzle”. This kind of critique was usually voiced by participants right after entering a long, complex query into AUDiaL’s input area. Initial frustration provoked a change of strategy for solving tasks by most participants, who then resorted to employing shorter and simpler input queries void of subordinate clauses that prompted fewer system dialogs. Other participants reported a preference for their traditional means of accessing diagrams even if AUDiaL is more efficient and effective for doing so: “I would rather read a table of the diagram because I can use the screen reader shortcuts I am used to”. She however conceded that AUDiaL was “interesting and useful for blind people”. This was a common motif among participants, who often requested for a list of available commands or queries understood by the system, which again relates to the limited natural language understanding capabilities of the prototype.

Conversely, many generally positive comments were also voiced by participants. Some of these opinions include “I would use this system if it was available online, it is better than a table to understand the data”, “textbooks are full on inaccessible graphics, a tool like this would be useful to understand them”, and “[AUDiaL] is very interesting, I wish I had more time to try it out in my own computer”. One participant voiced her frustration about having employ a traditional diagram alternative after having tried AUDiaL: “oh, this is not so much fun!”.

As shown in Figure 6.25, there is a strongly positive correlation (Pearson’s  $r = .874$ ) between STEM interest in participants and collected SUS scores for AUDiaL. Generally speaking, participants who gave AUDiaL a low SUS score were generally not engaged in the validation session as a whole since they exhibited very low interest in the tasks at hand. When enquired individually, some of them voiced comments such as “I did not like this session, I thought it would be about geographical maps. I hate maths”, “I felt like I was crunching numbers the whole time”, or “I would never do this kind of task in my daily life”. AUDiaL clearly failed to instigate an interest in these participants in the tasks being carried out or the data expressed by the diagrams. However, participants also reported skipping this type of complex diagrams in their daily lives regardless of access method, and these participants did not report a strong preference in using a traditional access means as opposed to AUDiaL if they were given such

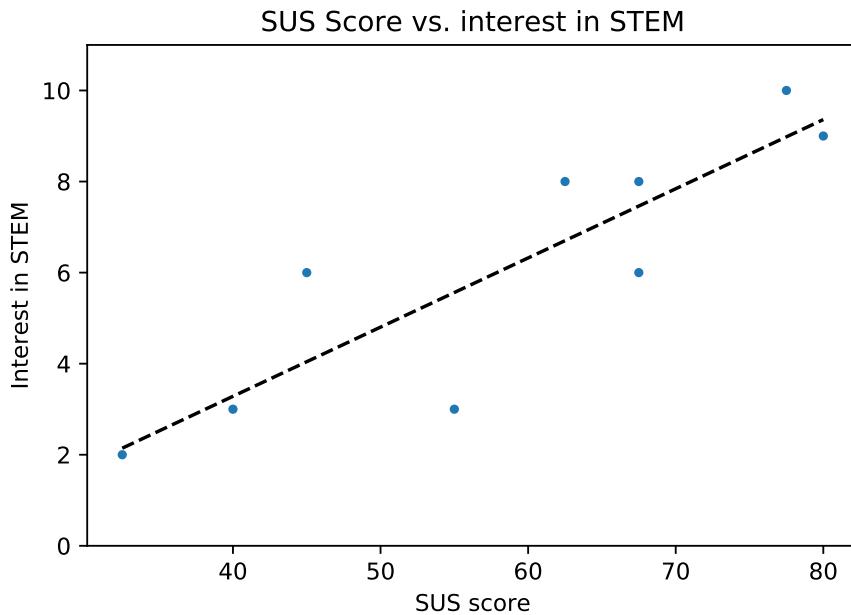


Figure 6.25: Correlation between gathered AUDiaL SUS scores and reported STEM interest among participants.

an alternative in the future.

In conclusion, AUDiaL is an effective and efficient manner for accessing diagrammatically-displayed data. Subjective opinions on its user-friendliness were, on the other hand, more disputed. However, most negative complains were related to the NLP capabilities of the prototype, especially when the system did not understand complex input queries. The inclusion of more powerful NLP capabilities would likely result in a vastly improved perceived usability. Blind persons with a previous interest in data analysis and statistics found AUDiaL an engaging and effective means of accessing diagrams preferable to their traditional counterparts. On the other hand, the NLI prototype failed to provoke an interest in such matters in the case of participants already apathetic about them. These results suggest that such an approach may be of special interest in educational settings, where blind and sighted students could engage together in the same problems supported by diagrams in an inclusive environment.

Lastly, cooperative validation of AUDiaL resulted in the detection of a number of issues with the current prototype that ought to be improved in the next iteration of the design cycle (c.f. Section 1.5). Table 6.5 summarizes these problems along with the amount of participants that voiced them. These issues were individually accounted for in this chapter.

Problem	# Participants
Complex queries with subordinate clauses are not resolved even if mapping/disambiguation dialogs are properly handled	8
High-level summary is too long and could be reduced without loss of important information	5
Trend summaries between distant data points do not contribute to diagram understanding	4
During navigation activities, object annotations are displayed too far into an answer, making them hard to read at a quick glance	2
Quick jumps only allow the user to go from one graphic object to an adjacent one, or to those carrying extreme values. It would be desirable to have a command to jump to the $n$ th object.	2

Table 6.5: Problems detected in the current version of AUDiaL during its validation.

## 6.14 Summary

This chapter set off in Section 6.1.1 by analyzing the cognitive benefits that diagrammatic representations offer to sighted persons and which make diagrams such a powerful and ubiquitous means for representing data. An introduced in Section 6.1.2, effective accessible Natural Language Interface (NLI) to diagrams must include techniques that make up, insofar as possible, for the lack of these benefits that affect VIPs. Due to the sequential nature and low informational bandwidth of natural language (as opposed to multidimensional, high-bandwidth sight), designing such mechanisms is particularly challenging.

A number of Human-Computer Interaction (HCI) techniques, grouped into five main categories, have been ultimately devised to compensate for the lack of these cognitive advantages thereby satisfying functional requirement FR2 (c.f. Section 5.4) of the designed artifact. Namely:

- The lack of *resemblance preservation* is restituted, to some extent, by leveraging ontological class extensions for visual attributes in UVQ and domain-specific formal representations, as described in Section 6.2. Node annotation techniques (c.f. Section 6.4) and automatic visual trend descriptions, described in Section 6.6.2, may also be operated to this end.
- As framed by the functional framework for wayfinding of Allen [All99], *spatial indexing and navigation* has been tackled by designing non-visual methods for performing the tasks of piloting, path integration, and cognitive map-based navigation, described in Section 6.3. Namely, these methods in AUDiaL consist of hierarchical and sequential

navigation mechanism aided by quick jumps, node annotation techniques, and home nodes.

- *Computational off-loading* is attained through automatic high-level summaries of the diagram (Section 6.6), hierarchical and sequential navigation commands (Section 6.3), node annotation techniques, and the possibility to establish home nodes whereupon system answers will be compared to it (Section 6.5).
- *Breath-then-depth search*, also known as the Visual Information-Seeking mantra, is implemented by offering users high-level overviews of the diagram, followed by hierarchical and sequential navigation aided by quick jumps to salient graphic objects, and home nodes.

Note how the HCI techniques listed above are reprocessed to realize more than one cognitive benefit simultaneously. In addition, these techniques are proposed as an extension of the typical capabilities of the NLI described in Chapter 5, many of which are already capable of compensating to some extent for the cognitive benefits previously detailed. For example, many analytical tasks of VTO of an aggregative nature (e.g. vto:Compute\_Average\_Task) can be understood as tackling both *breath-then-depth search* and *computational off-loading*.

With the inclusion of the accessibility features specified in this chapter (summarized in Section 6.7), AUDiaL fulfills all the requirements spelled out in Section 5.4, thereby completing a full iteration of its design cycle. Next, the final prototype was validated with regard to its technical accessibility in Section 6.8. Validation with target users in the empirical cycle followed. Section 6.9 commenced by describing the phases that make up the empirical cycle. The following sections discussed the research design (Section 6.10.3) and research execution (Section 6.11) that were devised to validate AUDiaL in a model of its intended problem context. A number of difference-making experiments were designed in which blind participants had to make use of AUDiaL and a traditional non-visual diagram counterpart (acting as the baseline method) in order to solve a number of common analytical tasks of varying difficulties supported by a diagrammatic representation of data.

Lastly, Section 6.12 analyzed and interpreted the data gathered during the validation process. The interpretation of these results, summarized in Section 6.13, shows that Natural Language Interface (NLI) are a much more effective and efficient means than traditional diagram counterparts for solving complex cognitive tasks supported by diagrams. In addition, the user friendliness of AUDiaL has been deemed as appropriate; however, several enhancements have been proposed to improve its usability in future iterations of the design cycle. An interesting correlation between personal interest in STEM subjects and the satisfaction of employing AUDiaL was observed. Next chapter will derive answers to the research questions originally posed at the beginning of this thesis by interpreting and summarizing the findings reported in this and previous chapters.



# CHAPTER

# 7

## Discussion

This chapter wraps up this thesis by summarizing its findings, discussing their significance, and, most importantly, giving answers to the research questions posed in Chapter 1. In addition, further work that is pending and areas of improvement are discussed for the different sections of this thesis. Lastly, potential avenues for future research stemming from this project are briefly considered.

### 7.1 Overview

Visual representations of data (i.e. diagrams) exploit the human visual system thereby enabling efficient cognitive access to large amounts of data. Consequently, diagrams have become ubiquitous and their access is a necessity for daily life. However, blind persons are excluded from accessing diagrams because alternative access methods are either too simple, too complex and/or expensive to use, or they require external support from a sighted expert. This thesis has put forward a framework supporting non-visual access to diagrams via an adapted Natural Language Interface (NLI) that aims to overcome these limitations.

The proposed framework is made up of three complementary parts. First, diagram semantics were characterized as a hierarchical set of ontologies that formalize state-of-the-art findings in data visualization research. Next, the graphical primitives that make up a (vector) diagram may be associated to specific instantiations of these ontological resources in order to create a semantically-enhanced diagram. This kind of diagram is extended with the formal semantics of what it depicts in a transparent manner and can be created with Semantic Annotator for Inkscape (SAI), a specialized authoring tool developed in the context of this thesis. Lastly, this thesis documents the design and validation process of Accessible Universal Diagrams through Dialogue (AUDiaL), an adapted, accessible NLI for non-visual access to semantically-enhanced diagrams by blind users. Validation of AUDiaL with end users demonstrates that NLIs can be a usable, for the most part, means for performing cognitive tasks non-visually on diagrammatically displayed data. They allow blind persons to access diagrams in a fully

## 7. DISCUSSION

---

autonomous, effective, and efficient manner purely employing natural language. Nevertheless, the current framework still presents areas for improvement so that it can grow into a practical tool.

### 7.2 Results

The contributions of this thesis correspond to the answers to the research questions posed in Section 1.4. This section aggregates and expands on these answers from the research undertaken in the entirety of this thesis. Note that the ordering of research questions in this section differs from that of Section 1.4, because, in this section, they are answered in the same order in which they turned up during the development of this thesis.

#### **What are the benefits and drawbacks of current approaches to non-visual access to diagrams?**

This research question was thoroughly examined in Chapter 2 by performing an exhaustive literature review, and its answer can be summarized as: invariably, state-of-the-art approaches to non-visual access to diagrams fail to meet the usability requirements of the blind population. Specifically, five main categories of access methods were identified and analysed, as follows.

Linear approaches, such as long descriptions, are inexpensive, easy to produce, and efficient for end users. However, they are far from being functionally equivalent to their visual counterparts. They overload the working memory of the reader, and do not enable the realization of cognitive tasks with ease on diagram data. This kind of approach also includes modern machine-learning methods for automatic diagram parsing, which, in addition, suffer from low perceptual accuracy yielding error-prone results that hinder their efficacy.

Haptic approaches, for example tactile graphics, can carry over many of the original visual attributes of the diagram, such as size, position, and texture. Nonetheless, current technological limitations bring about a necessary simplification of complex diagrams, thereby hindering their effectiveness. Haptic graphics moreover require expensive hardware and cannot be autonomously generated by blind users, which greatly limits their take-up.

Sonification approaches are very limited by their own nature; specific data values and visual attributes cannot be communicated by these means. In addition, they require lengthy training for effective use and must generally be authored by an expert.

Speech-supported approaches enable efficient, effective, and familiar access to diagrammatically displayed data because human-computer interaction in natural language can express information at all level of abstraction in an instinctive manner. In addition, no specific hardware or software is necessary for operating a Web-based Natural Language Interface (NLI), which stimulates its take-up by the blind population. On the other hand, traditional NLIs, being text-based, suffer from most of the drawbacks of linear approaches. Because research in NLI is rarely motivated by the goal of accessibility, no studies have been conducted on how they ought to be adapted for the special needs of blind users. In addition, current speech-enabled systems require users to employ a number of predetermined input commands

to navigate the graphic, which require previous user training and do not leverage the full potential of NLIs. These findings motivated the main resolution of this thesis; namely, to design and validate an accessible NLI to diagrams.

Lastly, hybrid approaches employ two or more modalities in association with the goal of overcoming the limitations of one approach with the strengths of a complimentary method. For example, by enhancing a tactile graphic with speech output. However, those methods that employ tactile output always require costly hardware and user training to be employed effectively. Moreover, authoring this kind of accessible diagram is a very complex and time-consuming process. The combination of speech and sonified output, on the other hand, opens interesting avenues for future research.

### **How can the semantics of diagrams be formalized for their efficient natural language-based retrieval?**

The idea of systematically formalizing diagram elements and their semantics into a set of general principles is coherent with current research in visualization semantics, in which so-called *metalanguages* for discussing diagrams are laid out. Stemming from a categorical adoption of the agreements found within these metalanguages, this thesis has proposed, in Chapter 3, a hierarchically arranged set of ontologies as the most suitable means for encoding diagram semantics at all levels of abstraction<sup>1</sup>. In addition, by formalizing diagram semantics in this manner, complex cognitive tasks can be effectively performed on the diagrammatically-displayed data by non-technical users via queries and answers in natural language.

More specifically, perceivable graphic objects, their visual attributes, and the graphic spaces they occupy are defined at the bottom of the hierarchy i.e. the Upper Visualization Ontology (UVO). The graphic syntax associating these elements to constitute graphic relations in which each element plays a certain syntactic role is characterized immediately above by the Visualization Domain Ontology (VDO). Specialized visualization and/or domain semantics may then be embedded to the visualization elements by employing resources at the next level of the hierarchy, namely the Data Domain Ontology (DDO) and domain ontology. Lastly, the topmost level of the hierarchy defines analytical and navigational tasks commonly supported by diagrams. Moreover, specific user- and system-specific resources may be included as well. This design pattern is displayed in Figure 3.11.

Graphic syntax and diagram semantics lend themselves to being readily typified by ontological constructs. The two bottom-most levels of the proposed hierarchy leverage the so-called principle of compositionality of meaning, also known as *Frege's principle*, which defines graphic syntax in a recursive manner such that a collection of perceivable objects arranged in a given structure in two-dimensional space may function as a single graphic object at a higher level of abstraction. Data semantics such as variables and their data types are also represented by class instances and linked to the constituent elements of the diagram via object properties. In

---

<sup>1</sup>The aggregated, up-to-date vocabulary is freely available at <https://ontovis.integriert-studieren.jku.at/ontovis-full/>

## 7. DISCUSSION

---

addition, resources from a myriad of available external domain ontologies can be associated to the elements of a diagram thereby extending them with world semantics beyond what is immediately depicted and which may be enquired by end users.

Ontologies are moreover especially well-suited to being queried via a Natural Language Interface (NLI). In recent years, great research effort has been given to the design of user-friendly approaches to querying the large amount of conceptual data models that make up the Semantic Web, of which NLI-based approaches make up a considerable subset. Results in this research area, known as Question-Answering over Knowledge Graphs (QA-KG), show that NLI to ontologies can be a usable method for finding information in an arbitrarily large knowledge base by non-technical users. Ontologies may additionally provide state-of-the-art statistical vision models with *a priori* knowledge that could help mitigate problems such as overfitting as well as an extreme reliance on the quantity and quality of training data.

Lastly, ontologies may be serialized into a number of XML-based formats. This makes it possible to embed an ontology directly into an image in SVG (an XML-based format itself) in a straightforward and transparent manner to the end user. The proposed SVG images shipped along a knowledge base characterizing the semantics of what they depict are known as semantically-enhanced graphics, and are leveraged extensively in this thesis.

### How can semantically-enhanced graphics be authored effectively?

This thesis has put forward an authoring tool that greatly simplifies the semantic image annotation process to create semantically-enhanced graphics, and, more specifically, semantically-enhanced diagrams underpinned by the ontologies for visualization described in Chapter 3. The design process of this tool, that we have named Semantic Annotator for Inkscape (SAI)<sup>2</sup>, is thoroughly documented in Chapter 4.

The treatment design of SAI resulted in a number of features enabling an effective authoring process lacked by current semantic image annotators. Namely, the constituent elements of an SVG file may be selected individually or grouped together to be associated to one or more resources of an ontology via a user-friendly GUI. The relevant resources may be already present in the ontology, manually input by the user, or generated by SAI in an automatic fashion. The namespaces appearing in the supporting ontology are automatically detected and included in the URI of the generated resources. Moreover, the full URI of the resulting resources, including resource name, may be also automatically generated. Owing to this naming mechanism, several object instances or property occurrences may be added simultaneously to the ontology. The most important structural annotations underpinned by UVO are additionally included into the resulting ontology in a transparent manner to the user. These features make the semantic annotation process of diagrams much more efficient as opposed to traditional disjoint authoring workflows or employing any other available image semantic annotator.

---

<sup>2</sup>Semantic Annotator for Inkscape is free software; its source code and installation instructions are available at <https://github.com/tomurillo/SAI>

SAI in its current status is a usable vector image annotator tool that has been employed to author the semantically-enhanced diagrams produced during the development of this thesis. In addition to its current features, several avenues for improvement have been suggested in Section 4.5.3 that would result in an increased efficiency and user-friendliness of the authoring process.

### How should a Natural Language Interface (NLI) be adapted to the specific needs of blind persons accessing diagrams?

Chapter 5 tackled the design problem of developing a prototype NLI to semantically-enhanced statistical charts enabling users to perform analytical tasks commonly supported by diagrammatically displayed data. This prototype was named AUDiaL<sup>3</sup>. Chapter 6 tackled this research question by adapting AUDiaL to the special access needs of Visually Impaired Persons (VIPs) with the inclusion of a number of accessible HCI techniques fully operable by means of natural language. Their design was preceded by an analysis of the cognitive benefits that diagrams provide to sighted persons. In turn, these HCI techniques were devised with the goal of compensating, to the greatest extent possible, for the lack of these benefits that VIPs suffer from. Five techniques were included in the final prototype, which are summarized in Section 6.7.

The first accessibility technique enables users of AUDiaL to sequentially navigate the informational graphic objects of a diagram taking into account their logical hierarchy and spatial arrangement. Another wayfinding technique supports quick navigational jumps to salient objects, such as those carrying the maximum and minimum value of the represented data. These two features comprise the backbone of the non-visual navigation capabilities of AUDiaL, enabling users to perform common wayfinding tasks such as *piloting* and *cognitive map-based navigation*.

Two methods letting graphics object to be labelled with user-specific annotations were also included. The first one enables users to add free-formed textual annotations to any informational graphic object, whereas the second method lets users set a *home* object that every system answer will relate to. Users may additionally choose to input natural language queries pertaining to any personal annotation. Annotations are useful in order to lighten the mental load of users, to help users in wayfinding tasks e.g. by preventing lostness, and to replace features of visual presentation.

Lastly, a high-level summary of the diagram as a whole and the data it depicts may be requested by users at any time. In this manner, the ability of quick glancing at a diagram to obtain an overview of its contents is replaced by a brief verbal description, which aims to acquaint users with the main characteristics of the diagram and the data it depicts before they delve into specific tasks. Consulting a summary of the diagram as a whole corresponds to the first step in the cognitive process of *breadth-then-depth search* that sighted persons carry out instinctively, thereby assisting blind users in reducing their working memory load. In

---

<sup>3</sup>AUDiaL is free software; its source code and installation instructions are available at <https://github.com/tomurillo/AUDiaL>

## 7. DISCUSSION

---

addition, short descriptions of the visual trend between data points navigated by the user is offered by AUDiaL, with the aim of getting across some aspects of the visual presentation necessary for its interpretation.

Among these methods, some of them have been shown to be especially effective for non-visually accessing diagrams. The importance of adding accessible high-level overviews to the NLI was demonstrated by the fact that all AUDiaL validation participants consulted and valued them. Likewise, navigational features were very positively received and effectively employed. Personalized annotations were exercised not as much; however, participants who made use of them reported a generally positive impression. On the other hand, home nodes and visual trend descriptions failed to meet their expectations for usability, and may have to be revised in a future iteration of AUDiaL's design cycle.

### **Are Natural Language Interface (NLI) a suitable means for accessing statistical charts non-visually?**

The main research question of this thesis was answered by validating the implemented accessible NLI prototype (AUDiaL) with blind participants, a pursuit documented in the second half of Chapter 6. The term *suitability* in the context of this question was described as encompassing three complementary dimensions that were assessed independently; namely, *effectiveness*, *efficiency*, and *user satisfaction*. These indicators were evaluated qualitatively in difference-making experiments in which the NLI prototype was compared against a traditional non-visual diagram counterpart for solving the same kind of task.

Regarding effectiveness, AUDiaL has been shown to be a much more effective means for solving complex analytical tasks on diagrammatically displayed data. Participants could solve the given assignments more accurately and withdrew complex tasks less often when employing the NLI prototype. In addition, participants could solve tasks considerably faster using AUDiaL than a traditional counterpart, especially the more complex ones that involved keeping a large mental model of the diagram. Therefore, NLIs to diagrams also demonstrated a better efficiency than traditional access methods.

Conversely, the question of the users' subjective satisfaction with AUDiaL proved to be more contended. Some participants greatly enjoyed interacting with the NLI prototype, whereas others found it tedious and cumbersome and reported preferring employing traditional diagram counterparts. As a direct result of the collected feedback, a number of enhancements to the current prototype have been suggested with the goal of enhancing the prototype's usability. It must be noted that a strong correlation was detected between user satisfaction and personal interest in STEM subjects. Participants with a personal interest in science and mathematics were more engaged in the laboratory sessions and reported higher satisfaction levels.

All in all, this research has revealed that accessible NLIs are a potentially suitable alternative to traditional non-visual access methods to diagrams. Complex cognitive tasks commonly supported by data visualization can be solved more effectively and efficiently with a properly designed NLI by blind persons than with the use of traditional alternatives. In addition, persons with an interest in the data at hand reported a preference for the NLI approach

over traditional alternatives. However, the NLI prototype failed to provoke an interest in such matters in persons already apathetic about them. These results suggest that NLIs to diagrams may be of special interest in educational settings where blind and sighted students could engage together in problem-solving activities aided by statistical charts in inclusive classrooms, or in specialized publications consulted by an expert audience.

### 7.3 Outstanding Issues

The contributions of this thesis did not resolve all areas explored during its development, and some outstanding issues remain. This section discusses such issues following the same order as the chapters of this thesis where they were first encountered.

With relation to the hierarchy of ontologies devised in Chapter 3, only diagrams from a subset of visualization domains may be currently underpinned by them. More precisely, the VDO in its current status characterizes resources pertaining to bar charts, line charts, area charts (e.g. pie charts), and volume charts. Even if these classes make up the majority of statistical charts commonly encountered on the Web, a number of typical visualization domains (c.f. Section 1.1.3) have remained out of scope in this thesis. Perhaps most importantly, link diagrams, geographical maps, and grouping diagrams have been disregarded in this work. Hybrid and nested graphic representations have also been given very little attention, beyond simple bar/line combination charts. A truly universal NLI to diagrams designed following the method proposed in this thesis must enable access to these domains, which in turn would require them to be formally characterized following the approach introduced in Chapter 3.

The overall usability AUDiaL is hindered by a number of obstacles. The most pressing one of them relates to the natural language processing capabilities of the prototype. AUDiaL was designed so that it returns accurate answers at all level of abstraction, including communicating individual data point values. Therefore, state-of-the-art sub-symbolic NLP methods were not considered as the backbone of AUDiaL, because, as described in Section 2.5.3, their lower accuracy would hinder the effectiveness of the prototype. However, the inclusion of a traditional NLP pipeline alternative results in less powerful natural language understanding, especially when confronted with long, complex queries input by participants. In short, the trade-off between effectiveness and user satisfaction of the prototype was decided in favor of a more effective artifact because it was determined as the more important metric in its validation context.

Another important issue stems from the analysis of the validation process of AUDiaL. Namely, it was observed that blind participants did not enquire about any aspects related to the visual presentation of the diagrams at hand, who were instead more concerned about the underlying data itself. Diagram- and visualization-specific terms such as 'bar', 'axis', and 'slope' often confused participants when given as part of the system's output, and were not once employed by participants for querying the diagram. This outcome suggests that AUDiaL may be over-engineered with respect to its main purpose i.e. accessibility. Fully automated end-to-end pipelines that aim to extract the numerical values of data points present in visualization are gaining research traction in recent years; approaches combining

these methodologies with human support for fine-tuning and error correction may be a more practical endeavor towards making diagrams on the Web fully accessible. Nevertheless, the particular accessibility HCI features put forward by this thesis (c.f. Section 6.1.2) would be an advantageous addition to technologies inspired by these approaches. Furthermore, automatic approaches may benefit from the *a priori* knowledge offered by the proposed knowledge base in order to increase their perceptual and analytical capabilities, as discussed in Section 3.5.2.

The validation of the NLI prototype, including its HCI accessibility features, would benefit from supplementary quantitative approaches. Such validation methods may include statistical difference-making experiments in which the effectiveness and efficiency of the prototype would be measured up against a traditional non-visual diagram alternative, similarly to the method undertaken in Section 6.9. However, as opposed to the qualitative methods carried out in this thesis, increasing participant sample size would enable to statistically infer these random variables via statistical hypothesis testing or confidence interval estimation. These are well-established and, if properly designed, very robust methods for estimating the population mean of variables (in our case, the effectiveness and efficiency of the OoS defined in Section 6.10.1). However, qualitative methods require gathering a large amount of participants, which, as previously discussed in Section 6.10.3, is a very challenging task given the particular characteristics of our target group.

Some of the HCI accessibility features of AUDiaL were barely leveraged by participants during the course of the validation sessions, and further investigation is necessary to determine the cause of their low demand. For example, most participants did not employ the home node feature, even if related research has shown that similar mechanisms are useful in non-visual access to link diagrams. Personalized annotations of graphic objects was also an underutilized feature despite its hypothesized potential for improving the system's usability. It remains to be demonstrated whether these features are not advantageous in the context of non-visual access to statistical charts, or whether participants were simply not aware of their utility but would have found them favorable given enough training time.

## 7.4 Future Work

The research put forward in this thesis is suitable to being explored and expanded in a diverse number of directions, some of which are introduced in this section.

As stated in previous section, the characterization of additional diagrammatic domains could be incorporated as an expansion to hierarchy of ontologies developed in Chapter 3. Such an enlargement of the knowledge base could stimulate similar approaches to AUDiaL for accessing disparate diagrams via natural language as well as further heterogeneous applications. At the moment, the developed ontologies are rich enough for descriptive purposes by employing axioms and constructs from the OWL universe. In addition, the knowledge base as it currently stands could benefit from the inclusion of advanced OWL constructors enabling high-level reasoning to take place directly on it by standard reasoners. The vast majority of inferences made by AUDiaL on the ontologies is done externally by the NLI prototype itself; a more self-contained knowledge base would benefit its take-up in a variety of application domains.

Semantic Annotator for Inkscape (SAI) presents room for improvement in several areas. Most notably, in its current state the semantic annotation of bitmap images is very inefficient. The user must start this process by manually authoring a vector counterpart of the bitmap graphic, a very time-consuming process. Ideally, automatic image segmentation and/or object detection techniques ought to be included. In this manner, the user would merely have to assign ontological resources to the automatically inferred regions/objects of the image, which would drastically increase the effectiveness of the semantic annotation process. Additional avenues for advancing the usability of SAI were already explored in Section 4.5.3.

The HCI features included in AUDiaL to improve its accessibility, discussed in Section 6.1.2, also present interesting courses for further research. The high-level diagram overviews offered by AUDiaL only take into account the structural properties of the diagram. Domain-aware summaries could be more informative for end users, as they leverage information that cannot be automatically inferred from the visualization aspects themselves; for example, clustering datapoints in a scatterplot according to domain-specific attributes not present in the graphic relations of the diagram. Likewise, navigational quick jumps to salient nodes according to domain-specific criteria could be of great benefit for users. More complex trend description and summarization algorithms may be investigated; advanced tagging/annotation mechanisms derived from their use in e.g. knowledge management software could be contemplated as well. Most importantly, the features that were underutilized by participants during validation of AUDiaL (see Section 6.12.5) should be studied more deeply. The conduction of difference-making experiments comparing the usability of AUDiaL with and without its accessibility features enabled may shed a light on how they ought to be adapted and improved.

The integration of sonification (c.f. Section 2.3) and/or haptic (c.f. Section 2.2) approaches into AUDiaL could be an interesting addition in order to research the effects of enhancing an NLI with additional sensory output for accessibility. Lastly, the framework proposed in this thesis has numerous compelling potential application scenarios beyond that of non-visual accessibility to diagrams. Semantically-enhanced diagrams may be the foundation of automatic diagram simplification methods for cognitive accessibility, or may be employed by the general public to access very large and complex diagrams more effectively via specialized user interfaces.

The field of Natural Language Processing (NLP), and, by extension, NLIs have experienced in recent years a thrilling research increase in terms of quantity as well as quality. It is only natural that a myriad of exciting applications enabled by natural language interactions will continue emerging. Hopefully, upcoming NLI research will lay its focus on making the world a more inclusive place for all, this thesis being a small step forward in that direction.



# List of Figures

1.1	A mathematical graph containing three cycles . . . . .	5
1.2	Grand Staff, a classic example of diagram . . . . .	7
1.3	External topography of a French Bulldog . . . . .	8
1.4	Examples of map . . . . .	10
1.5	Examples of statistical chart . . . . .	10
1.6	Examples of time chart . . . . .	11
1.7	Examples of link diagram . . . . .	12
1.8	Examples of hybrid graphic representations . . . . .	13
1.9	Example of a nested graphic representation . . . . .	14
1.10	High-level block diagram of the proposed framework for natural-language-based access to graphics . . . . .	19
1.11	Design science research iterates over solving design problems and answering knowledge questions . . . . .	21
1.12	The Design Cycle of the Design Science methodology . . . . .	22
1.13	Methodological structure of this thesis . . . . .	25
2.1	A simple bar chart . . . . .	29
2.2	An example of Braille text . . . . .	31
2.3	Architecture of screen readers and graphical user interfaces . . . . .	35
2.4	Automatic captioning of a picture employing a neural network approach .	38
2.5	A hard-copy tactile high school floor plan . . . . .	43
2.6	Decision tree for choosing which print images need tactile alternatives .	44
2.7	A tactile matrix display displaying a bar chart . . . . .	47
2.8	A sonified line chart . . . . .	52
2.9	Visual representation of an auditory city map . . . . .	55
2.10	Sonified summarization of a link diagram . . . . .	56
2.11	The Talking Tactile Tablet, a multimodal diagram approach . . . . .	59
2.12	Samples from a dataset of images used for building VQA systems . . . . .	65
2.13	Structure of a Communicative Image . . . . .	73
2.14	Graphical Ontology super-class employed in Communicative Images . . .	74
2.15	Image Segmentation in Communicative Images . . . . .	76
2.16	RDF ontology embedded within an SVG file . . . . .	77
2.17	Example dialog between a user and Communicative Image . . . . .	78
2.18	Generated grammar patterns from a Communicative Image . . . . .	79

2.19 Components of Web accessibility . . . . .	83
2.20 Example of an accessible simple HTML graphic . . . . .	85
3.1 The “Semantic Gap” . . . . .	94
3.2 Abstraction in Object-oriented design . . . . .	95
3.3 A small, informal RDF graph . . . . .	101
3.4 Parts hierarchy of the OWL 2 RDF-based semantics . . . . .	110
3.5 Technological stack of the Semantic Web . . . . .	112
3.6 A subgraph of the Visual Genome ontology . . . . .	113
3.7 Partial semantic annotation of the regions of a Communicative Image . . . . .	115
3.8 Partial taxonomy of diagram elements according to Cleveland . . . . .	117
3.9 Rule categories of signs according to Morris . . . . .	119
3.10 Three Layer Approach (TLA) to ontology engineering . . . . .	122
3.11 Hierarchical five-layer ontology pattern for visualization . . . . .	124
3.12 Automatic diagram understanding processing pipeline . . . . .	125
3.13 Recursive nature of graphics in UV0 . . . . .	134
3.14 Aggregation of Graphic Objects characterized as containment relationships . . . . .	150
3.15 Syntactic overload of the Labelling Graphic Relation . . . . .	154
3.16 Suitability of a graphic relation to represent an information type as characterized by DDO . . . . .	163
4.1 Embedding UV0 in a SVG image . . . . .	180
4.2 A semantically-enhanced diagram . . . . .	182
4.3 Importing an external ontology into a semantically-enhanced diagram . . . . .	183
4.4 A simple bar chart void of associated semantics . . . . .	184
4.5 Semantic annotation of a diagram with UV0 resources . . . . .	185
4.6 Semantic annotation of a diagram with VDO and DDO resources . . . . .	186
4.7 Main elements of Inkscape’s Graphical User Interface . . . . .	190
4.8 Protégé’s main workspace . . . . .	192
4.9 Two approaches to object instance annotation . . . . .	194
4.10 Automatic object detection in images via Convolutional Neural Networks . . . . .	195
4.11 LabelMe: a crowd-sourced image annotation Web tool . . . . .	198
4.12 The M-OntoMat-Annotizer, a manual semantic image annotation tool . . . . .	201
4.13 Semantic Annotator for Inkscape’s GUI . . . . .	204
4.14 Integration of SAI as an Inkscape extension . . . . .	205
4.15 Automatic namespace detection in SAI . . . . .	206
4.16 Automatic generation of graphic object instances in SAI . . . . .	207
5.1 General architecture of a Natural Language Interface . . . . .	216
5.2 Prototypical process flow of a system for question answering over knowledge bases . . . . .	219
5.3 High-level overview of AUDiaL in its problem context . . . . .	224
5.4 Client-server application model of AUDiaL . . . . .	231
5.5 Main elements of AUDiaL’s Web interface . . . . .	232

5.6	Source code of AUDiaL's dynamic result area . . . . .	234
5.7	Partial source code of AUDiaL's modal tabs and input areas . . . . .	235
5.8	Question Analysis phase of AUDiaL . . . . .	237
5.9	Result of the Question Analysis phase: a Parsed Query instance . . . . .	245
5.10	Phrase Mapping phase of AUDiaL . . . . .	246
5.11	Ontology Concept instance generation . . . . .	250
5.12	Disambiguation phase of AUDiaL . . . . .	253
5.13	Automatic consolidation of a Parsed Query object . . . . .	255
5.14	A disambiguation dialog . . . . .	256
5.15	Workflow for the generation of the label of an Ontology Concept to be displayed in a user dialog . . . . .	257
5.16	A mapping dialog . . . . .	259
5.17	A filter clarification dialog . . . . .	261
5.18	UML diagram of AUDiaL's disambiguation and learning model . . . . .	264
5.19	Workflow of AUDiaL's learning subsystem . . . . .	265
5.20	Task Execution Phase of AUDiaL . . . . .	267
5.21	The result of a Compare Task . . . . .	270
5.22	Finding anomalies in a 2-dimensional dataset via DBSCAN clustering . . . . .	274
5.23	Answer Generation phase of AUDiaL . . . . .	276
5.24	Example answer given when no formal task can be inferred from a user's query . . . . .	277
5.25	Example answer for a clustering task . . . . .	278
6.1	Resemblance preservation of a visual attribute in a dialog . . . . .	286
6.2	Navigation modal panel of AUDiaL . . . . .	291
6.3	Command Parser module of AUDiaL . . . . .	292
6.4	Example output of a sequential navigation between objects in AUDiaL . . . . .	293
6.5	Semantic annotation of a link diagram . . . . .	294
6.6	The answer to an input query in AUDiaL, showing several annotation techniques in use . . . . .	295
6.7	High-level overview (summary) of a diagram generated by AUDiaL . . . . .	300
6.8	General diagram shapes considered in AUDiaL's summaries . . . . .	302
6.9	Example output of a 'jump' navigational command between distant objects in AUDiaL . . . . .	304
6.10	Partial source code of AUDiaL's dialog tab panel area . . . . .	312
6.11	User permissions back-end of AUDiaL . . . . .	321
6.12	A running laboratory validation session . . . . .	322
6.13	Answer counts to the question "Which of these diagram types do you know about?" by validation participants . . . . .	331
6.14	Answer counts to the question "Where do you find diagrams in Web pages?" by validation participants . . . . .	332
6.15	Individual subjective personal interest of validation participants in STEM subjects . . . . .	333
		361

6.16 Analysis of task difficulties employing AUDiaL . . . . .	334
6.17 Comparison of task difficulty between AUDiaL and traditional counterparts . . . . .	335
6.18 Duration of tasks completion employing AUDiaL . . . . .	337
6.19 Comparison of task efficiency between AUDiaL and traditional counterparts . . . . .	338
6.20 Comparison of average NASA-TLX scores between AUDiaL and traditional counterparts . . . . .	339
6.21 Distributions of individual and aggregated NASA-TLX score differences . . . . .	340
6.22 Final SUS scores per participant . . . . .	341
6.23 Distribution of individual SUS scores . . . . .	342
6.24 Frequency of use of AUDiaL's accessibility features by participants . . . . .	343
6.25 Correlation between SUS scores and STEM interest among participants . . . . .	345

# List of Tables

1.1	Number of people visually impaired and corresponding percentage of the global impairment by WHO region and country, 2010 . . . . .	15
2.1	Limits of information bandwidth for three senses . . . . .	27
2.2	Support of metadata formats by image formats . . . . .	40
3.1	Transformations of the plane according to Wilkinson . . . . .	136
3.2	Area-fill visual attributes characterized in UVO . . . . .	140
3.3	Spatial visual attributes characterized in UVO . . . . .	142
3.4	Spatial object-to-object graphic relations characterized in VDO . . . . .	146
5.1	Template for design problems . . . . .	224
6.1	Summary of accessibility features in AUDiaL according to cognitive benefit	305
6.2	Evaluation of AUDiaL regarding relevant accessibility criteria of WCAG 2.1	311
6.3	Summary of effectiveness scores . . . . .	336
6.4	Summary of efficiency scores . . . . .	338
6.5	Problems detected in the current version of AUDiaL during its validation	346



# List of Algorithms

5.1	POC identification algorithm . . . . .	240
5.2	CQF identification algorithm . . . . .	242
5.3	Detecting potential CQF phrases . . . . .	243
5.4	Detecting potential CQF operand and operators . . . . .	244
5.5	Find distribution analytical task . . . . .	273
6.1	Computation of navigational order of graphic objects . . . . .	289
6.2	Overview of a statistical chart in natural language . . . . .	299
6.3	Overview of a metric axis . . . . .	299
6.4	Summary of labels present in a diagram . . . . .	301
6.5	Description of a bar chart's extrema in natural language . . . . .	303



# Glossary

**assistive technology** Any device, whether physical or digital, or combination of devices that is specifically designed to assist people with disabilities in performing some task. 14, 17, 22, 28, 57, 229, 230, 233, 282, 285, 305, 311, 313, 317, 330, 375

**controlled natural language** A subset of a natural language that has restricted grammar and vocabulary for eliminating ambiguities and complexity, either with the goal of making it easier to read by humans or simpler to process by natural language processing systems. 77

**domain ontology** An ontology made up of terms specific to a given domain of knowledge. Resources in a domain ontology are ranked under the terms of an upper ontology (e.g. classes in a domain ontology are sub-classes of classes in the upper ontology). For instance, the VDO is a domain ontology of diagram concepts. 75, 79, 216, 235, 246, 251, 257, 260, 261, 263, 265, 266, 287, 351

**GIF** Graphics Interchange Format; a raster image format commonly used on the WWW. 91, 176

**HTML** Hypertext Markup Language; the standard markup language for documents designed to be displayed in a web browser. 2, 28, 32, 39, 72, 83, 85–87, 99–101, 176, 177, 230, 232–234, 277, 311–314, 321, 381, 382

**JPG** JPG, or JPEG, is a very popular method used to compress image data, employed by a number of digital raster image formats, such as JPEG/JFIF. 91, 176

**MP3** a lossy coding format for digital audio, one of the most popular formats employed to create and share general audio content. 34

**natural language** Any human language that has evolved naturally through use and repetition without conscious planning or premeditation, such as English or American Sign Language (ASL). 2, 18, 29, 31, 38, 39, 62–66, 68–71, 77, 79, 81, 88, 89, 93, 98, 102, 118, 129, 139, 143, 155, 167, 187, 213–220, 222, 224, 225, 227–231, 233, 235, 239, 251, 252, 255, 265, 266, 268, 272, 276, 278–280, 282, 285, 291, 294, 297, 299, 301–303, 305, 318, 319, 344, 346, 350, 351, 353, 355–357, 367, 368

**natural language processing** a field that combines linguistics, computer science, and artificial intelligence in order to create computer programs that can analyze and process natural language data. 38, 79, 231, 367

**refreshable braille display** An electro-mechanical output device that displays braille characters, arranged in a tactile display, made up of individual dots that can be raised or lowered through piezoelectricity in order to update the displayed characters. 2, 28, 29, 35

**screen reader** A form of assistive technology (AT) that renders text and image content as speech or braille output, usually to a computer user. 2, 23, 28, 29, 33, 35, 36, 69, 82, 178, 227, 228, 230, 233, 234, 258, 297, 305, 311–314, 330

**semantically-enhanced diagram** A semantically-enhanced (c.f. semantically-enhanced graphic) diagrammatic representation of data. 27, 89, 123, 127, 133, 141, 153, 155, 171, 187, 188, 199, 202, 203, 209, 211, 214, 215, 218, 225, 229, 236, 251, 267, 279, 280, 285, 287, 288, 290, 292, 320, 329, 349, 353, 357

**semantically-enhanced graphic** An image formally enriched with the semantics of what it depicts by means of ontological resources mapped to its constituent graphic primitives. This enables client applications to perform inference and cognitive tasks at different conceptual levels by leveraging its semantic markup, instead of having to operate on the graphic content itself. Semantically-enhanced graphics are described in detail in Section 4.3. 20, 23, 114, 172, 175, 176, 187, 188, 194, 197, 200, 202, 203, 207–211, 230–232, 246, 258, 260–263, 266, 273, 276, 281, 289, 290, 299, 301, 303, 316, 352, 368, 377

**speech synthesizer** A form of assistive technology (AT) that reads aloud an input text using generated human-like speech sounds. It is currently the most employed assistive technology for the visually impaired, especially in virtual environments. 29, 33, 34, 37, 67, 217

**upper ontology** A foundational ontology which consists of terms common across all relevant sub-domains, supporting semantic interoperability among them by providing a common starting point for the formulation of definitions. For instance, the UVO is an upper ontology of visualization terms. 73, 75, 123, 130, 172, 367

**Visual Information-Seeking mantra** A widely adopted guideline to visual design proposed by Shneiderman in 1996, which states: “Overview first, zoom and filter, then details-on-demand” [Shn96]. This short mantra summarizes many recognized guidelines in information visualization and provides a memorable framework for the design of information visualization computer applications. 199, 272, 284, 296, 297, 303, 305, 347

**Web Content Accessibility Guidelines** A set of guidelines, developed through the W3C process in cooperation with individuals and organizations around the world, with a goal of providing a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally. 83, 305

**WordNet** A free and publicly available lexical database of semantic relations between words in English and other languages. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (*synsets*), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations<sup>4</sup>. 114, 129, 220, 238, 247, 251, 262

---

<sup>4</sup><https://wordnet.princeton.edu/>



# Acronyms

**AI** Artificial Intelligence. 64, 97, 113, 193

**API** Application Programming Interface. 99, 191, 231

**ARIA** Accessible Rich Internet Applications. 41, 84, 233, 234, 311–314

**AT** Assistive Technology. 18, 22, 27, 28, 31–35, 82, 91, 121, 215, 227, 330

**ATAG** Authoring Tool Accessibility Guidelines. 84

**AUDiaL** Accessible Universal Diagrams through Dialogue. 22, 23, 82, 121, 128, 129, 132, 165, 170, 187, 211, 213, 214, 218, 223–225, 228–239, 241, 242, 246, 247, 250–258, 260–266, 268–273, 275–282, 285–298, 300, 302, 304–323, 325–330, 333–347, 349, 353–357, 375, 377, 381, 383

**BTS** Braille translation software. 32

**CBIR** Content-Based Image Retrieval. 93, 114, 172

**CNN** Convolutional Neural Network. 195–197

**CQF** Cardinal Query Filter. 241–244, 246, 251–255, 261, 262, 266, 268, 269, 271, 272, 275, 279, 280

**CSS** Cascading Style Sheets. 176–178, 232

**CV** Computer Vision. 97, 193–196, 199, 200, 210

**CSV** Comma-Separated Values. 92

**D2T** data-to-text. 36, 37

**Daisy** Digital Accessible Information System. 34, 39

**DDO** Data Domain Ontology. 105, 123, 127–129, 158–163, 173, 179–181, 186, 211, 221, 351

**DTB** Digital Talking Book. 34

**EXIF** Exchangeable Image File Format. 40, 209

**FOAF** Friend-Of-A-Friend. 102, 104, 106, 200

**GUI** Graphical User Interface. 29, 35, 36, 50, 52, 62, 68, 133, 171, 188, 190, 204–206, 208, 214, 215, 232–234, 290, 291, 352

**HCI** Human-Computer Interaction. 9, 23, 50, 89, 164, 170, 214, 228, 241, 296, 319, 325, 330, 346, 347, 353, 356, 357

**HTTP** Hypertext Transfer Protocol. 99, 100, 231

**IR** Information Retrieval. 18, 62, 282

**IRI** Internationalized Resource Identifier. 111, 262

**KB** Knowledge Base. 216, 218–223, 227, 229–231, 235, 239, 241–243, 246–252, 254, 255, 258, 260–262, 265, 266, 268, 275, 279, 280

**MathML** Mathematical Markup Language. 34

**MIDI** Musical Instrument Digital Interface. 53

**MLE** Maximum Likelihood Estimation. 272, 273

**NASA-TLX** NASA Task Load Index. 326, 328, 329, 339, 340

**NLI** Natural Language Interface. xii, 2, 9, 19–23, 27, 62, 63, 66, 69, 82, 88, 89, 91, 96, 98, 114, 116, 121, 129, 170–172, 179, 196, 210, 211, 213–216, 218, 219, 221, 223, 225–227, 229, 231, 233, 235, 236, 246, 263, 268, 279–282, 284–290, 292, 294, 296–298, 300, 302, 304, 306, 308, 310, 312, 314, 316–318, 320–322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344–347, 349–357

**NLP** Natural Language Processing. 19, 39, 97, 228, 238, 239, 241, 243, 245, 250, 279, 291, 292, 325, 345, 355, 357

**NQF** Nominal Query Filter. 241, 242, 266–269, 271, 272, 275

**OC** Ontology Concept. 246, 249–264, 266, 268

**OM** Object Model. 92

**OoS** Object of Study. 315–319, 324, 325, 327, 356

- OWL** Web Ontology Language. 68, 103, 106–113, 122, 126, 127, 130, 160, 179, 183, 191, 200, 356
- PDF** Portable Document Format. 32, 65
- POC** Potential Ontology Concept. 237–241, 243, 246, 249, 251–255, 258–262, 279, 280
- PoS** Part-of-Speech. 236–238, 254
- PT** Parse Tree. 236–239, 245, 247, 256, 279
- QA-KG** Question-Answering over Knowledge Graphs. 219, 222, 352
- QF** Query Filter. 237, 238, 241, 252
- RDF** Resource Description Framework. 70, 77, 96, 100–116, 120, 164, 178, 179, 181, 187, 191, 200, 204, 209, 231, 373
- RDFS** RDF Schema. 103–107, 109, 112, 200, 201
- RL** Reinforcement Learning. 63, 263
- RNN** Recurrent Neural Network. 39
- SAI** Semantic Annotator for Inkscape. 23, 131, 134, 141, 145, 147, 175, 186, 188, 191, 202–209, 211, 251, 315, 320, 349, 352, 353, 357, 377
- SCM** Semantic Color Model. 74, 140
- SPARQL** SPARQL Protocol and RDF Query Language. 116, 219–222, 227, 248, 266, 287
- STEM** science, technology, engineering, and mathematics. 16, 322, 332, 333, 344, 345, 347, 354
- SUS** System Usability Scale. 325, 326, 340–342, 344, 345
- SVG** Scalable Vector Graphics. 35, 40, 41, 60, 72, 73, 75–77, 85, 87, 132, 141, 142, 144, 145, 147, 154, 175–192, 202–211, 230, 231, 233, 288, 352
- TLA** Three Layer Approach. 121–123
- TTS** text-to-speech. 28, 33–35, 68, 87, 230, 330
- UAAG** User Agent Accessibility Guidelines. 84
- UML** Unified Modeling Language. 17
- URI** Uniform Resource Identifier. 99–104, 107, 108, 111, 113, 115, 116, 222, 248, 352

- UVÖ** Upper Visualization Ontology. 105, 108, 120, 123–126, 128–131, 133–135, 137–144, 147, 148, 150, 152, 158, 172, 173, 179, 180, 185, 187, 191, 203, 204, 207, 208, 211, 221, 286, 289, 346, 351, 352, 368
- VDO** Visualization Domain Ontology. 120, 123–126, 128–131, 143–146, 149, 153–158, 173, 179–181, 186, 187, 208, 211, 221, 351, 355, 367
- VIP** Visually Impaired Person. 17, 18, 22, 23, 28, 31–34, 37, 49, 55, 56, 58, 63, 66, 69, 71, 72, 74, 81, 82, 87, 88, 91, 121, 124, 128, 197, 214, 215, 223, 229, 273, 281, 346, 353
- VQA** Visual Question Answering. 39, 64–66, 88, 172, 194, 195, 359
- VTO** Visualization Tasks Ontology. 123, 127–129, 163–168, 171, 173, 187, 225, 228, 239, 241, 251, 256, 257, 260, 262, 263, 266–273, 280, 282, 285, 289, 293, 347
- W3C** World Wide Web Consortium. 29, 83–85, 100, 103, 109, 176, 178, 233, 313, 321, 369
- WAI** Web Accessibility Initiative. 41, 83, 84, 233
- WCAG** Web Content Accessibility Guidelines. 83, 84, 227, 233, 305–313
- WHO** World Health Organization. 14
- WWL** What-Where Language. 77, 78, 81
- WWW** World Wide Web. 2, 3, 36, 41, 62, 82, 92, 95, 99, 100, 176, 228, 323, 330, 332, 367
- XML** Extensible Markup Language. 34, 68, 72, 92, 102, 110, 113, 176, 178, 179, 181, 191, 352

# Appendix A: Instructions to Participants

AUDiaL (c.f. Chapter 5) was validated with target users in laboratory sessions equipped with personal computers, an Internet connection, and suitable assistive technology, as described in Chapter 6. AUDiaL could be accessed by participants via an accessible Web application. This appendix contains the contents of the homepage of said Web application, which informed participants about the upcoming session. Note that the order of assigned tasks and the methods for accessing each diagram could vary depending on the user account according to research setup. What follows is an example homepage for a given user.

## Welcome, <username>

Thank you for taking part in this workshop! In the next few hours we will try out and compare different methods for accessing diagrams, such as bar charts, in a non-visual manner. The main focus of the workshop consists on the evaluation of a web application in which diagrams can be queried through natural language. Natural language is the normal speech you use daily to communicate with other persons. However, the web application you are about to try is a prototype, not a finished product, and as such it currently only understands a small portion of the English language.

The goal of the evaluation is to better understand whether natural language could be used in a satisfactory manner as an alternative to other existing methods for accessing diagrams that you may already be familiar with, such as static descriptions in text and tables, and tactile graphics. You will be asked to perform a few tasks on three different diagrams, from lower to higher complexity (more complex diagrams display a higher amount of information than simpler diagrams). Afterwards, we will ask you about your opinion regarding each method's efficiency, ease of use, and how likely you would be to use them in your daily life.

Please note that, even if you will be asked several questions throughout the workshop, we are not testing you! Instead, with your help, we are testing the methods you will be using to access diagrams non-visually. We are interested in your opinion, which will be very valuable in order to improve these methods in the future. You are not expected to be able to answer all questions, and most likely you will not! Please remember those things that you found confusing or unclear, and, if you have any doubts, simply ask.

## Workshop Structure

First, you will be asked to complete a short questionnaire about yourself, containing basic information such as your age, computer skills, etc. This data is completely anonymous and you will never be asked to add any information that may identify you.

After filling in the questionnaire, three different diagrams about completely different topics will be given to you in some alternative form. For each of them, you will be asked a few questions of different difficulty levels. Some questions require you to find a single item in the diagram (for example: how many people lived in the city of Vienna in 2014?), whereas others require you to combine and reason about several different data points (for example: what was the trend of the population in Vienna between 1994 and 2014?). There is no time limit for answering any question.

Again, if you have any questions just ask!

## Workshop Steps

These are the steps we will go through during this workshop. You can use them as a guide if at some point you are not sure how to proceed:

1. Please fill in the personal questionnaire <[link](#)> first.
2. The first diagram we will explore is about Leonardo DiCaprio's girlfriends. Open your assigned version of the diagram (<[link](#)> - opens in a new window). Then, take a look at the tasks that you will perform on this chart (tasks form <[link](#)> - opens in new window), and complete them using your assigned method.
3. If you completed the previous tasks using the dialogue system, please fill in the feedback form (feedback form <[link](#)> - opens in new window)
4. Next, we will explore a bar chart about the population in Austria (<[link](#)> - opens in a new window). Take a look at the tasks that you will perform on this chart (tasks form <[link](#)> - opens in new window), and complete them using your assigned method.
5. If you completed the previous tasks using the dialogue system, please fill in the feedback form (feedback form <[link](#)> - opens in new window)
6. The last diagram we will explore is about energy generation in Europe (<[link](#)> - opens in a new window). Take a look at the tasks that you will perform on this chart (tasks form <[link](#)> - opens in new window), and complete them using your assigned method.
7. If you completed the previous tasks using the dialogue system, please fill in the feedback form (feedback form <[link](#)> - opens in new window)

## **Appendix B: Validation Diagrams - Images**

This appendix shows the three statistical charts that were selected to be employed in validation of AUDiaL, as described in Section 6.10.3. These diagrams were converted to semantically-enhanced graphics with Semantic Annotator for Inkscape (SAI) (c.f. Section 4.5) and uploaded to a running instance of AUDiaL. Blind participants were asked to perform analytical tasks supported by these diagrams and their underlying data as part of the validation sessions described in Section 6.11.

## Diagram 1

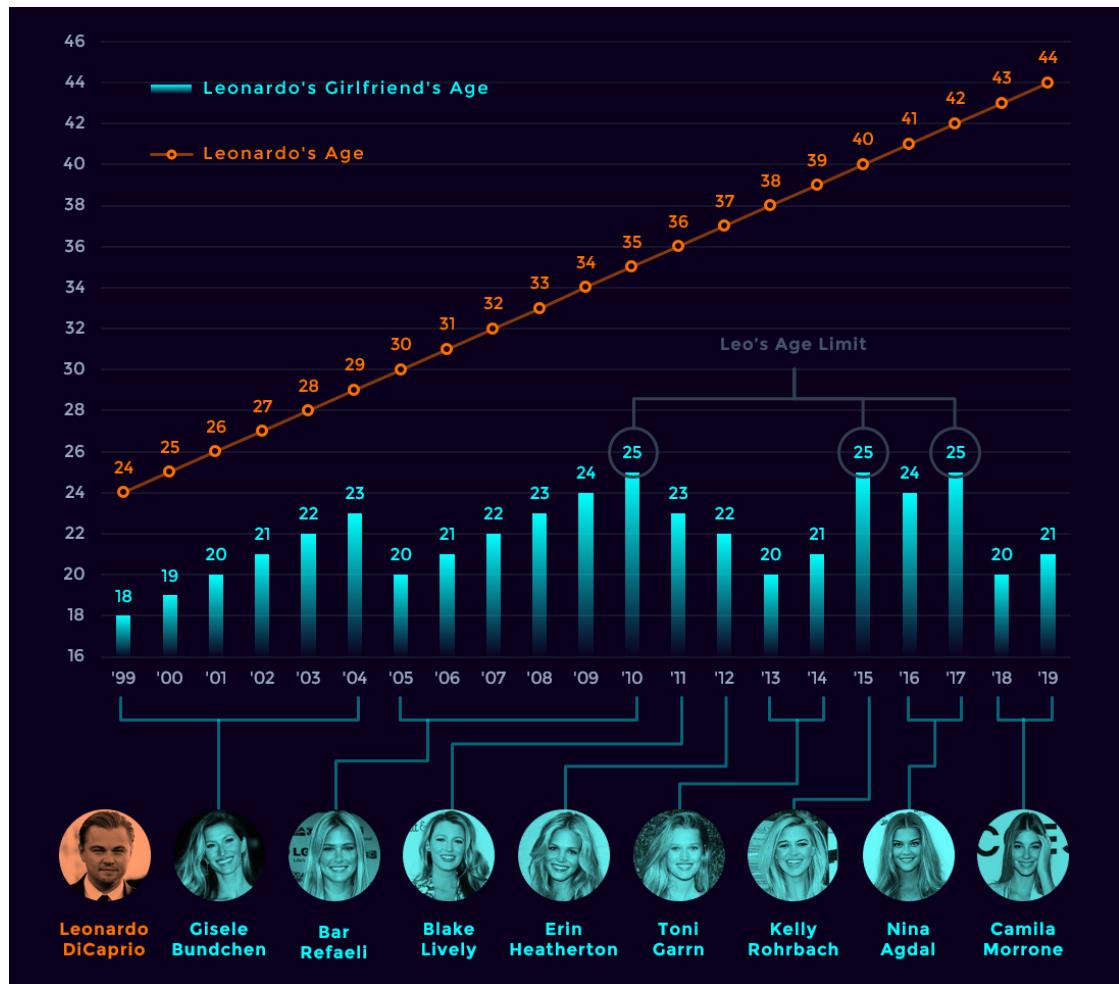


Figure 1: *Diagram 1* - Leonardo DiCaprio's girlfriends throughout the years  
(source: [https://www.reddit.com/r/dataisbeautiful/comments/azjti7/leonardo\\_dicaprio\\_refuses\\_to\\_date\\_a\\_woman\\_over\\_25/](https://www.reddit.com/r/dataisbeautiful/comments/azjti7/leonardo_dicaprio_refuses_to_date_a_woman_over_25/))

**Diagram 2**

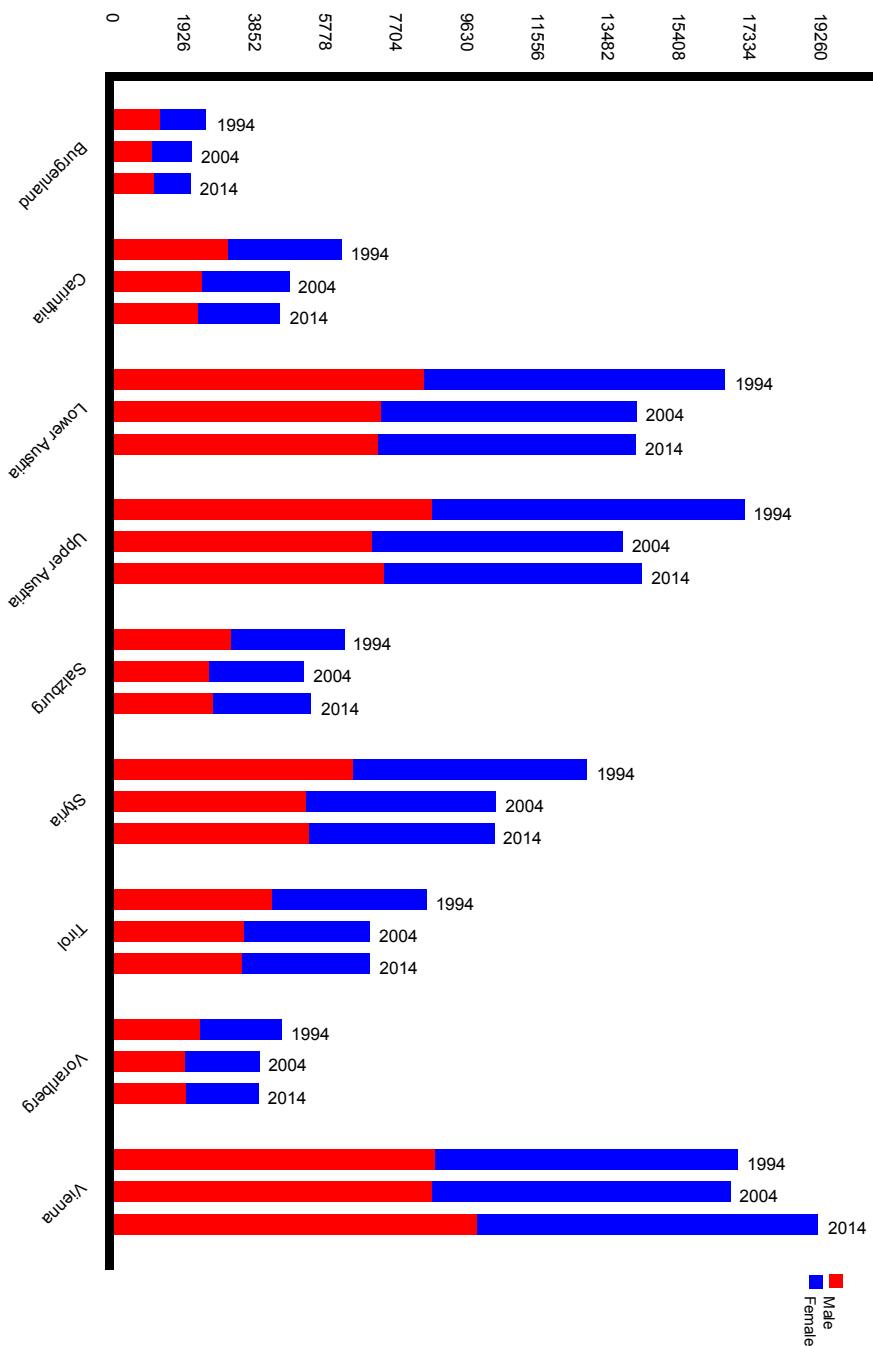


Figure 2: *Diagram 2* - Evolution of the population of Austria by region and sex (own creation; rotated)

Diagram 3

Power generation by source (2000-2018)

■ coal ■ other fossil fuels ■ nuclear  
■ renewables

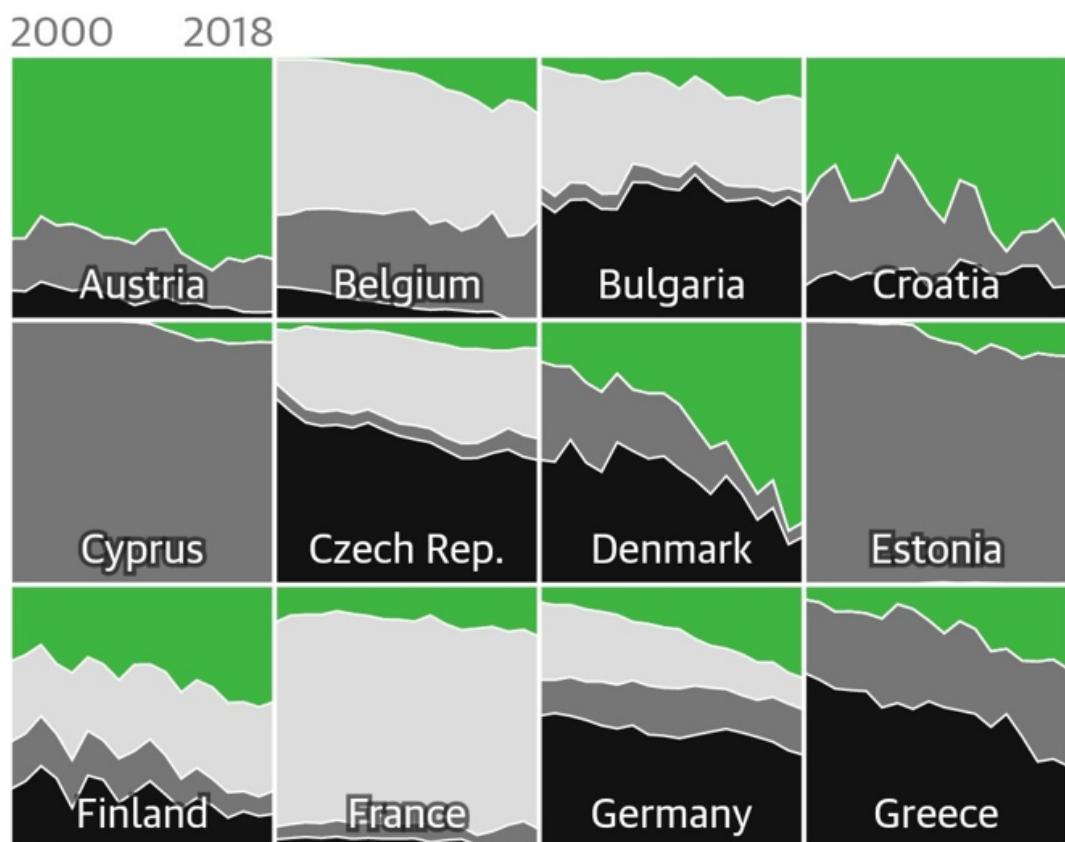


Figure 3: *Diagram 3 - Power Generation by Source in selected European countries 2001-2018* (source: <https://www.derstandard.at/story/2000105904484/kohlekraftwerke-in-europa-diese-laender-und-konzerne-verpesten-die-luft>)

# Appendix C: Validation Diagrams

## - Long descriptions

The alternative long descriptions for the diagrams used during validation of AUDiaL, listed in Appendix B, are shown in this appendix. Note that their data tables are omitted for brevity. For more information on alternatives to diagrams in the context of AUDiaL's validation see Section 6.11.

### Diagram 1

#### Overview

This chart displays the age of actor Leonardo DiCaprio with each passing year from 1999 to 2019. Moreover, the name of DiCaprio's girlfriend on a given year is displayed along with her age. For example, the chart shows how in 1999 DiCaprio's age was 24, and he was dating Gisele Bundchen, age 18.

#### Values

Numerical values presented on the image: <HTML table omitted>

#### Presentation

This figure consists on a bar chart and a line chart arranged along two shared metric axes. The vertical axis measures a person's age starting from 16 to a maximum of 46 years of age. The horizontal axis measures time by displaying years starting from 1999 until 2019. The line chart consists on a single orange line displaying Leonardo DiCaprio's age as the years pass. The bar chart consists on a number of labelled blue metric bars, one per year, displaying Leonardo's girlfriend's age on a given year. Each bar is labelled with the name of Leonardo's girlfriend that year along with a picture of her.

## Diagram 2

### Overview

This bar chart shows the population of Austria divided into states, years, and sex. For each Austrian state, the population for the years 1994, 2004, and 2014 is shown in a stacked bar. Each stacked bar is further divided into two metric bars, namely, the male and female population for a given state and year is displayed.

### Values

Numerical values presented on the image: <HTML table omitted>

### Presentation

This bar chart is made up of several grouped stacked bars. Each group of bars corresponds to one Austrian state and consists of a line-up of three bars, one per year (1994, 2004, and 2014). Each stacked bar is divided into a blue bar which corresponds to the male population, and a red bar displaying the female population for its corresponding state and year. The metric axis to the left of the bars is labelled from 0 to 1926000, indicating number of people.

## Diagram 3

### Overview

This chart shows the energy generation by source in selected European countries. For each country, the percentage of generated energy according to its source from 2002 to 2018 is shown. Energy sources may be one of the following: renewable, nuclear, coal, and other fossil fuels.

### Values

Numerical values presented on the image: <HTML table omitted><sup>1</sup>

### Presentation

This chart is made up of a number of individual area charts, one per country. Each area chart is divided into energy sources on the y-axis and individual years on the x-axis such that the total area for each year adds up to 100%.

---

<sup>1</sup>To allow for a fair comparison between *Diagram 2* and *Diagram 3*, the table corresponding to *Diagram 3* was simplified from the original data depicted by the diagram so that it contained a total of 102 data points.

# Appendix D: Validation Tasks

This appendix lists the analytical tasks that participants were asked to perform on the given diagrammatically-represented data in validation sessions. For each diagram, eight tasks of varying difficulty were prescribed. Participants could try to solve these tasks using either AUDiaL or a diagram counterpart of their choice (tactile diagram or long description), as described in Section 6.11. The same tasks were given to all participants for each diagram regardless of access method.

## Diagram 1

- *Task 1:* What do the values of this diagram's bars (for example, the number 24) represent?. Hint: if using the dialogue-enabled diagram, you can ask the system for a summary.
- *Task 2:* Very generally, what does this diagram depict?
- *Task 3:* How old was Leonardo DiCaprio in 2009? And in 1998?
- *Task 4:* Who was Leonardo's girlfriend when he was 43 years old? How old was she? Hint: if using the dialogue system, you will have to divide this query into two simpler ones.
- *Task 5:* Write the names of the oldest girlfriends that DiCaprio has had. How old were they?
- *Task 6:* How many years has lasted DiCaprio's longest relationship(s)? Is there any relation between DiCaprio's age and the length of his relationships? Hint: navigate the diagram bar by bar and keep track of when the name of DiCaprio's girlfriend changes.
- *Task 7:* What is the average age difference between DiCaprio and his girlfriends? Hint: if using the dialogue system, you will have to divide this query into two simpler ones and compare their results, or you can navigate the bars one by one.
- *Task 8:* Group DiCaprio's girlfriends according to their age and write their first names for each age. For example, he had three girlfriends aged 25: Bar, Kelly, and Nina.

## Diagram 2

- *Task 1:* How many bars are there in the graph? What is the difference between a stacked bar and a metric bar?
- *Task 2:* Very generally, what does this diagram depict?
- *Task 3:* Write the two Austrian regions having the maximum and minimum population in 2004, respectively.
- *Task 4:* Navigate, one by one, through all the stacked bars in the diagram. Write down the values of the first and the last bar that you see.
- *Task 5:* Write the three regions having a total population smaller than 500000 people in 2014.
- *Task 6:* How many more women lived in Vienna in 2014 compared to 2004? Hint: if using the dialogue-enabled system, set one of the bars as your home bar.
- *Task 7:* Which regions increased in population between 2004 and 2014?
- *Task 8:* What would you say is the general trend of the population in Austria? Is it growing or decreasing with each passing decade? Where are people moving to?

## Diagram 3

- *Task 1:* How many countries are depicted in this diagram? Hint: if using the dialogue-enabled diagram, you can ask the system for a summary (it can take a few seconds to compute).
- *Task 2:* Very generally, what does this diagram depict?
- *Task 3:* What was the energy generation of Germany in 2018 like?
- *Task 4:* Which countries generate no nuclear power at all?
- *Task 5:* How did the energy generation of Denmark evolve with the years? Hint: navigate the bars one by one, starting with the bar labeled (Denmark, coal, 2002), and compare their values
- *Task 6:* How does the nuclear energy generation of France compare to that of the remaining countries?
- *Task 7:* Group countries according to their generation of renewable energies in 2018 into three categories: high generation, medium generation, low generation.
- *Task 8:* Which country's energy generation looks the most similar to that of Austria?

# Appendix E: Publications

The work presented within this thesis is partially published in the following papers:

- Tomás Murillo-Morales and Klaus Miesenberger. Formalizing visualization semantics for accessibility. In Katsuhito Yamaguchi, editor, *Proceedings of the 4th International Workshop on Digitization and E-Inclusion in Mathematics and Science 2021*, pages 47–56, Tokyo, Japan, 2021. Nihon University
- Tomás Murillo-Morales and Klaus Miesenberger. Audial: A natural language interface to make statistical charts accessible to blind persons. In Klaus Miesenberger, Roberto Manduchi, Mario Covarrubias Rodriguez, and Petr Peňáz, editors, *Computers Helping People with Special Needs*, Lecture Notes in Computer Science, pages 373–384. Springer, Cham, 2020
- Tomás Murillo-Morales, Jaromír Plhák, and Klaus Miesenberger. Authoring semantic annotations for non-visual access to graphics. *The Journal on Technology and Persons with Disabilities*, 6:398–413, 2018
- Tomás Murillo-Morales and Klaus Miesenberger. Techniques for improved speech-based access to diagrammatic representations. In Klaus Miesenberger and Georgios Kouroupetroglou, editors, *Computers Helping People with Special Needs. 16th International Conference*, Lecture Notes in Computer Science, pages 636–643. Springer, 2018
- Tomás Murillo-Morales and Klaus Miesenberger. Non-visually performing analytical tasks on statistical charts. In P. Cudd and L. de Witte, editors, *Studies in health technology and informatics*, volume 242, pages 339–346. IOS Press, 2017
- Tomás Murillo-Morales and Klaus Miesenberger. Ontology-based semantic support to improve accessibility of graphics. In C. Sik-Lányi, E.-J. Hoogerwerf, and K. Miesenberger, editors, *Studies in health technology and informatics*, volume 217, pages 255–260. IOS Press, 2015



# Bibliography

- [AAL<sup>+</sup>15] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433. 2015.
- [AES05] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, page 15, Washington, DC, USA, 2005. IEEE Computer Society.
- [AL09] Chieko Asakawa and Barbara Leporini. Screen readers. In Constantine Stephanidis, editor, *The universal access handbook*, Human factors and ergonomics. CRC Press, Boca Raton, 2009.
- [Ala13] E. S. Alatrish. Comparison of ontology editors. *Journal of Management Information Systems*, 8(2):18–24, 2013.
- [ALG] Phil Archer, Nikolaos Loutas, and Stijn Goedertier. Cookbook for translating relational data models to rdf schemas.
- [All99] Gary L. Allen. Cognitive abilities in the service of wayfinding: A functional approach. *The Professional Geographer*, 51(4):555–561, 1999.
- [All08] User agent accessibility guidelines (uaag) 2.0, 2008.
- [AOS05] H. Alani, K. O'Hara, and N. Shadbolt. Common features of killer apps: A comparison with protégé. In *8th International Protégé Conference*, 2005.
- [APS<sup>+</sup>10] Iyad Abu Doush, Enrico Pontelli, Tran Cao Son, Dominic Simon, and Ou Ma. Multimodal presentation of two-dimensional charts: An investigation using open office xml and microsoft excel. *ACM Transactions on Accessible Computing (TACCESS)*, 3(2):8, 2010.
- [AR98] James L. Alty and Dimitrios I. Rigas. Communicating graphical information to blind users using music: the role of context. In *CHI '98 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 574–581, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

- [Arn90] Rudolf Arnheim. Perceptual aspects of art for the blind. *Journal of Aesthetic Education*, 24(3):57, 1990.
- [AT13] Alexander Andreopoulos and John K. Tsotsos. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8):827–891, 2013.
- [ATdO<sup>+</sup>18] Souza Alves, Thamiris de, Caterine Silva de Oliveira, Cesar Sanin, and Edward Szczerbicki. From knowledge based vision systems to cognitive vision systems: A review. *Procedia Computer Science*, 126:1855–1864, 2018.
- [AW08] Kerstin Altmanninger and Wolfram Wöß. Accessible graphics in web applications: Dynamic generation, analysis and verification. In Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler, and Arthur Karshmer, editors, *Computers Helping People with Special Needs*, pages 378–385, Berlin, Heidelberg, 2008. Springer-Verlag Berlin Heidelberg.
- [AYB11] Alaeldin A. Ahmed, Mustafa A. A. Yasin, and Sharief F. Babiker. Tactile web navigator device for blind and visually impaired people. In *2011 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies*, pages 1–5, Piscataway, N.J., 2011. IEEE.
- [Bab97] Christopher Baber. *Beyond the desktop: Designing and using interaction devices*. Computers and people series. Academic, San Diego and London, 1997.
- [Bar67] Roland Barthes. *Elements of Semiology*. NoonDay Press, New York, 1967.
- [BBR<sup>+</sup>03] L. M. Brown, S. A. Brewster, S. A. Ramloll, R. Burton, and B. Riedel. Design guidelines for audio presentation of graphs and tables. In *Proceedings of the 9th International Conference on Auditory Display*, Boston, MA, 2003. Boston University.
- [BCB<sup>+</sup>07] Jeffrey P. Bigham, Anna C. Cavender, Jeremy T. Brudvik, Jacob O. Wobbrock, and Richard E. Ladner. Webinsitu: a comparative analysis of blind and sighted browsing behavior. In *Assets '07 Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, pages 51–58, New York, NY, USA, 2007. ACM.
- [BCH98] S. A. Brewster, A. Capriotti, and C. V. Hall. Using compound earcons to represent hierarchies. *HCI Letters*, 1(1):6–8, 1998.
- [BE98] P. Blenkhorn and D. G. Evans. Using speech and touch to enable blind people to access schematic diagrams. *Journal of Network and Computer Applications*, 21(1):17–29, 1998.
- [Bec04] Owl web ontology language reference: W3c recommendation 10 february 2004, 2004.

- [BELP13] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. Graph markup language (graphml). In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, Discrete Mathematics and Its Applications, pages 517–541. CRC Press, Hoboken, 2013.
- [Ben02] David J. Bennett. Effects of navigation and position on task when presenting diagrams to blind people using sound. In Mary Hegarty, Bernd Meyer, and N. Hari Narayanan, editors, *Diagrammatic representation and inference*, volume 2317 of *Lecture notes in artificial intelligence*, 0302-9743, pages 161–175. Springer, Berlin and London, 2002.
- [Ber11] Jacques Bertin. *Semiology of graphics: Diagrams, networks, maps / Jacques Bertin ; translated by William J. Berg*. ESRI and London, Redlands, Calif., 1st ed. edition, 2011.
- [BFJ<sup>+</sup>11] Emily C. Bouck, Sara Flanagan, Gauri S. Joshi, Waseem Sheikh, and Dave Schleppenbach. Speaking math—a voice input, speech output calculator for students with visual impairments. *Journal of Special Education Technology*, 26(4):1–14, 2011.
- [BGB94] W. Buxton, W. Gaver, and S. Bly. Auditory interfaces: the use of non-speech audio at the interface, 1994.
- [BH10] Hichem Bannour and Celine Hudelot. Towards ontologies for image interpretation and annotation. In *9th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 211–216, Piscataway, NJ, 2010. IEEE.
- [BJJ<sup>+</sup>10] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. Vizwiz: Nearly real-time answers to visual questions. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM.
- [BJL93] Malcolm I. Bauer and P. N. Johnson-Laird. How diagrams can improve reasoning. *Psychological Science*, 4(6):372–378, 1993.
- [BJM<sup>+</sup>10] Jeffrey P. Bigham, Chandrika Jayant, Andrew Miller, Brandyn White, and Tom Yeh. Vizwiz::locateit - enabling blind people to locate objects in their environment. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 65–72. IEEE, 2010.
- [BKK<sup>+</sup>77] Daniel G. Bobrow, Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson, and Terry Winograd. Gus, a frame-driven dialog system. *Artificial Intelligence*, 8(2):155–173, 1977.
- [Bli07] Blinden-und Sehbehindertenverband Österreich. Wie viele menschen mit sehbehinderungen gibt es in österreich?, 2007.

- [BLJL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 5(284):28–37, 2001.
- [BLK<sup>+</sup>09] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.
- [BME<sup>+</sup>07] Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, and Kelli A. Houston. *Object-oriented analysis and design with applications*. The Addison-Wesley object technology series. Addison-Wesley, Upper Saddle River, NJ and Harlow, 3rd ed. edition, 2007.
- [BMS08] Thimothy Barbieri, Lorenzo Mosca, and Licia Sbattella. Learning math for visually impaired users. In Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler, and Arthur Karshmer, editors, *Computers Helping People with Special Needs*, volume 5105 of *Lecture Notes in Computer Science*, pages 907–914. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [BPM09] Olivier Bau, Uros Petrevski, and Wendy Mackay. Bubblewrap: a textile-based electromagnetic haptic display. In *Extended Abstracts on Human Factors in Computing Systems*, pages 3607–3612. ACM, New York, NY, USA, 2009.
- [BPS03] Andy Brown, Steve Pettifer, and Robert Stevens. Evaluation of a non-visual molecule browser. In Julie A. Jacko and Andrew Sears, editors, *Assets '04 Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, pages 40–47. ACM, New York, NY, USA, 2003.
- [Bra09] Namespaces in xml 1.0 (third edition), 2009.
- [Bre02] S. A. Brewster. Nonspeech auditory output. In Julie A. Jacko and Andrew Sears, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, Human factors and ergonomics, pages 220–239. Lawrence Erlbaum Associates, Mahwah, New Jersey, 2002.
- [Bri14] Rdf schema 1.1, 2014.
- [Bro71] Roy J. Brothers. Aural study systems for the visually handicapped: Effects of message length. *Education of the Visually Handicapped*, 3(3):65–70, 1971.
- [Bro08] Andrew Jeffrey Brown. *Non-Visual Interaction with Graphs*. Phd thesis, University of Manchester, UK, 2008.
- [Bro13] John Brooke. Sus: A retrospective. *J. Usability Studies*, 8(2):29–40, 2013.
- [BSP13] Andy Brown, Robert Stevens, and Steve Pettifer. Making graph-based diagrams work in sound: The role of annotation. *Human–Computer Interaction*, 28(3):193–221, 2013.

- [BSP<sup>+</sup>19] Sandra Bardot, Marcos Serrano, Simon Perrault, Shengdong Zhao, and Christophe Jouffrais. Investigating feedback for two-handed exploration of digital maps without vision. In David Lamas, editor, *17th IFIP TC 13 International Conference on Human-computer interaction*, volume 11746 of *LNCS sublibrary: SL3 - Information systems and applications, incl. Internet/Web, and HCI*, pages 305–324. Springer, Cham, Switzerland, 2019.
- [BTM<sup>+</sup>20] Emeline Brulé, Brianna J. Tomlinson, Oussama Metatla, Christophe Jouffrais, and Marcos Serrano. Review of quantitative empirical evaluations of technology for people with visual impairments. In Regina Bernhaupt, Florian 'Floyd' Mueller, David Verweij, Josh Andres, Joanna McGrenere, Andy Cockburn, Ignacio Avellino, Alix Goguey, Pernille Bjørn, Shengdong Zhao, Briane Paul Samson, and Rafal Kocielnik, editors, *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, New York, NY, USA, 2020. ACM.
- [BWE] S. Brewster, P. C. Wright, and A. D. Edwards. The application of a method for integrating non-speech audio into human-computer interfaces.
- [CCLX06] Matt Calder, Robert F. Cohen, Jessica Lanzoni, and Yun Xu. Plumb: an interface for users who are blind to display, create, and modify graphs. In *Assets '06 Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*. 2006.
- [CCS01] Peter Cheng, James Cupit, and Niger R. Shadbolt. Supporting diagrammatic knowledge acquisition: An ontological analysis of cartesian graphs. *International Journal of Human-Computer Studies*, 54(4):457–494, 2001.
- [CD15] Christopher Clark and Santosh Divvala. Looking beyond text: Extracting figures, tables and captions from computer science papers. In *AAAI Workshop: Scholarly Big Data*. 2015.
- [CED06] Sandra Carberry, Stephanie Elzer, and Seniz Demir. Information graphics: an untapped resource for digital libraries. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2006. ACM.
- [CF15] Vítor Carvalho and Diamantino Freitas. Converting svg images to text and speech. In Georgios Kouroupetroglou, editor, "Enabling Access for Persons with Visual Impairment", *Proceedings of the International Conference ICEAPVI-2015*, 2015.
- [Cha10] Ajay Chakravarthy. *Towards cross media document annotation*. Doctoral dissertation, University of Sheffield, 2010.

- [CHH07] Philipp Cimiano, Peter Haase, and Jörg Heizmann. Porting natural language interfaces between domains. In Angel Puerta and Tessa Lau, editors, *IUI 2007*, page 180, New York, N.Y., 2007. Association for Computing Machinery.
- [CJP<sup>+</sup>19] Jinho Choi, Sanghun Jung, Deok Gun Park, Jaegul Choo, and Niklas Elmquist. Visualizing for the non-visual: Enabling the visually impaired to use visualization. *Computer Graphics Forum*, 38(3):249–260, 2019.
- [CL98] Marc Corio and Guy Lapalme. Integrated generation of graphics and text: a corpus study. In *Proceedings of the Association of Computational Linguistics Workshop on Content Visualisation and Intermedia Representation*, pages 63–68, 1998.
- [CL99] Marc Corio and Guy Lapalme. Generation of texts for information graphics. In *Proceedings of the 7th European Workshop on Natural Language Generation EWNLG'99*, pages 49–58, 1999.
- [CL17] S. Cox and C. Little. Time ontology in owl, 2017.
- [Cle94] William S. Cleveland. *The elements of graphing data*. AT&T Bell Laboratories, Murray Hill, N.J., rev. ed. edition, 1994.
- [CM84] John Carroll and R. Mack. Learning to use a word processor: By doing, by thinking, and by knowing. In *Human factors in computer systems*, pages 13–51, 1984.
- [CM85] W. S. Cleveland and R. McGill. Graphical perception and graphical methods for analyzing scientific data. *Science (New York, N.Y.)*, 229(4716):828–833, 1985.
- [CMB<sup>+</sup>14] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanović, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, Wim Peters, and Leon Derczynski. Developing language processing components with gate: Version 8, 2014.
- [CMK<sup>+</sup>13] Sagnik Ray Choudhury, Prasenjit Mitra, Andi Kirk, Silvia Szep, Donald Pellegrino, Sue Jones, and C. Lee Giles. Figure metadata extraction from digital documents. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, pages 135–139. IEEE Computer Society, Washington, DC, USA, 2013.
- [Col99] Michael Collins. *Head-driven statistical models for natural language parsing*. Phd thesis, University of Pennsylvania, 1999.
- [Coo06] Wai-aria overview, 2006.
- [Cou07] Council of the European Union. Full text of the treaty of lisbon, 2007.

- [CPK<sup>+</sup>98] C. Colwell, H. Petrie, D. Kornbrot, A. Hardwick, and S. Furner. Use of a haptic device by blind and sighted people: perception of virtual textures and objects. In I. B. Placencia and E. Porrero, editors, *Improving the Quality of Life for the European Citizen: Technology for Inclusive Design and Equality*. IOS Press, 1998.
- [Cuk10] K. Cukier. Show me: New ways of visualising data. *The Economist*, February 2010.
- [Cyg14] Rdf 1.1 concepts and abstract syntax: W3c recommendation 25 february 2014, 2014.
- [DAC10a] Danica Damljanović, Milan Agatonović, and Hamish Cunningham. Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), 2010.
- [DAC10b] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In Lora Aroyo, editor, *The semantic web: research and applications*, volume 6088 of *Lecture Notes in Computer Science*, pages 106–120. Springer, Berlin, 2010.
- [DAC12] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Freya: An interactive way of querying linked data using natural language. In Raúl García Castro, Dieter Fensel, and Grigoris Antoniou, editors, *The semantic web: ESWC 2011 workshops*, Lecture Notes in Computer Science, pages 125–138, Heidelberg, 2012. Springer.
- [DACP13] Danica Damljanović, Milan Agatonović, Hamish Cunningham, and Kalina Bontcheva. Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. *Web Semantics: Science, Services and Agents on the World Wide Web*, 19:1–21, 2013.
- [Dam11] Danica Damljanović. *Natural Language Interfaces to Conceptual Models*. Phd thesis, University of Sheffield, 2011.
- [DB74] J. A. Desor and Gary K. Beauchamp. The human capacity to transmit olfactory information. *Perception & Psychophysics*, 16(3):551–556, 1974.
- [DCM12a] DCMI Usage Board. Dcmi: Dcmi metadata terms, 2012.
- [DCM12b] Seniz Demir, Sandra Carberry, and Kathleen F. McCoy. Summarizing information graphics textually. *Computational Linguistics*, 38(3):527–574, 2012.

- [DDS<sup>+</sup>09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Los Alamitos, California, 2009. IEEE.
- [DEM11] Nazan Dogruer, Ramadan Eyyam, and Ipek Menevis. The use of the internet for educational purposes. *Procedia - Social and Behavioral Sciences*, 28:606–611, 2011.
- [DFVR10] Michel Dumontier, Leo Ferres, and Natalia Villanueva-Rosales. Modeling and querying graphical representations of statistical data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2-3):241–254, 2010.
- [DGL<sup>+</sup>11] Stamatia Dasiopoulou, Eirini Giannakidou, Georgios Litos, Polyxeni Malasioti, and Yiannis Kompatsiaris. A survey of semantic image and video annotation tools. In Georgios Paliouras, Constantine D. Spyropoulos, and G. Tsatsaronis, editors, *Knowledge-driven multimedia information extraction and ontology evolution*, Lecture notes in computer science. State-of-the-art survey, pages 196–239. Springer, Heidelberg, 2011.
- [DIA11] DIAGRAM Center. Potential use of image description metadata for accessibility, 2011.
- [DIA16] DIAGRAM Center. Potential use of image-description metadata for accessibility: Addendum: Standards and recommendations, metadata, screen readers, tools, guidelines, 2016.
- [Dic11] Lutz Dickmann, editor. *Smart graphics: 11th International Symposium on Smart Graphics, SG 2011, Bremen, Germany, July 18-20, 2011 : proceedings / Lutz Dickmann ... [et al.] (eds.)*, volume 6815 of *LNCS sublibrary. SL 6, Image processing, computer vision, pattern recognition and graphics*. Springer, Heidelberg, 2011.
- [DK08] Freitas Diamantino and Georgios Kouroupetroglou. Speech technologies for blind and low vision persons. *Technology and Disability*, 20(2):135–156, 2008.
- [DL06] Aijuan Dong and Honglin Li. Multi-ontology based multimedia annotation for domain-specific information retrieval. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 158–165, Piscataway N.J., 2006. IEEE.
- [DLSM17] Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, 55(3):529–569, 2017.
- [DOS<sup>+</sup>10] Seniz Demir, David Oliver, Edward Schwartz, Stephanie Elzer, Sandra Carberry, and Kathleen F. McCoy. Interactive sight into information graphics. In

*Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, New York, NY, USA, 2010. ACM.

- [DPS04] N. Di Blas, P. Paolini, and M. Speroni. "usable accessibility" to the web for blind users. In *Proceedings of 8th ERCIM Workshop: User Interfaces for All*, Lecture Notes in Computer Science, Berlin, Heidelberg, 2004. Springer.
- [DVR07] Michel Dumontier and Natalia Villanueva-Rosales. Three-layer owl ontology design. In Bernardo Cuenca Grau, Vasant Honavar, Anne Schlicht, and Frank Wolter, editors, *Proceedings of the 2nd International Workshop on Modular Ontologies*, 2007.
- [DW07] Benjamin K. Davison and Bruce N. Walker. Sonification sandbox reconstruction: Software standard for auditory graphs. In *Proceedings of the 13th International Conference on Auditory Display*, pages 509–512. 2007.
- [Edd14] Matthew Daniel Eddy. How to see a diagram: A visual anthropology of chemical affinity. *Osiris*, 29(1):178–196, 2014.
- [Ede09] Shimon Edelman. On what it means to see, and what we can do about it. *Object Categorization: Computer and Human Vision Perspectives*, pages 69–86, 2009.
- [Egg14] Complex images: Web accessibility tutorials - guidance on how to create websites that meet wcag, 2014.
- [EH55] C. W. Eriksen and H. W. Hake. Absolute judgments as a function of stimulus range and number of stimulus and response categories. *Journal of Experimental Psychology*, 49(5):323–332, 1955.
- [EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 96(34):226–231, 1996.
- [EM13] Howard E. Evans and Malcolm E. Miller. *Miller's anatomy of the dog*. Elsevier, St. Louis, Missouri, fourth edition edition, 2013.
- [Eng02] Jörg von Engelhardt. *The Language of graphics: A framework for the analysis of syntax and meaning in maps, charts and diagrams*, volume 2002-03 of *ILLC dissertation series*. Institute for Logic, Language and Computation, Universiteit van Amsterdam, Amsterdam, 2002.
- [Eri98] Yvonne Eriksson. *Tactile pictures: Pictorial representations for the blind, 1784-1940*, volume 4 of *Gothenburg studies in art and architecture*. Acta Universitatis Gothoburgensis, Göteborg, 1998.

- [ESC<sup>+</sup>07] S. Elzer, E. Schwartz, S. Carberry, D. Chester, S. Demir, and P. Wu. A browser extension for providing visually impaired users access to the content of bar charts on the web. In *Proceedings of the International Conference on Web Information Systems (WEBIST)*, volume 2, pages 59–66. Springer, 2007.
- [Eur17] European Blind Union. Facts, figures and definitions concerning blindness and sight loss, 2017.
- [FB94] Joerg Fricke and Helmut Baehring. Displaying laterally moving tactile information. In Wolfgang L. Zagler, Geoffrey Busby, and Roland R. Wagner, editors, *Computers for Handicapped Persons*, Lecture Notes in Computer Science, pages 461–468. Springer-Verlag and Springer e-books, Berlin, Heidelberg, 1994.
- [FdCF12] Simone Bacellar Leal Ferreira, Denis Silva da Silveira, Eliane Pinheiro Capra, and Ariane Oliveira Ferreira. Protocols for evaluation of site accessibility with the participation of blind users. *Procedia Computer Science*, 14:47–55, 2012.
- [Fer06] Kay Alicyn Ferrell. Evidence-based practices for students with visual disabilities. *Communication Disorders Quarterly*, 28(1):42–49, 2006.
- [Fer09] Leo Ferres. The igraph-lite book, 2009.
- [FKL<sup>+</sup>17] Yuan Fang, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. Object detection meets knowledge graphs. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1661–1667. 2017.
- [FL10] Yansong Feng and Mirella Lapata. How many words is a picture worth? automatic caption generation for news images. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1239–1249. 2010.
- [FLST13] Leo Ferres, Gitte Lindgaard, Livia Sumegi, and Bruce Tsuji. Evaluating a tool for improving accessibility to charts and graphs. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 20(5):28, 2013.
- [FPRL06] Leo Ferres, Avi Parush, Shelley Roberts, and Gitte Lindgaard. Helping people with visual impairments gain access to graphical information through natural language: The igraph system. In Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler, Arthur I. Karshmer, Leo Ferres, Avi Parush, Shelley Roberts, and Gitte Lindgaard, editors, *Computers Helping People with Special Needs*, pages 1122–1130. Springer Berlin Heidelberg, 2006.
- [FR03] K. M. Franklin and J. C. Roberts. Pie chart sonification. In *Proceedings of the 7th International Conference on Information Visualization*, volume 4, pages 4–9. IEEE Comput. Soc, 2003.

- [FS09] Jinjuan Feng and Andrew Sears. Speech input to support universal access. In Constantine Stephanidis, editor, *The universal access handbook*, Human factors and ergonomics. CRC Press, Boca Raton, 2009.
- [FSK<sup>+</sup>11] Óscar Ferrández, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrández, Matteo Negri, Rubén Izquierdo, David Tomás, Constantin Orasan, Guenter Neumann, Bernardo Magnini, and Jose Luis Vicedo. The qall-me framework: A specifiable-domain multilingual question answering architecture. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):137–145, 2011.
- [FWB96] Jason P. Fritz, Thomas P. Way, and Kenneth E. Barner. Haptic representation of scientific data for visually impaired or blind persons. In *Proceedings of the CSUN Conference on Technology and Disability*. 1996.
- [GAMP87] Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C.N. Pereira. Team: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence*, 32(2):173–243, 1987.
- [Gao13] Jinglun Gao. *Information graphic classification, decomposition and alternative representation*. Phd dissertation, University of Delaware, 2013.
- [Gar79] K. Garland. Some general characteristics present in diagrams denoting activity, event and relationship. In *Information Design Journal*, volume 1(1), pages 15–22. 1979.
- [Gar14] John Gardner. Can mainstream graphics be accessible? *Information Technology and Disabilities E-Journal*, 14(1), 2014.
- [Gav89] William W. Gaver. The sonicfinder: an interface that uses auditory icons. *Human-Computer Interaction*, 4(1):67–94, 1989.
- [GB16] V. Gezer and S. Bergweiler. Service and workflow engineering based on semantic web technologies. In *Proceedings of the 10th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, volume 10, pages 152–157. IARIA XPS Press, 2016.
- [GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [GDK94] E. Goldberg, N. Driedger, and R. I. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53, 1994.
- [GGOEO<sup>+</sup>18] A. Garcia-Garcia, S. Orts-Escalano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez. A survey on deep learning techniques for

- image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.
- [Gil06] John M. Gill. Pharmaceutical packaging and labelling for use by blind and partially sighted people. *Pharmaceutical Technology Europe*, 18(11):34–36, 2006.
  - [GLS96] John A. Gardner, Randy Lundquist, and Steve Sahyun. Triangle: A practical application of non-speech audio for imparting information. In *Proceedings of the International Conference on Auditory Display*, pages 59–60. 1996.
  - [GLS<sup>+</sup>18] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people, 2018.
  - [GM11] Cagatay Goncu and Kim Marriott. Gravvitas: Generic multi-touch presentation of accessible graphics. In Pedro Campos, editor, *Human-Computer Interaction – INTERACT 2011 Proceedings of the 13th IFIP TC 13 International Conference*, Lecture Notes in Computer Science, pages 30–48. Springer, Heidelberg, 2011.
  - [GMA10] Cagatay Goncu, Kim Marriott, and Frances Aldrich. Tactile diagrams: Worth ten thousand words? In Ashok K. Goel, Mateja Jamnik, and N. Hari Narayanan, editors, *Diagrammatic representation and inference*, Lecture Notes in Computer Science, pages 257–263. Springer Heidelberg, Berlin, 2010.
  - [GO14] Thomas R. Gruber and Gregory R. Olsen. An ontology for engineering mathematics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, page 258, 2014.
  - [GOF08] B. Gould, T. O'Connell, and G. Freed. Effective practices for description of science content within digital talking books, 2008.
  - [Gol78] E. Goldsmith. *An analysis of the elements affecting comprehensibility of illustrations intended as supportive to text*. Phd thesis, Brighton Polytechnic, Brighton, 1978.
  - [Gol12] Owl 2 web ontology language new features and rationale (second edition): W3c recommendation 11 december 2012, 2012.
  - [GRM<sup>+</sup>19] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness: arxiv preprint arxiv:1811.12231, 2019.
  - [GRW97] A. L. Gorin, G. Riccardi, and J. H. Wright. How may i help you? *Speech communication*, 23(1), 1997.

- [GSG<sup>+</sup>09] Dimitris Grammenos, Anthony Savidis, Yannis Georgalis, Themistoklis Bourdenas, and Constantine Stephanidis. Electronic educational books for blind students. In Constantine Stephanidis, editor, *The universal access handbook*, Human factors and ergonomics. CRC Press, Boca Raton, 2009.
- [GWJ65] C. Gilson, B. Wurzburger, and D. E. Johnson. The use of the raised map in teaching mobility to blind children. *New Outlook for the Blind*, 59(2):59–62, 1965.
- [Hak12] Markku T. Hakkinen. Smart images, 2012.
- [Hal99] Jim Halliday. Braille vs. speech: Making sense of the debate. *Closing the Gap*, 17(6):6–7, 1999.
- [Har06] Sandra G. Hart. Nasa-task load index (nasa-tlx); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.
- [Hat93] Yvette Hatwell. Images and non-visual spatial representations in the blind. In D. Burger and J. C. Sperandio, editors, *Non-Visual Human-Computer Interactions*, Colloque Inserm, pages 13–35. INSERM, 1993.
- [HB11] Tom Health and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*, volume 1 of *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan & Claypool, 1 edition, 2011.
- [HD05] A. Harth and S. Decker. Optimized index structures for querying rdf from the web. In *Proceedings of the Third Latin American Web Congress. LA-WEB 2005.*, pages 71–80. IEEE, 2005.
- [Heg92] Mary Hegarty. Mental animation: Inferring motion from static displays of mechanical systems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(5):1084–1102, 1992.
- [Hen05] Essential components of web accessibility., 2005.
- [Hir06] Hiroshi Hirai. Characterization of the distance between subtrees of a tree by the associated tight span. *Annals of Combinatorics*, 10(1):111–128, 2006.
- [Hit12] Owl 2 web ontology language primer (second edition): W3c recommendation 11 december 2012, 2012.
- [HKP95] R. D. Howe, D. A. Kontarinis, and W. J. Peine. Shape memory alloy actuator controller design for tactile displays. In *Proceedings of the 34th IEEE conference on decision and control*, pages 3540–3544. IEEE, 1995.
- [HKRE14] S. L. Henry, H. Kramer, S. Rush, and B. Egan. Images concepts, 2014.

- [HMSD10] A. Holzinger, S. Mayr, W. Slany, and M. Debevc. The influence of ajax on web usability. In Boris Shishkov, David Marca, and Marten J. van Sinderen, editors, *Proceedings of the International Conference on e-Business*, pages 1–4, Portugal, 2010. SciTePress.
- [Hol93] Nigel Holmes. *The Best in Diagrammatic Graphics*. Rotovision, 1993.
- [Hor98] Robert E. Horn. *Visual language: Global communication for the 21st century*. MacroVU, Inc., Bainbridge Island, WA, 1998.
- [HR13] Vicki L. Hanson and John T. Richards. Progress on website accessibility? *ACM Transactions on the Web (TWEB)*, 7(1):Article No. 2, 2013.
- [HSR<sup>+</sup>13] Ulrike Hochpöchler, Wolfgang Schnotz, Thorsten Rasch, Mark Ullrich, Holger Horz, Nele McElvany, and Jürgen Baumert. Dynamics of mental model construction from text and graphics. *European Journal of Psychology of Education*, 28(4):1105–1126, 2013.
- [HSSW06] Richard C. Holt, Andy Schürr, Susan Elliott Sim, and Andreas Winter. Gxl: A graph-based standard exchange format for reengineering. *Science of Computer Programming*, 60(2):149–170, 2006.
- [Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
- [HWB06] W. Heuten, D. Wichmann, and S. Boll. Interactive 3d sonification for the exploration of city maps. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, pages 155–164, New York, NY, USA, 2006. ACM.
- [HWGS<sup>+</sup>06] C. Halaschek-Wiener, J. Golbeck, A. Schain, M. Grove, B. Parsia, and J. Hendler. Photostuff: an image annotation tool for the semantic web. In *Proceedings of the 4th International Semantic Web Conference*, pages 6–10, 2006.
- [HZLL19] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman, editors, *Proceedings of the 12th ACM International Conference*, pages 105–113. Association for Computing Machinery, 2019.
- [IGSM14] D. K. Iakovidis, T. Goudas, C. Smailis, and I. Maglogiannis. Ratsnake: A versatile image annotation tool with application to computer-aided diagnosis. *The Scientific World Journal*, 2014, 2014.
- [Int] International Council of Ophthalmology. Aspects and ranges of vision loss with emphasis on population surveys.

- [Int13] Internet.org. Whitepaper: a focus on efficiency, 2013.
- [Int16] International Telecommunications Union. The world in 2016: Ict facts and figures, 2016.
- [Int18] International Organization for Standardization. Ergonomics of human-system interaction: Part 11: Usability: Definitions and concepts, 2018.
- [JBGG09] Sergio Jimenez, Claudia Becerra, Alexander Gelbukh, and Fabio Gonzalez. Generalized mongue-elkan method for approximate text string comparison. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 559–570, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [Jen16] Sheri Wells Jensen. Just saying no to reading braille. In *The Braille Monitor*, volume 59. 2016.
- [JKS<sup>+</sup>17] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6706–6717, New York, NY, USA, 2017. ACM.
- [JLLC07] Chul-Jin Jang, Ji-Yeon Lee, Jeong-Won Lee, and Hwan-Gue Cho. Smart management system for digital photographs using temporal and spatial features with exif metadata. In *2nd International Conference on Digital Information Management, 2007*, pages 110–115, Piscataway, NJ, 2007. IEEE.
- [JM18] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd ed. draft edition, 2018.
- [Jok09] Kristiina Jokinen. Natural language and dialogue interfaces. In Constantine Stephanidis, editor, *The universal access handbook*, Human factors and ergonomics. CRC Press, Boca Raton, 2009.
- [Jon00] David H. Jonassen. Toward a design theory of problem solving. *Educational Technology Research and Development*, 48(4):63–85, 2000.
- [JPSM16] Prajaks Jitngernmadan, Andrea Petz, Bernhard Stöger, and Klaus Miesenberger. Analysis of implicit didactics in math schoolbooks for interactive non-visual user interface development. In Klaus Miesenberger, C. Bühlert, and Petr Penaz, editors, *Computers helping people with special needs*, volume 1, pages 19–26. Springer International Publishing, Cham, 2016.
- [Kas19] Alboukadel Kassambara. Dbscan: density-based clustering for discovering clusters in large datasets with noise - unsupervised machine learning, 2019.
- [Kea14] Kyle Keane. Interactive scientific graphics: Recommended practices for verbal description, 2014.

- [Ker03] George Kerscher. *Daisy is*, 2003.
- [KGBL14] Roberta L. Klatzky, Nicholas A. Giudice, Christopher R. Bennett, and Jack M. Loomis. Touch-screen technology for the dynamic display of 2d spatial information without vision: Promise and progress. *Multisensory Research*, 27(5-6):359–378, 2014.
- [KHS85] K. Kotovsky, J.R Hayes, and H.A Simon. Why are some problems hard? evidence from tower of hanoi. *Cognitive Psychology*, 17(2):248–294, 1985.
- [Kir09] Dmitry Kirsanov. *The book of Inkscape*. No Starch Press, San Francisco, 1 edition, 2009.
- [Kir18] Web content accessibility guidelines (wcag) 2.1: W3c recommendation, 2018.
- [KJ18a] Halina Kwaśnicka and L. C. Jain, editors. *Bridging the semantic gap in image and video analysis*, volume 145 of *Intelligent systems reference library*. Springer, Cham, Switzerland, 2018.
- [KJ18b] Halina Kwaśnicka and L. C. Jain. Semantic gap in image and video analysis: An introduction. In Halina Kwaśnicka and L. C. Jain, editors, *Bridging the semantic gap in image and video analysis*, Intelligent systems reference library, pages 1–6. Springer, Cham, Switzerland, 2018.
- [KL02] Hesham M. Kamel and James A. Landay. Sketching images eyes-free: a grid-based dynamic drawing tool for the blind. In *Assets '02 Proceedings of the fifth international ACM conference on Assistive technologies*, pages 33–40. ACM, 2002.
- [KMA<sup>+</sup>18] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Akos Kadar, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning, 2018.
- [KMS<sup>+</sup>06] Kristin Koch, Judith McLean, Ronen Segev, Michael A. Freed, Michael J. Berry, Vijay Balasubramanian, and Peter Sterling. How much the eye tells the brain. *Current biology : CB*, 16(14):1428–1434, 2006.
- [KO08a] Ivan Kopeček and Radek Ošlejšek. Gate to accessibility of computer graphics. In Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler, and Arthur Karshmer, editors, *Computers Helping People with Special Needs*, pages 295–302. Springer, Berlin, Heidelberg, 2008.
- [KO08b] Ivan Kopecek and Radek Oslejsek. Hybrid approach to sonification of color images. In *ICCIIT 2008*, pages 722–727, Los Alamitos, Calif., 2008. IEEE Computer Society.

- [KO11] Ivan Kopeček and Radek Ošlejšek. Communicative images. In Lutz Dickmann, editor, *Smart graphics*, volume 6815 of *LNCS sublibrary. SL 6, Image processing, computer vision, pattern recognition and graphics*, pages 163–173. Springer, Heidelberg, 2011.
- [Kok87] Kenneth Kokjer. The information capacity of the human fingertip. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(1):100–102, 1987.
- [Kos89] Stephen M. Kosslyn. Understanding charts and graphs. *Applied Cognitive Psychology*, 3(3):185–225, 1989.
- [Kos94] Stephen Michael Kosslyn. *Elements of graph design*. W.H. Freeman and Company, New York, 1994.
- [KPB<sup>+</sup>09] Ivan Kopeček, Jaromír Plhák, Bartek. L., Radek Ošlejšek, and F. Tirsel. Detection and annotation of graphical objects in raster images within the gate project. In H. R. Arabnia and Victor A. Clincy, editors, *Proceedings of the 2009 International Conference on Internet Computing*, pages 285–290. CSREA Press, Las Vegas Nevada, USA, 2009.
- [KPO<sup>+</sup>13] Girish Kulkarni, Visruth Premraj, Vicente Ordóñez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. Babytalk: understanding and generating simple image descriptions. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2891–2903, 2013.
- [KR01] I. Kocur and S. Resnikoff. Visual impairment and blindness in europe and their prevention. *The British journal of ophthalmology*, 86(7):716–722, 2001.
- [KS08] A. F. Kay and M. A. Siller. Dcmp description key, 2008.
- [KS11] Stacy M. Kelly and Derrick W. Smith. The impact of assistive technology on the educational performance of students with visual impairments: A synthesis of the research. *Journal of Visual Impairment & Blindness*, 105(2):73–83, 2011.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [KT06] Shuaib Karim and A. Min Tjoa. Towards the use of ontologies for improving user interaction for people with special needs. In Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler, Arthur I. Karshmer, Leo Ferres, Avi Parush, Shelley Roberts, and Gitte Lindgaard, editors, *Computers Helping People with Special Needs*, pages 77–84. Springer Berlin Heidelberg, 2006.
- [KTM09] Nikolaos Kaklanis, Dimitrios Tzovaras, and Konstantinos Moustakas. Haptic navigation in the world wide web. In Constantine Stephanidis, editor, *Proceedings of the 5th international conference on Universal access in human-computer*

- interaction. Applications and services*, volume 3 of *Lecture Notes in Computer Science*, pages 707–715. Springer, Berlin, 2009.
- [Kuk83] Karen Kukich. Design of a knowledge-based report generator. In *ACL '83 Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150, Stroudsburg, PA, USA, 1983. Association for Computational Linguistics.
  - [Kur95] Martin Kurze. Giving blind people access to graphics (example: Business graphics). In *Proceedings of the Software-Ergonomie'95 Workshop on Nicht-visuelle graphische Benutzungsoberflächen (Non-visual Graphical User Interfaces)*, Darmstadt, Germany, 1995.
  - [Kur96] Martin Kurze. Tdraw: a computer-based tactile drawing tool for blind people. In Ephraim P. Glinert and David L. Jaffe, editors, *ASSETS '96*, pages 131–138, New York, NY, USA, 1996. ACM.
  - [KWB<sup>+</sup>] Gregory Kramer, Bruce Walker, Terri Bonebright, Perry Cook, John Flowers, Nadine Miner, and John Neuhoff. Sonification report: Status of the field and research agenda: Technical report.
  - [KYM07] Ravi Kuber, Wai Yu, and Graham McAllister. Towards developing assistive haptic feedback for visually impaired internet users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1525–1534, New York, NY, USA, 2007. ACM.
  - [KZG<sup>+17</sup>] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, and David A. Shamma. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.
  - [LAC87] Joel Levin, Gary Anglin, and Russel Carney. On empirically validating functions of pictures in prose. In D.M Willows and H. A. Houghton, editors, *The Psychology of Illustration: Basic Research*, volume 1, pages 51–85. Springer, New York, NY, USA, 1987.
  - [LAKM07] Jonathan Lazar, Aaron Allen, Jason Kleinman, and Chris Malarkey. What frustrates screen reader users on the web: A study of 100 blind users. *International Journal of Human-Computer Interaction*, 22(3):247–269, 2007.
  - [Las99] Resource description framework (rdf) model and syntax specification, 1999.
  - [LB66] J. G. Linvill and J. C. Bliss. A direct translation reading aid for the blind. *Proceedings of the IEEE*, 54(1):40–51, 1966.

- [LB08] Mag Barbara Leitner and Erika Baldaszti. *Menschen mit Beeinträchtigungen Ergebnisse der Mikrozensus-Zusatzfragen 4. Quartal 2007*. Statistik Austria, Bericht im Auftrag des Bundesministeriums für Soziales und Konsumentenschutz, 2008.
- [LBWR94] Gerald L. Lohse, Kevin Biolsi, Neff Walker, and Henry H. Rueter. A classification of visual representations. *Communications of the ACM*, 37(12):36–49, 1994.
- [LFLA17] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In Rick Barrett, editor, *Proceedings of the 26th International Conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee, 2017.
- [LFMS12] Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265, 2012.
- [LGS<sup>+</sup>08] Nuria Lloret Romero, Vicent V.G.C. Gimenez Chornet, Jorge Serrano Cobos, Alicia A.S.C Selles Carot, F. Canet Centellas, and M. Cabrera Mendez. Recovery of descriptive information in images from digital libraries by means of exif metadata. *Library Hi Tech*, 26(2):302–315, 2008.
- [Liu10] Yuee Liu. *Ontology-based image annotation*. Phd thesis, Queensland University of Technology, Queensland, 2010.
- [LL07] L. Yujian and L. Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.
- [LMB<sup>+</sup>] David Langdon, George McKittrick, David Beede, Beethika Khan, and Mark Doms. Stem: Good jobs now and for the future.
- [LMS19] R. Lerch, C. Marquardt, and P. Storz. Extension subsystem, 2019.
- [LMV<sup>+</sup>08] Gordon E. Legge, Cindee Madison, Brenna N. Vaughn, Allen M. Y. Cheong, and Joseph C. Miller. Retention of high tactile acuity throughout the life span in blindness. *Perception & Psychophysics*, 70(8):1471–1488, 2008.
- [Loh93] Gerald Lee Lohse. A cognitive model for understanding graphical perception. *Human-Computer Interaction*, 8(4):353–388, 1993.
- [LOW<sup>+</sup>19] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey, 2019.
- [LPHL05] Vincent Lévesque, Jérôme Pasquier, Vincent Hayward, and Maryse Legault. Display of virtual braille dots by lateral skin deformation: feasibility study. *ACM Transactions on Applied Perception (TAP)*, 2(2):132–149, 2005.

- [LPP<sup>+</sup>06] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. ACM, 2006.
- [LPSZ08] Dennis Leas, Emilia Persoon, Neil Soiffer, and Michael Zacherle. Daisy 3: A standard for accessible multimedia books. *IEEE Multimedia*, 15(4):28–37, 2008.
- [LS87] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987.
- [LS89] Stephan Lewandowsky and Ian Spence. The perception of statistical graphs. *Sociological Methods & Research*, 18(2-3):200–242, 1989.
- [Luh58] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.
- [LW03] Steven Landua and Lesley Wells. Merging tactile sensory input and audio data by means of the talking tactile tablet. In *Proceedings of Eurohaptics '03*, pages 414–418. 2003.
- [LZLM07] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.
- [Mac95] Alan M. MacEachren. *How maps work: Representation, visualization, and design*. Guilford Press, New York and London, 1995.
- [Max10] J. C. Maxwell. Diagram. In *Encyclopedia Britannica*, volume 8, pages 146–149. Cambridge University Press, Cambridge, 1910.
- [MB03] David Milward and Martin Beveridge. Ontology-based dialogue systems. In *Proceedings of the 3rd Workshop on Knowledge and reasoning in practical dialogue systems (IJCAI03)*, pages 9–18. Citeseer, 2003.
- [MB06] David K. McGookin and Stephen A. Brewster. Contextual audio in haptic graph browsing. In *Proceedings of the 12th International Conference on Auditory Display*, pages 91–94, 2006.
- [MBJ85] Douglass L. Mansur, Merra M. Blattner, and Kenneth I. Joy. Sound graphs: A numerical data analysis method for the blind. *Journal of Medical Systems*, 9(3):163–174, 1985.
- [MBKS08] Oussama Metatla, Nick Bryan-Kinns, and Tony Stockman. Constructing relational diagrams in audio: the multiple perspective hierarchical approach. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 97–104, New York, NY, USA, 2008. ACM.

- [MF14] M. Malinowski and M. Fritz. Towards a visual turing challenge. In *Learning Semantics (NIPS workshop)*. 2014.
- [MG90] Richard E. Mayer and Joan K. Gallini. When is an illustration worth ten thousand words? *Journal of educational psychology*, 82(4):715, 1990.
- [MGH<sup>+</sup>14] Austin M. Mulloy, Cindy Gevarter, Megan Hopkins, Kevin S. Sutherland, and Sathiyaprakash T. Ramdoss. Assistive technology for students with visual impairments and blindness. In Giulio E. Lancioni and Nirbhay N. Singh, editors, *Assistive technologies for people with diverse abilities*, Autism and child psychopathology series, pages 113–156. Springer, New York, 2014.
- [MHP<sup>+</sup>00] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. The structure and performance of an open-domain question answering system. In Hitoshi Iida, editor, *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics - ACL '00*, pages 563–570, Morristown, NJ, USA, 2000. Association for Computational Linguistics.
- [Mil56] G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. In *Psychological Review*, volume 2, pages 81–97. 1956.
- [Mil87] Susanna Millar. Perceptual and task factors in fluent braille. *Perception*, 16(4):521–536, 1987.
- [Mil05] Skos core vocabulary specification: W3c working draft 2 november 2005, 2005.
- [MMM15] Tomás Murillo-Morales and Klaus Miesenberger. Ontology-based semantic support to improve accessibility of graphics. In C. Sik-Lányi, E.-J. Hoogerwerf, and K. Miesenberger, editors, *Studies in health technology and informatics*, volume 217, pages 255–260. IOS Press, 2015.
- [MMM17] Tomás Murillo-Morales and Klaus Miesenberger. Non-visually performing analytical tasks on statistical charts. In P. Cudd and L. de Witte, editors, *Studies in health technology and informatics*, volume 242, pages 339–346. IOS Press, 2017.
- [MMM18] Tomás Murillo-Morales and Klaus Miesenberger. Techniques for improved speech-based access to diagrammatic representations. In Klaus Miesenberger and Georgios Kouroupetroglou, editors, *Computers Helping People with Special Needs. 16th International Conference*, Lecture Notes in Computer Science, pages 636–643. Springer, 2018.
- [MMM20] Tomás Murillo-Morales and Klaus Miesenberger. Audial: A natural language interface to make statistical charts accessible to blind persons. In Klaus Miesenberger, Roberto Manduchi, Mario Covarrubias Rodriguez, and Petr

Peňáz, editors, *Computers Helping People with Special Needs*, Lecture Notes in Computer Science, pages 373–384. Springer, Cham, 2020.

- [MMM21] Tomás Murillo-Morales and Klaus Miesenberger. Formalizing visualization semantics for accessibility. In Katsuhito Yamaguchi, editor, *Proceedings of the 4th International Workshop on Digitization and E-Inclusion in Mathematics and Science 2021*, pages 47–56, Tokyo, Japan, 2021. Nihon University.
- [MMMR16] Tomás Murillo-Morales, Klaus Miesenberger, and Reinhard Ruemer. A latex to braille conversion tool for creating accessible schoolbooks in austria. In Klaus Miesenberger, C. Bühler, and Petr Penaz, editors, *Computers helping people with special needs*, Lecture Notes in Computer Science, pages 397–400. Springer, 2016.
- [MMPM18] Tomás Murillo-Morales, Jaromír Plhák, and Klaus Miesenberger. Authoring semantic annotations for non-visual access to graphics. *The Journal on Technology and Persons with Disabilities*, 6:398–413, 2018.
- [Moo00] Nick Moore. The information needs of visually impaired people. *A Review of Research for the RNIB*, 2000.
- [Mor38] C. Morris. Foundations of the theory of signs. In *International Encyclopedia of Unified Science*, volume 1. University of Chicago Press, Chicago, London, 1938.
- [Mor12] Nick Morgan. What is the future of publishing?, 2012.
- [MRR11] Peter Murray-Rust and Henry S Rzepa. Cml: Evolution and design. *Journal of cheminformatics*, 3(1):1–15, 2011.
- [MS01] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
- [MSB<sup>+</sup>14] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. 2014.
- [Mur08] Iain Murray. *Instructional eLearning technologies for the vision impaired*. Phd thesis, Curtin University, 2008.
- [Mus15] Mark A. Musen. The protégé project: A look back and a look forward. *AI matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence*, 1(4):4–12, 2015.
- [Myn95] Elizabeth D. Mynatt. Transforming graphical interfaces into auditory interfaces. In *Conference Companion on Human Factors in Computing Systems*, CHI '95, pages 67–68, New York, NY, USA, 1995. ACM.

- [Nah04] Fiona Fui-Hoon Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.
- [Nat] National Federation of the Blind. The braille literacy crisis in america: Facing the truth, reversing the trend, empowering the blind.
- [New93] William Newman. A preliminary analysis of the products of hci research, using pro forma abstracts. In Beth Adelson, editor, *Human factors in computing systems*, pages 278–284, Reading, Mass., 1993. Addison-Wesley.
- [NLQ<sup>+</sup>04] I. Nasidze, E. Y. S. Ling, D. Quinque, I. Dupanloup, R. Cordaux, S. Rychkov, O. Naumova, O. Zhukova, N. Sarraf-Zadegan, G. A. Naderi, S. Asgary, S. Sardas, D. D. Farhud, T. Sarkisian, C. Asadov, A. Kerimov, and M. Stoneking. Mitochondrial dna and y-chromosome variation in the caucasus. *Annals of human genetics*, 68(Pt 3):205–221, 2004.
- [Noy17] N. Noy. Representing classes as property values on the semantic web: W3c working group note 5 april 2005, 2017.
- [NR07] Tooba Nasir and Jonathan C. Roberts. Sonification of spatial data. In William L. Martens, Rene Quesnel, and Gary Scavone, editors, *Proceedings of the 13th International Conference on Auditory Display*, pages 112–119, 2007.
- [OB97] W. C. Ogden and P. Bernick. Using natural language interfaces. In M. G. Helander, P. V. Prabhu, and T. K. Landauer, editors, *Handbook of human-computer interaction*, pages 137–161. Elsevier, 1997.
- [OCC<sup>+</sup>19] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In Kohei Arai and Supriya Kapoor, editors, *Advances in computer vision*, volume 943 of *Advances in Intelligent Systems and Computing*, pages 128–144. Springer, Cham, Switzerland, 2019.
- [OGGL15] Sile O'Modhrain, Nicholas A. Giudice, John A. Gardner, and Gordon E. Legge. Designing media for visually-impaired users of refreshable touch displays: Possibilities and pitfalls. *IEEE transactions on haptics*, 8(3):248–257, 2015.
- [OKB11] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 1143–1151, 2011.
- [OMBC06] William Ogden, James McDonald, Philip Bernick, and Roger Chadwick. Habitability in question-answering systems. In Tomek Strzalkowski and Sanda M. Harabagiu, editors, *Advances in open domain question answering*, Text, speech, and language technology, pages 457–473. Springer, Dordrecht, 2006.

- [OR18] K. M. Odell and R. C. Russell. Soundex phonetic comparison system. *US Patent*, 1261167, 1918.
- [Ošl09] Radek Ošlejšek. Annotation of pictures by means of graphical ontologies. In *Proceedings of the 2009 International Conference on Internet Computing ICOMP 2009*, pages 296–300, USA, 2009. CSREA Press.
- [Ost02] S. A. Osterhaus. Susan's math technology corner: The accessible graphing calculator (agc) from viewplus software. *Division on Visual Impairments Quarterly*, 47(2):55–58, 2002.
- [Pal77] Stephen E. Palmer. Hierarchical structure in perceptual representation. *Cognitive Psychology*, 9(4):441–474, 1977.
- [Pal13] Hari Prasath Palani. *Making Graphical Information Accessible Without Vision Using Touch-based Device*. Ms thesis, University of Maine, Orono, Maine, US, 2013.
- [Par88] Don Parkes. "nomad": An audio-tactile tool for the acquisition, use and management of spatially distributed information by partially sighted and blind persons. In *Proceedings of the Second International Conference on Maps and Graphics for Visually Disabled People*, pages 24–29. 1988.
- [Pat04] Rdf-compatible model-theoretic semantics for owl, 2004.
- [PBB<sup>+</sup>08] S. C. Peres, V. Best, D. Brock, B. Shinn-Cunningham, C. Frauenberger, T. Hermann, J. G. Neuhoff, L. V. Nickerson, and T. Stockman. Auditory interfaces. In Philip Kortum, editor, *HCI Beyond the GUI*, pages 147–195. Morgan Kaufmann, 2008.
- [PE03] Ian Pitt and Alistair Edwards. *Design of Speech-based Devices: A Practical Guide*. Practitioner Series. Springer, London, 2003.
- [Pet12] W3c xml schema definition language (xsd) 1.1 part 2: Datatypes: W3c recommendation 5 april 2012, 2012.
- [Pin90] Steven Pinker. A theory of graph comprehension. *Artificial intelligence and the future of testing*, pages 73–126, 1990.
- [Plh14] Jaromír Plhák. *Dialogue-based Exploration of Graphics for Users with a Visual Disability*. Doctoral thesis, Masaryk University, Faculty of Informatics, Brno, 1/1/2014.
- [PM12] D. Pascolini and S. P. Mariotti. Global estimates of visual impairment: 2010. *The British journal of ophthalmology*, 96(5):614–618, 2012.
- [PM15] Andrea Petz and Klaus Miesenberger. Supporting blind students in stem education in austria. *Studies in Health Technology and Informatics*, 217:27–31, 2015.

- [PRG<sup>+</sup>09] François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7-8):789–816, 2009.
- [PSB<sup>+</sup>02] Helen Petrie, C. Schlieder, P. Blenkhorn, G. Evans, A. King, Anne-Marie O'Neill, G. T. Ioannidis, B. Gallagher, D. Crombie, R. Mager, and M. Alafaci. Tedub: a system for presenting and exploring technical drawings for blind people. In Klaus Miesenberger, Joachim Klaus, and Wolfgang Zagler, editors, *Computers Helping People with Special Needs : 8th International Conference, ICCHP 2002 Linz, Austria, July 15–20, 2002 Proceedings*, Lecture Notes in Computer Science, pages 537–539. Springer-Verlag Berlin Heidelberg, Berlin, 2002.
- [PV13] Jan Polowinski and Martin Voigt. Viso: a shared, formal knowledge base as a foundation for semi-automatic infovis systems. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems. Proceedings.*, pages 1791–1796, New York, NY, USA, 2013. ACM.
- [Ras00] Jef Raskin. *The humane interface: New directions for designing interactive systems*. Addison-Wesley, Reading, Mass. and Harlow, 2000.
- [Rec03] Alan L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In John Gennari, Bruce Porter, and Yolanda Gil, editors, *Proceedings of the second international conference on knowledge capture*, page 121, New York, New York, USA, 2003. ACM Press.
- [Red14] Patricia Redsicker. Social photos generate more engagement: New research, 2014.
- [Ric84] Clive James. Richards. *Diagrammatics : an investigation aimed at providing a theoretical framework for studying diagrams and for establishing a taxonomy of their fundamental modes of graphic organization: Ph.D Thesis*. PhD thesis, Royal College of Art, 1984.
- [Ric15] Authoring tool accessibility guidelines (atag) 2.0: W3c recommendation, 2015.
- [RKE05] Martin Rotard, Sven Knödler, and Thomas Ertl. A tactile web browser for the visually disabled. In *HYPERTEXT '05 Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 15–22, New York, NY, USA, 2005. ACM.
- [RKPP02] P. Roth, H. Kamel, L. S. Petrucci, and T. Pun. A comparison of three nonvisual methods for presenting scientific graphs. *Journal of Visual Impairment and Blindness*, 96(6):420–428, 2002.

- [ROE04] Martin Rotard, Kerstin Otte, and Thomas Ertl. Exploring scalable vector graphics for visually impaired users. In Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler, and Dominique Burger, editors, *Proceedings of the 9th International Conference on Computers Helping People with Special Needs*, Lecture Notes in Computer Science, pages 725–730. Springer Berlin Heidelberg, 2004.
- [RP02] Milan Randić and Dejan Plavšić. On the concept of molecular complexity. *Croatica Chemica Acta*, 75(1):107–116, 2002.
- [RPAP00] Patrick Roth, Lori Stefano Petrucci, André Assimacopoulos, and Thierry Pun. Audio-haptic internet browser and associated tools for blind users and visually impaired computer users. In C. Germain, O. Lavialle, and E. Grivel, editors, *COST 254 Intelligent Terminals, Workshop on Friendly Exchanging Through the Net*, pages 57–62. Bordeaux, France, 2000.
- [RRSB08] Daniel L. Rubin, Cesar Rodriguez, Priyanka Shah, and Chris Beaulieu. ipad: Semantic annotation and markup of radiological images. *AMIA Annual Symposium Proceedings*, 2008:626–630, 2008.
- [RTMF08] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.
- [Ryl96] R. Ryles. The impact of braille reading skills on employment, income, education, and reading habits. In *Journal of Visual Impairment and Blindness*, volume 90, pages 219–226. 1996.
- [RYRB01] R. Ramloll, W. Yu, B. Riedel, and S. A. Brewster. Using non-speech sounds to improve access to 2d tabular numerical information for visually impaired users. In *Proceedings of the 15th Annual Conference of the British HCI Group*, pages 515–529. Springer Verlag, 2001.
- [San10] Frode Eika Sandnes. Where was that photo taken? deriving geographical information from image collections based on temporal exposure attributes. *Multimedia Systems*, 16(4-5):309–318, 2010.
- [SBF98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data and knowledge engineering*, 25(1):161–198, 1998.
- [SCD<sup>+</sup>17] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In IEEE International Conference on Computer Vision, editor, *2017 IEEE International Conference on Computer Vision*, pages 618–626, Piscataway, NJ, 2017. IEEE.
- [Sch14] Rdf 1.1 primer: W3c working group note 24 june 2014, 2014.

- [Sch19] Owl 2 web ontology language: Rdf-based semantics (second edition), 01/04/2019.
- [SDK15] Dimitris Spiliopoulos, Athanasios Dalianis, and Dimitris Koryzis. Speech enabled ontology graph navigation and editing. In *International Conference on Universal Access in Human-Computer Interaction*, pages 487–494. Springer, Cham, 2015.
- [SDMRG03] C. Sjöström, H. Danielsson, C. Magnusson, and K. Rassmus-Gröhn. Phantom-based haptic line graphics for blind persons. *Visual Impairment Research*, 5(1):13–32, 2003.
- [SDP<sup>+</sup>19] M. Strehl, R. Della Martire, J. Patonnier, T. Hunt, and T. Brierley. Tools for svg, 2019.
- [SH10] Edward Segel and Jeffrey Heer. Narrative visualization: telling stories with data. *IEEE transactions on visualization and computer graphics*, 16(6):1139–1148, 2010.
- [Sha14] Kartik Shawhney. Stem access for the blind and visually impaired, 2014.
- [SHL<sup>+</sup>16] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. Figureseer: Parsing result-figures in research papers. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision*, Lecture Notes in Computer Science. Springer, Cham, 2016.
- [Shn96] B. Schneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE symposium on visual languages*, pages 336–343. IEEE Comput. Soc. Press, 1996.
- [SKC<sup>+</sup>11] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402, New York, NY, USA, 2011. ACM.
- [SKLW00] S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker. Reinforcement learning for spoken dialogue systems. *Advances in Neural Information Processing Systems*, pages 956–962, 2000.
- [SMLG03] N. Sribunruangrit, C. Marque, C. Lenay, and O. Gapenne. Improving blind people’s spatial ability by bimodal-perception assistive device for accessing graphic information. In *Proceedings of the 7th Biannual*, volume 3, pages 476–480. 2003.
- [SSM98] M. Shinohara, Y. Shimizu, and A. Mochizuki. Three-dimensional tactile display for the blind. *IEEE Transactions on Rehabilitation Engineering*, 6(3):249–256, 1998.

- [ST15] Jaka Sodnik and Sašo Tomažič. *Spatial auditory human-computer interfaces*. SpringerBriefs in computer science. Springer, Berlin, Heidelberg, 2015.
- [Ste02] Nico Stehr. *Knowledge and economic conduct: The social foundations of the modern economy / Nico Stehr*. Studies in comparative political economy and public policy. University of Toronto Press, Toronto and London, 2002.
- [STS<sup>+</sup>06] C. Saathoff, N. Timmermann, S. Staab, K. Petridis, D. Anastasopoulos, and Y. Kompatsiaris. M-ontomat-annotizer: Linking ontologies with multimedia low-level features for automatic image annotation. In *Demos and Posters of the 3rd European SemanticWeb Conference (ESWC)*. 2006.
- [SV02] J. Singer and N. G. Vinson. Ethical issues in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 28(12):1171–1180, 2002.
- [Tan06] Lk Tan. Image file formats. *Biomedical imaging and intervention journal*, 2(1):e6, 2006.
- [The12] The Braille Authority of North America. Guidelines and standards for tactile graphics: Web version, 2012.
- [TKMS03] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Marti Hearst and Mari Ostendorf, editors, *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, pages 173–180, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [TM04] M. Tory and T. Moller. Rethinking visualization: A high-level taxonomy. In Matthew Ward and Tamara Munzner, editors, *IEEE symposium on information visualization*, pages 151–158. IEEE, 2004.
- [TMS03] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: An overview. In Anne Abeillé, editor, *Treebanks*, volume 20 of *Text, Speech and Language Technology*, pages 5–22. Springer Netherlands, Dordrecht, 2003.
- [Tow04] Christopher Phillip Town. *Ontology based Visual Information Processing*. Doctoral dissertation, University of Cambridge, Cambridge, 2004.
- [Try16] Robert Trypuz. About schema.org initiative, 2016.
- [TS04] T. S. Tullis and J. N. Stetson. A comparison of questionnaires for assessing website usability. In *Usability professional association conference*. 2004.
- [TSR11] Ross Turner, Somayajulu Sripada, and Ehud Reiter. Generating approximate geographic descriptions. In Emiel Krahmer and Mariët Theune, editors, *Empirical Methods in Natural Language Generation*, Lecture Notes in Computer

- Science, pages 121–140. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [Tuf97] Edward R. Tufte. *Visual explanations: Images and quantities, evidence and narrative*. Graphics Press, Cheshire, Connecticut, 1997.
- [Tuf01] Edward R. Tufte. *The visual display of quantitative information*. Graphics Press LLC, Cheshire, Conn., USA, 2nd ed. edition, 2001.
- [Uni11] Universität Karlsruhe. Handbuch zur erstellung taktiler graphiken, 2011.
- [U.S] U.S. General Services Administration, Office of Governmentwide Policy. Section508.gov.
- [Usc98] Mike Uschold. Knowledge modelling: concepts and terminology. *The Knowledge Engineering Review*, 13(1):5–29, 1998.
- [VH13] Markel Vigo and Simon Harper. Coping tactics employed by visually disabled users on the web. *International Journal of Human-Computer Studies*, 71(11):1013–1025, 2013.
- [VP] Martin Voigt and Jan Polowinski. Towards a unifying visualization ontology: Technical report tud-fi11-01.
- [vSOH97] J.P.H. van Santen, R. Sproat, J. Olive, and J. Hirschberg, editors. *Progress in Speech Synthesis*. Springer New York, 1997.
- [VTBE15] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164. 2015.
- [W3C12] Owl 2 web ontology language document overview (second edition): W3c recommendation 11 december 2012, 2012.
- [Wat13] Léonie Watson. Using aria to enhance svg accessibility, 2013.
- [WB97a] T. P. Way and K. E. Barner. Automatic visual to tactile translation. i. human factors, access methods and image manipulation. *IEEE Transactions on Rehabilitation Engineering*, 5(1):81–94, 1997.
- [WB97b] T. P. Way and K. E. Barner. Automatic visual to tactile translation. ii. evaluation of the tactile image creation system. *IEEE Transactions on Rehabilitation Engineering*, 5(1):95–105, 1997.
- [WB06] Steven Wall and Stephen Brewster. Feeling what you hear: tactile feedback for navigation of audio graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1123–1132. ACM, New York, NY, USA, 2006.

- [WDH<sup>+</sup>01] R. H. Wiggins, H. C. Davidson, H. R. Harnsberger, J. R. Lauman, and P. A. Goede. Image file formats: past, present, and future. *Radiographics*, 21(3):789–798, 2001.
- [Web06] Frank Webster. *Theories of the information society*. International library of sociology. Routledge, London, 3rd ed. edition, 2006.
- [Web17] WebAIM. Webaim: Alternative text, 2017.
- [WGCP07] Laura Winfield, John Glassmire, J. Edward Colgate, and Michael Peshkin. T-pad: Tactile pattern display through variable friction reduction. In *World Haptics 2007*, pages 421–426, Los Alamitos Calif., 2007. IEEE Computer Society.
- [WGG11] Michael Wong, Vishi Gnanakumaran, and Daniel Goldreich. Tactile spatial acuity enhancement in blindness: evidence for experience-dependent mechanisms. *The Journal of Neuroscience*, 31(19):7028–7037, 2011.
- [WGK<sup>+</sup>12] Wagemans, J., Feldman, J., S. Gepshtein, R. Kimchi, J. R. Pomerantz, van der Helm, P. A., and C. van Leeuwen. A century of gestalt psychology in visual perception ii. conceptual and theoretical foundations. *Psychological bulletin*, 138(6):1218–1252, 2012.
- [Wic10] Hadley Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010.
- [Wie14] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 2014.
- [Wila] G. Willetts. Services for people with visual impairments in luton: Review report for the social services department.
- [Wilb] William E. Winkler. The state of record linkage and current research problems.
- [Wil05] Leland Wilkinson. *The grammar of graphics*. Statistics and computing. Springer, New York, 2nd ed. edition, 2005.
- [WK05] Bruce N. Walker and Gregory Kramer. Mappings and metaphors in auditory displays: An experimental assessment. *ACM Transactions on Applied Perception (TAP)*, 2(4):407–412, 2005.
- [WLH02] C. R. Wagner, S. J. Lederman, and R. D. Howe. A tactile shape display using rc servomotors. In *10th symposium on haptic interfaces for virtual environment and teleoperator systems*, pages 354–355. IEEE Comput. Soc, 2002.
- [WM91] Peter C. Wright and Andrew F. Monk. A cost-effective evaluation method for use by designers. *International Journal of Man-Machine Studies*, 35(6):891–912, 1991.

- [WM10] B. N. Walker and L. M. Mauney. Universal design of auditory graphs: A comparison of sonification mappings for visually impaired and sighted listeners. *ACM Transactions on Accessible Computing (TACCESS)*, 2(3):12, 2010.
- [WNL06] Bruce N. Walker, Amanda Nance, and Jeffrey Lindsay. Spearcons: speech-based earcons improve navigation performance in auditory menus. In *Proceedings of the International Conference on Auditory Display*, pages 63–68. 2006.
- [Wor04a] World Wide Web Consortium. Mapping to rdf graphs for owl: Owl web ontology language semantics and abstract syntax, 2004.
- [Wor04b] World Wide Web Consortium. Speech recognition grammar specification version 1.0, 2004.
- [Wor11a] World Wide Web Consortium. Coordinate systems, transformations and units – svg 1.1 (second edition), 2011.
- [Wor11b] World Wide Web Consortium. Html5. a vocabulary and associated apis for html and xhtml: W3c working draft 13 january 2011, 2011.
- [Wor12] World Wide Web Consortium. Accessible infographics community group, 2012.
- [Wor17] World Wide Web Consortium. W3c semantic web activity homepage, 22/08/2017.
- [Wor18] World Wide Web Consortium. Scalable vector graphics (svg) 2, 2018.
- [WSL<sup>+</sup>16] Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton van den Hengel. What value do explicit high level concepts have in vision to language problems? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–212, 2016.
- [Wu16] Shaomei Wu. How blind people interact with visual content on social networking sites, 2016.
- [XIP<sup>+</sup>11] Cheng Xu, Ali Israr, Ivan Poupyrev, Olivier Bau, and Chris Harrison. Tactile display for the visually impaired using teslatouch. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, pages 317–322, New York, NY, USA, 2011. ACM.
- [YKM<sup>+</sup>06] Wai Yu, Ravi Kuber, Emma Murphy, Philip Strain, and Graham McAllister. A novel multimodal interface for improving visually impaired people's web accessibility. *Virtual Reality*, 9(2-3):133–148, 2006.
- [YRB01] Wai Yu, Ramesh Ramloll, and Stephen Brewster. Haptic graphs for blind computer users. In Stephen Brewster and Roderick Murray-Smith, editors, *Haptic human-computer interaction*, volume 2058 of *Lecture Notes in Computer Science*, pages 41–51. Springer, Berlin and London, 2001.

- [YRB02] W. Yu, D. Reid, and S. Brewster. Web-based multimodal graphs for visually impaired people. In Simeon Keates, Patrick Langdon, P. John Clarkson, and Peter Robinson, editors, *Universal Access and Assistive Technology*, pages 97–108, London, 2002. Springer London.
- [ZMW15] Limin Zeng, Mei Miao, and Gerhard Weber. Interactive audio-haptic map explorer on a tactile display. *Interacting with Computers*, 27(4):413–429, 2015.