Localization Engineering:
Developing Two Applications Tailored for Translators

Aloísio João Spínola Ferreira

Trabalho de Projeto de Mestrado em Tradução

(Especialização em Inglês)

ACKNOWLEDGEMENTS

I would like to dedicate this section to express my heartfelt gratitude to several individuals who have been instrumental in the completion of this paper.

First and foremost, I extend my deepest appreciation to my family, whose unwavering support, encouragement, and sacrifices have been the bedrock of my journey. To my parents, whose belief in my abilities has been unwavering; to my grandparents, whose bewildered yet unwavering faith in my pursuit of a Master's degree has been a source of inspiration; and to my sister, whose unwavering support and inspiration have been a guiding light.

I am profoundly thankful to my partner, Tiago, for his steadfast companionship, understanding, and patience during the demanding phases of this endeavor. Your unwavering support and delicious cooking have been the pillars upon which I leaned in moments of uncertainty.

I owe an immense debt of gratitude to the exceptional team at Linguaemundi for their invaluable guidance, collaboration, and camaraderie. Working alongside such dedicated professionals was an enriching experience, and I am grateful for their contributions to making this project a reality.

Special thanks to Kevin Lossner, Marco Neves, David Hardisty, and every other professor who generously shared their expertise, provided guidance, and offered invaluable feedback throughout my academic and professional journey and general generosity. Your mentorship has been instrumental in shaping this paper and personal growth, and I am deeply and eternally grateful for your support.

I extend my sincere appreciation to everyone else who has supported me in various ways, whether through words of encouragement, constructive feedback, or acts of kindness. Your collective encouragement has been invaluable, and I am deeply grateful for your belief in me.

Finally, I pay tribute to those who are no longer with me here but whose belief in me remains an enduring legacy, I dedicate this accomplishment to their memory. I wish to continue making you all proud and living on behalf of those who were tragically taken too soon, especially those who had their future robbed from them.

Localization Engineering:
Hidden in the Gap Between Worlds

Aloísio João Spínola Ferreira

ABSTRACT

This project focuses on one of the many aspects of a localization engineer's role through the development of two different software applications aimed at professional translators: CAT Nav and CAT Pal. Both applications aid in the process of searching for the terminology, with the first one streamlining the browsing process, while the second serves to create and edit term bases. As a theoretical background for the development of both apps, this paper will also explore translation technology and its history through recounts of translators and other scholars such as Paillet, O'Hagan, Pym, and Somers. Through the dissection of CAT tools, namely memoQ, we will also be analyzing how these tools work, how external apps may enhance them, including the apps developed herein, and their impact on the translation industry and its continued development.

Contents

1. Introduction

This paper revolves around a project consisting of the development of two software applications aimed at professional translators and terminologists. It was done under the span of approximately eight months, from August 2023 to March 2024.

My fascination with CAT (Computer-Assisted Translation) tools and the expansive field of translation technology has been longstanding. I was keenly aware of the fact that my background in programming could give me an edge to pull off a distinct and innovative project few could undertake relating to this field, but finding the right idea proved challenging initially.

Since early 2023, I was lucky enough to land a job working as a project manager and localization engineer at Linguaemundi working alongside a team of professional translators within a translation company. Through months of observation, dialogue, and problem-solving alongside this team, I identified areas where technological solutions in the form of custom applications could address their challenges. It was then that I came to the realization that these applications held the potential to fulfill the vision I had been seeking for my Master's degree project.

The subsequent chapters of this paper unfold with a dual purpose. The second chapter delves into the academic exploration of translation technology, tracing its evolution and impact on the translation industry. Drawing upon the works of esteemed scholars such as O'Hagan, Somers, and Pym, this chapter contextualizes the role of localization engineers and their significance in bridging the gap between technology and human translation.

The third chapter focuses entirely on the first application named CAT Nav. Here, we explore its genesis, guiding principles, unique features, and design choices. This chapter will touch briefly on fundamental concepts surrounding internet browsing, particularly URLs and how they function by scholars such as Berners-Lee.

The fourth chapter will focus on the second application named CAT Pal. As the vastly more intricate application of the two, this chapter will also be significantly lengthier compared to the others. As this application delves entirely into term bases, This section offers a comprehensive analysis of the historical context of term bases, informed by firsthand accounts from professional translators such as Paillet. Furthermore, it

provides an in-depth exploration of term base management within CAT tools, with a specific focus on memoQ and its relevance to the functionality of CAT Pal.

The heart of this paper lies in the final two chapters, which encapsulate the essence of the project, documenting the challenges, decisions, and thought processes involved in the inception and development of the applications. Meanwhile, the initial chapters serve to establish an academic foundation, setting the stage for the practical insights and innovations presented in the subsequent chapters.

2. Translation and Technology

2.1. Translation Technology and CAT Tools

Translation is a practice as old as human communication itself. It is one of the invisible and uncredited pillars of society as it follows in its trail with every step humankind takes. It is thus unsurprising that the technology boom that marked the 20th century and the birth of the first computers bared its mark on the profession.

Translation technology serves to aid translators in maximizing output in quantity and quality while also minimizing effort and time taken. This technology has taken countless shapes and forms, be it machine translation (MT), term bases (TBs), or translation memories (TMs). The first instances of technology being used for translation date back to as far as the 1950s and 1970s in the form of machine translation and translation memories respectively (O'Hagan, 2013, p. 505)

Over the years, these applications, despite their differences, were unified in their one true goal to help translators and were coined under the same umbrella term: computer-assisted translation tools (CAT tools). These tools picked up steam once they started being commercialized and more readily available for the public in the mid to late 1990s (Somers, 2003, p. 31). It was around this time that CAT tools graduated from being a luxury to a bare necessity. This software has become so omnipresent in fact that some scholars even question the redundancy of the term in the modern day, such as Pym (2010), "since almost all translation is done with computers… so all processes are 'computer-aided' to some extent" (p. 123).

In its still relatively short lifespan, the term CAT tool has felt other subtle but still noticeable changes in its meaning that go beyond the notion of redundancy. As mentioned, CAT tools served as an umbrella term to encapsulate a plethora of different applications, such as term bases, machine translation, and translation memories. However, as newer tools such as SDL Trados and memoQ have made themselves the industry standard, there has been a noticeable shift. The term has plateaued into accepting these applications as the true CAT tools, whereas TBs and TMs have been delegated as mere features that make up the CAT tool itself. In short, modern CAT tools have become an agglomerate of smaller tools.

Furthermore, as some scholars have pointed out over the years, other tools that are not traditionally counted as CAT tools have also helped translators immensely, marking a significant shift in how they work, such as internet search engines and spell checkers (Pym A. &., 2006). With a term as broad as computer-assisted translation tools, one could argue that it is not a stretch to claim that these systems could be grouped in with the rest, at least when MT and TM on their own were considered independent CAT tools. This ultimately means that although term CAT tool has gained some shape, there is still much to be debated and gray areas to be addressed.

While the history of CAT tools and their impact on the translation industry have and continue to be studied on an academic level, especially since the start of the millennium (Vieira, 2021, p. 395), the same cannot be said for how these operate from a technical perspective. Translation technology exists in a peculiar limbo as it exists between two metaphorical tectonic plates, linguistics and IT. As a result, it presents some blind spots from both perspectives, as the average linguist lacks the technological knowledge to fully grasp how these tools work while garnering almost no interest from programmers to understand the translator's perspective.

2.2. What is Localization engineering?

The term "Localization Engineer" is a rather recent term, coined around the early 2000s but only gaining real traction in the last decade. Due to the term's novelty and how niche it is, there is still much to be defined within the role, as the lines are still very much drawn in the sand. Esselink (Localization Engineering: The Dream Job?, 2002, p. 1) summed it best when he claimed "It is not translation but still very related to language. It is not engineering but still much related to building products."

The role's ambiguity can even be discerned by how wildly expectations vary from employer to employer. While some require a background in IT, others in linguistics, both, or even none, it's not difficult to conclude that, much like translation technology, it's difficult to pinpoint exactly where this role stands. Should the role be given to a tech savvy linguist? Or perhaps a knowledgeable programmer? The scarcity of references to localization engineers and localization engineering in academic literature, at least by name, further illustrates this identity crisis.

Their main goal is to solve technical problems related to the tools used by translators, plan and optimize the translation workflow, and handle the importing,

exporting, and re-integration of the translated content into the final product (Esselink, 2002, p. 2). In other words, localization engineers are expected to leverage translation technology using their knowledge to ensure maximum efficiency.

The name localization engineer itself may also appear misleading. Despite being mostly popular in the localization industry, this role is not exclusive to this type of translation, as many of its tasks are very much transversal to all other areas of translation. Workflow inefficiencies, issues in handling files or CAT tools are a few examples of problems that every company and freelancer face no matter their expertise.

Although much can be discussed about this role, there is one more common truth that can be observed. Localization engineers serve to close the gap between these two fields. It is simultaneously a linguist who is well-versed in technology and a programmer who understands the translator's perspective. This gap can rival canyons in how different both worlds are, as documented historically by scholars and translators since the birth of CAT tools. Mayorcas (1988) in a secondhand account retells:

> She also had some disturbing experiences to recount the difficulties that small users may experience in their relationships with suppliers who, apparently, may lack the linguistic knowledge to explain the functioning of certain features of the system, sort out translation-related difficulties, and assist with customization. (p. 10)

It is also not uncommon for the technology available to be insufficient and dedicated programmers are out of the table. In these cases, some localization engineers may also be expected to find a solution in the form of custom software or by creating very specialized resources and settings. One may argue that in these cases, the engineers birth their own CAT tools, at least in the traditional sense of the word. It is a situation not too dissimilar to the one I just described that inspired the whole premise of this project.

2.3. How the applications were conceived

As a localization engineer in a team of translators in my day-to-day life, I am tasked with solving many issues and optimizing their maximum output, usually by following direct orders on what to do and acting reactively. However, it is also my duty to analyze issues and inefficiencies that may be invisible to translators and higher-ups as they consciously or subconsciously chalk it up as something outside of anyone's control.

As I identified underlying issues about their workflow, I quickly realized that no existing tool satisfied all the conditions necessary for the desired result and as such, I decided to create them myself on my own free time.

These applications were developed using C#, a programming language I had experience with, as Windows Presentation Foundation (WPF) apps in Visual Studio 2022. I knew from the very beginning that I wanted to have a Graphic User Interface (GUI) since it would appeal and be easier to use to the average user. This would mean that the apps were going to be exclusive to the Windows operating system, and thus not accessible to users using Mac or Linux. However, since most notable CAT tools were similarly made and thus also exclusively on Windows, this did not deter me too much.

These applications were created continuously for the course of approximately 5 months with each release being distributed in intervals of two weeks. Within these two weeks, I would collect user feedback and work on bug fixes and implementing features requested by the users.

3. The first app: CAT Nav

MemoQ Web Search is a feature exclusive to memoQ that allows users to set up a number of websites of their choosing to search for a given term directly from the translation grid (MemoQ2). MemoQ describes on its own website how to properly set it up in its settings:

> In the Search URL box, enter the web address of the search page (not the main page!) of this web page. When you search on a web page, it will use an address where the address itself contains the text you are searching for. In the Edit search provider window, use a pair of curly brackets {} where the search text will be. (MemoQ1)

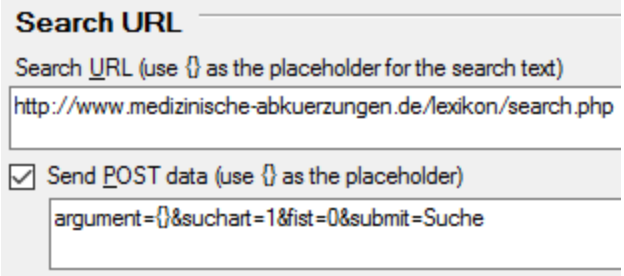

Figure 1. Search URL in memoQ Web Search's menu

In theory, this eliminates the need to open a browser and manually search for whatever you are looking for, thus saving an immense amount of time. However, despite this group of translators' best efforts to incorporate this feature into their workflow, this was quickly dropped.

Firstly, memoQ uses its own browser through the use of chromium as of its 8.2 version (MemoQ, 2017), which feels sluggish and lacks major quality-of-life features. Translators found it difficult to go back and forth as they wished or to open a new tab to do supplementary searches.

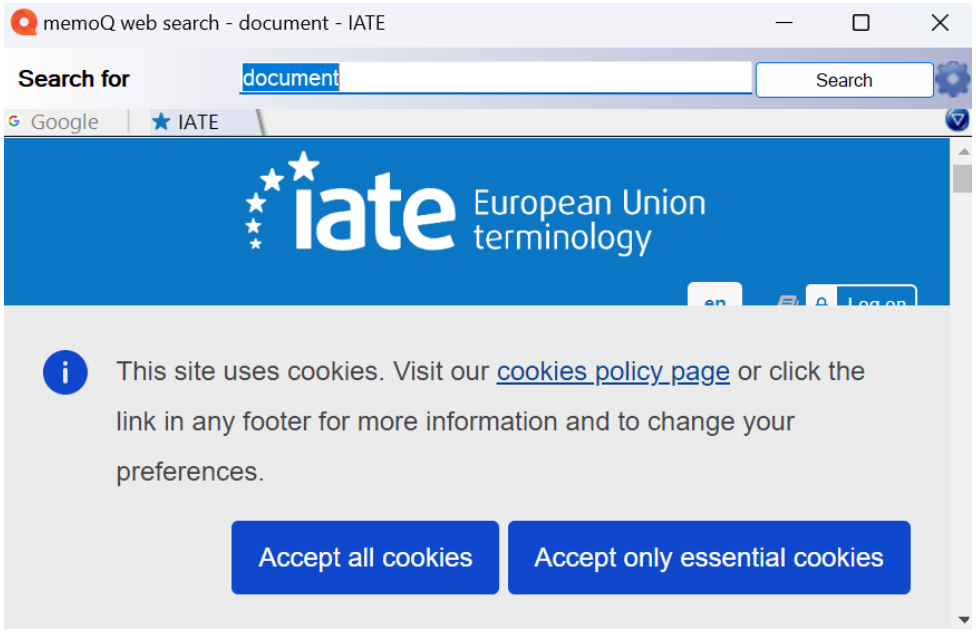

Figure 2. memoQ Web Search after user input.

Secondly, this specific translation team had a client who demanded links to every single term they used in their translation, which was not possible using memoQ Web Search as it does not offer its users the option to check or copy any of its URLs.

Furthermore, even if it did operate as needed, this feature is only present in memoQ, to the annoyance of many translators who wished to use this in other CAT Tools that are usually forced upon by clients.

An idea was instantaneously sparked. “What if memoQ Web Search used your browser instead?”. During my research, I was able to find an app similar to what I had envisioned named InteliWebSearch. However, it quickly became apparent that it did not quite fit the bill, as the application forced the usage of Internet Explorer, which is a browser that is both unpopular and discontinued as of 2023. However, this application ended up inspiring me to dust up my own programming skills and develop something similar that would fall in line with everything that was needed.

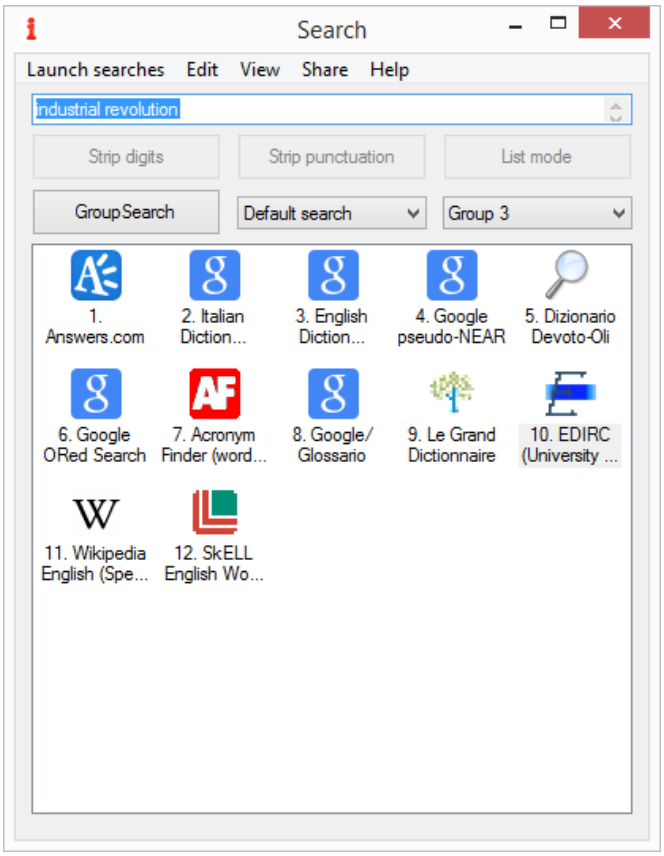

Figure 3. Screenshot of IntelliWebSearch.

I quickly drafted a simple prototype to serve as a proof of concept in a little under a week and presented it to the team, which was received with overwhelming positive responses. This positive reaction led to the whole application being greenlighted, allowing me to allocate more time and energy to this project.

3.1. The initial design

During the development of the first prototype, most decisions were made on the fly, as I just wanted to get the point across with a proof of concept. I wanted the app to be a more streamlined version of its memoQ counterpart without the unnecessary menus and submenus, a decision that ended up inadvertently shaping the identity of this software. For this purpose, I split the app into two sides, the left which would be reserved for everything regarding links, and the right side which would be reserved for term search.

I first created “link boxes”, based on memoQ Web Search’s Search URL boxes that behave just like memoQ, with the key difference being the use of the vertical bar (|) instead of the curly braces. The curly braces and vertical bars are part of a group called unsafe characters that are not present in any URL. These unsafe characters are crucial for

how these apps operate, as they serve as placeholders for the user's terms. Unsafe characters are a group of characters that can never appear in a URL (Berners-Lee, 1994) , meaning they act as the perfect placeholder for a term. Since no working URL as these characters, it means that as a placeholder, the software will never encounter a situation where they might accidentally replace them by accident.

The choice for the vertical bar was merely personal out of all unsafe characters, as I didn't want to use two or more characters like memoQ, as I thought, in the long run, it would be a hassle for the user. Furthermore, I find the vertical bar a good visual indicator, as it both stands out in any link and sells the idea that there is where the link will be split to place the term.

Figure 4. A link box with a link ready to be used in the app

While I enjoyed InteliWebSearch's use of icons instead of links that were always available to the user, I didn't want to commit as much effort at this point in development, and since I thought it was just easier for debugging purposes, I opted to just leave the link boxes as is on the left side of the screen.

Finally, on the right side of the screen, I went for a minimalistic approach, adding a simple textbox to insert the term the user wanted to search for and a big button underneath that simply read "Search" or "Pesquisar", depending on the build and version.

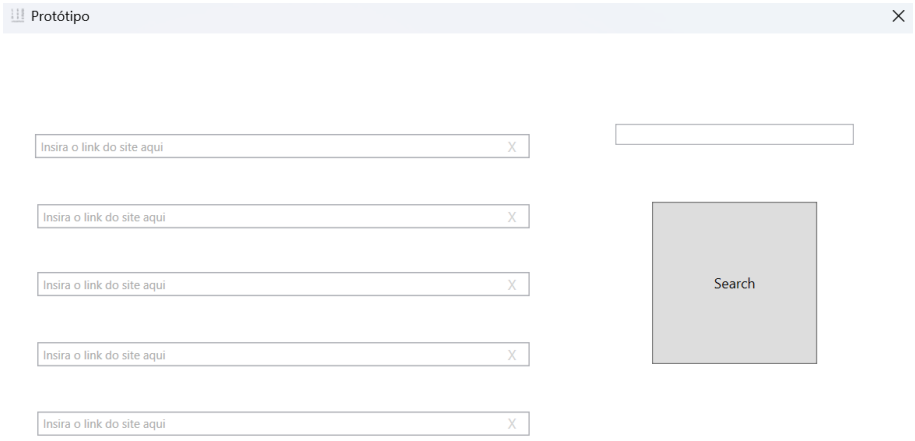

Figure 5. Image of the very first build of the CAT Nav app built around August 2023

As you may see, at this point in time, the application is very barebone as it served more as a proof of concept to gauge interest and decide whether or not to continue this endeavor. As the translators tested and gave their feedback, it became clear that the application was a success and the focus turned to materializing my other ideas coupled with the ideas pitched from users on possible new quality-of-life features.

3.2. The evolution of the app

Throughout its conception, the app flipped and flopped around using English and Portuguese as its main language as some users were native Portuguese speakers while others only understood English. Although the idea is to develop definitive versions for both languages, there is still much to focus on first. It quickly became apparent that it would benefit greatly from using icons as opposed to written words to convey features, thus avoiding most of these issues.

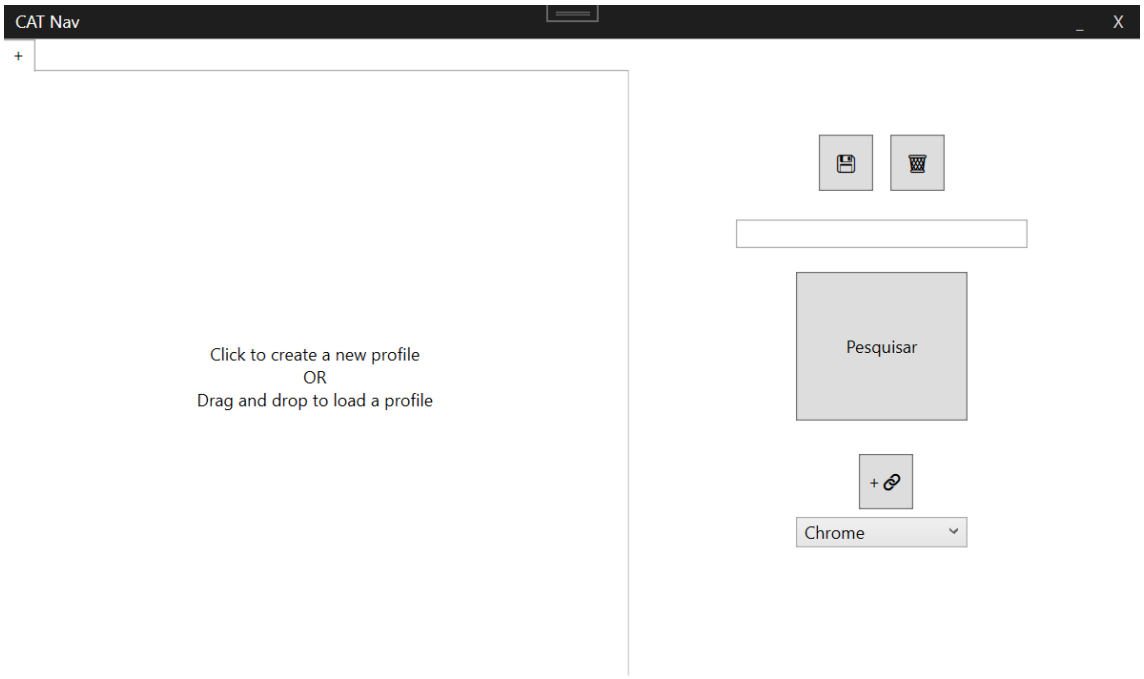

Figure 6. The most recent build of CAT Nav as of March 2023

In the most recent version as of the writing of this paper, you may observe the implementation of new features and buttons that use icons and symbols to convey universal ideas, such as the trash can to represent deleting link boxes, the floppy disk to save the current profile, and the plus sign followed by the icon of a chain to represent adding a link or link box.

One of the first features added was the ability to freely add and delete link boxes, as opposed to the fixed five in the first version. This was followed by the creation of profiles, tabs that users could create that would contain their own independent link boxes. This way, users could organize their links in any way they wanted, be it by language pair, type of website, field of expertise, and so on.

As time went on, the users started sharing more and more links with each other. To facilitate this, I created a feature that allowed users to share their profiles by saving them as separate files. To further drive the idea of keeping the application clean and menu-free, I opted to allow users to simply drag and drop their files into the application in an attempt to be as intuitive as possible.

Not long after, I added hotkeys to further improve user experience and to minimize the need to have the app window constantly open. This proved to be a little more difficult than anticipated, as shortcuts when an app is inactive or is not focused tend to not work. Nevertheless, I was able to find a solution, with the tradeoff that the user cannot change it themselves. As of writing this paper, my current goal is to add the ability for each profile to have its own unique hotkey and allow the user to fully customize this feature.

The most recently added feature was an experiment to only allow one browser window at a time. This feature came at the request of some users who found themselves flooded with many browser windows open simultaneously after researching a few terms and found it inconvenient to have to manually close a window every time they needed to do a new one. As it stands currently, when this option is checked, the app will close all instances of that said browser before performing a search. This is in turn a double-edged sword, as it can compromise any window or tab open that is unrelated to the search and may lead to accidentally closing them. From my research and attempts to find another solution, I found that it's extremely difficult, if not impossible, to keep track of all windows that are opened by the application, and consequently close only those windows. Furthermore, even if it was somehow possible, there would be no way to tell if the user had used any windows opened by the app for anything else and thus ineligible to be closed. Overall, it's a delicate problem with no sweeping solution that offers no downside, but it is one I am willing to revisit in the event of a possible breakthrough.

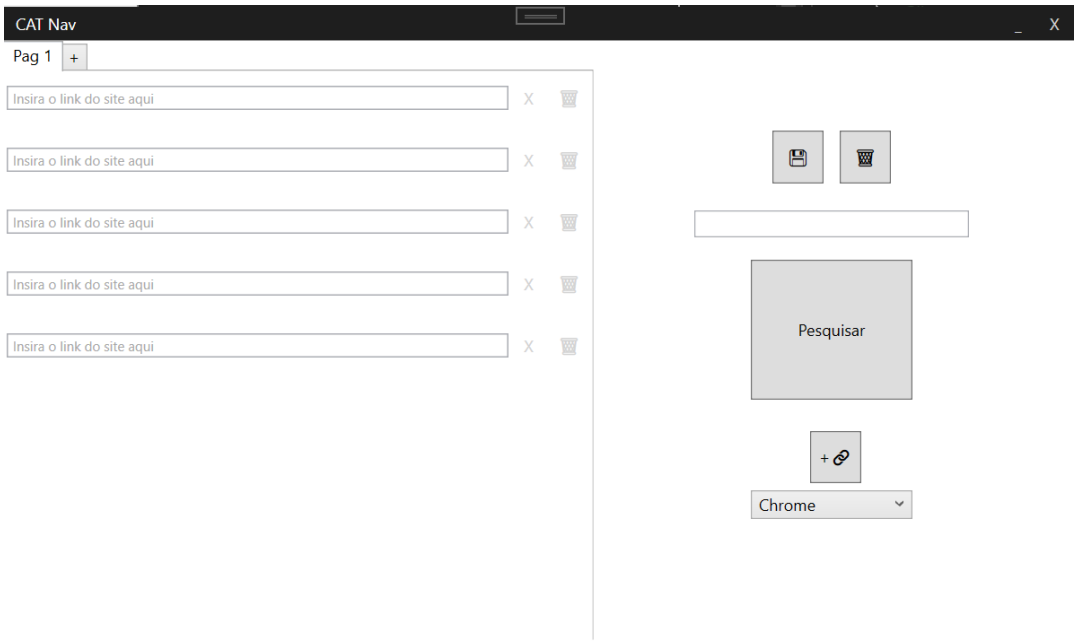

Figure 7. An example of a profile

At some point, the name CAT Nav occurred naturally to me, as it served as it served to navigate terms while using CAT tools. This name was later cemented when developing the second app, as you may find later in this paper, as I knew since the inception of this very first software that I wanted to brand all my applications in a way that was easily recognizable as being part of a whole.

3.3. The future of the app

CAT Nav, barring some features mentioned throughout this paper and bugfixes, from a technical standpoint, is mostly finished and unlikely to receive any new major updates when it comes to new content. The idea for this application was to keep it as simple and intuitive as possible from the cradle as I feel like memoQ Web Search and InteliWebSearch, two sources of inspiration for this app, feel too overcrowded and very beginner-unfriendly for the average translator that may or may not struggle with technology. I am satisfied when it comes to CAT Nav's identity and its features as it fulfilled every goal in a way that feels efficient and unique, despite its inspirations and, in a way, adversaries.

I intend on releasing this application to the public sometime in 2024, though there is still much that must be sorted out before this happens. Apart from the last additions and bug fixes, I intend on working on the UI and the visual presentation almost from the ground up, as I am currently dissatisfied with how it looks visually. Additionally, I still am not sure if I should commercialize this tool or not and, if I do, how to do so, as there are many angles I can aim for, from a monthly subscription to a one-time purchase and much more. I would also probably need to look into marketing and any other legal procedures which I am not acquainted with.

4. The second app: CAT Pal

The second app developed was aimed at terminologists and translators who love to perform long sessions of terminology research, creating or altering new or preexisting term bases on the fly. The idea came mainly from being exposed to numerous comments from both senior and junior translators about how dull, bothersome, and time-consuming it can be to copy and paste the content into an Excel file to later import into a CAT Tool and tailor those terms with the additional bells and whistles available within the tool.

It was clear from the beginning that this second app was going to be a lot more intricate and time-consuming for a myriad of reasons, some of which included the fact that it would need to interact with CAT tool resources directly and the inexistence of any similar application that I could draw inspiration. Although it was clear that this would be a much more demanding task, I had the added advantage of already having developed the CAT Nav beforehand, which ended up speeding up the process tremendously as I already had some experience.

However, before delving into the details of how this application was produced, it is imperative to understand the very foundation on which this project was built: Term Bases.

4.1. Term Bases: What they were

Terminological Databases, or Term Bases (TB), are a core essential of modern CAT Tools and a translator's life. On their official website, memoQ describes TBs as a "database containing single words or expressions related to a specific subject. Terms in term bases are often bilingual or even multilingual." (MemoQ6)

However, such an integral part of the translation cycle is far from being a recent development. We can find records as far back as the 80s of translators discussing their experience with this software, years before it was commercialized and accessible to the general public. Alain Paillet, a translator working at a pharmaceutical company, documented his experiences when his workplace decided to "step into the modern age of office automation" (Paillet, 1988, pp. 1-2) by introducing PCs and, most importantly, a new feature unlocked by these machines named simply "Termbase". It's important to remember that at this point in time, CAT tools as we know currently were mere pipe dreams idealized by some, such as Martin Kay in an article published in 1980 (Chan,

2015), as technical limitations only allowed for smaller scale applications (Vieira, 2021, p. 393).

Along the way, Paillet describes his journey and the hardships he encountered, such as not having enough memory to run both the terminological database and the word processing programs at the same time, a reality completely incomprehensible with modern our modern technology. This program also lacked features so minuscule, yet monumental that the average user takes for granted, such as copy and pasting. Paillet (1988) describes:

> A cut-and-paste facility has not been implemented as yet. We do not think that this is absolutely necessary because most of the time users do not take a term as it is retrieved from the database nor at the exact place where the cursor stands in the file. (p. 10)

From a technical point of view, we can also grasp that the information retained in these TBs is very much barebone in comparison to more modern versions, as we will discuss soon. At this point, the only information available regarding a term would be how it's written in a given language (in this case, only providing French, German, English, Italian, and Spanish) and its definition.

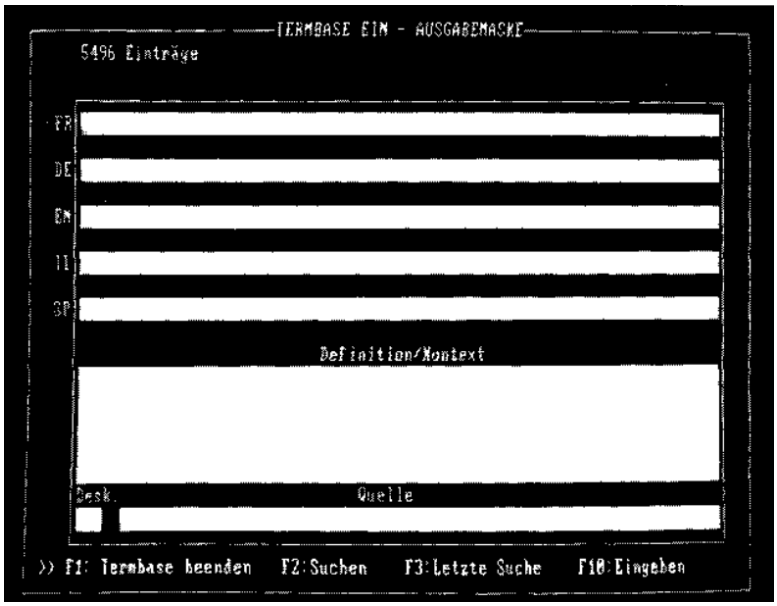

Figure 8. An image depicting a TB from the 1980s

4.2. Term Bases: What they are

In the present day, TBs have evolved. Long gone are the days when TBs served mostly as a simple dictionary that served as a visual reference for translators and terminologists alike. While this is still possible, more tech-savvy users are now able to leverage more out of this feature by using the plethora of new properties available that go hand in hand with other features offered by CAT Tools.

Such is the example of memoQ and its Forbidden Term property. Users are capable of assigning a term as forbidden within their TB. Reasons to use this feature vary, as it can be used to remind translators of specific language the client does not want to use, to highlight outdated or offensive terms, etc. This results in memoQ not only showing the term in a different color to note that the user should NOT use it, but it will also warn the user during a quality-assurance check (QA check) that the term was used and, depending on memoQ's settings, may even prevent it from exporting the finished product until the issue is resolved.

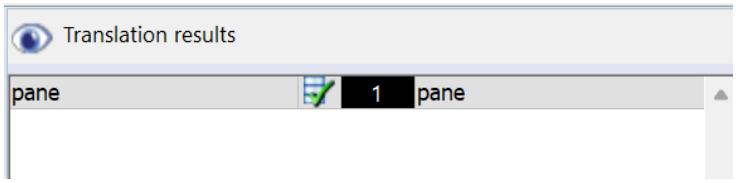

Figure 9. Forbidden Terms show up as black in memoQ and will signal in QA when present in the target text.

However, TBs are far from being homogenous. Although it is utilized in every single major CAT Tool, every single one of them operates differently, both in the properties it offers to its users and the their levels of customization, but also in how they work from a technological perspective internally.

Taking SDL Trados and memoQ as examples, the latter offers a plethora of different properties and fields as a default, including client information, notes, and more, whereas its counterpart does not. However, in turn, SDL Trados offers its users the ability to create new and original fields for any TB, although the amount of freedom it bestows is debatable.

Another point of contention is file support. Despite the existence of ISO 30042:2019 (Vezzani, 2021), which defines the existence of a TBX file, a type of XML

file that most CAT Tools support, not every single one can read or produce the same types of files for its TBs. As an example, while memoQ can both import and export Excel files for its TBs, SDL Trados is not.

4.3. Considerations before development

Before anything else, the idea is to expand later to produce different types of term bases to accommodate all major CAT Tools, such as SDL Trados, memoQ, and Phrase (formerly known as Memsource). However, due to time constraints and how daunting it would be to produce something worthwhile with equal results in multiple CAT Tools from a technical standpoint, the scope was reduced to only include memoQ's format. As to why memoQ exactly, the answer is due to being the tool I'm most experienced with, as well as the most compelling to work with, for reasons we will discuss later. In turn, this decision played an instrumental role in how I approached the development of this application as you may find later.

Furthermore, unlike the previous application, there was no real preexisting feature or application from which I could draw direct inspiration on how it should look and function. However, after studying other applications and CAT Tools, brainstorming, and refining ideas, I was able to come up with the first drafts of how this application would be.

4.4. Term Bases in memoQ

Firstly, I had to figure out what I was working with and what my building blocks were. In this case, it's terms and term bases. After analyzing how memoQ's term bases work by both experimenting with the tool and reading up on its support website, I was able to extract the properties that make up a Termbase and how they work and interact as highlighted in red in Figure 2.

a) How Term Bases are stored

The information that makes up each TB can be stored and edited in a multitude of ways. Currently, memoQ can save its TBs in XLSX (more commonly known as Excel), CSV, and MultiTerm XML files, as well as inside the tool itself.

The Excel (or XLSX) file extension is possibly the most well-known format for the average translator as it presents information in an easily readable and user-friendly

format. Each property is displayed in different columns, with the user choosing how many properties they wish to work with, while each line displays the information for each term.

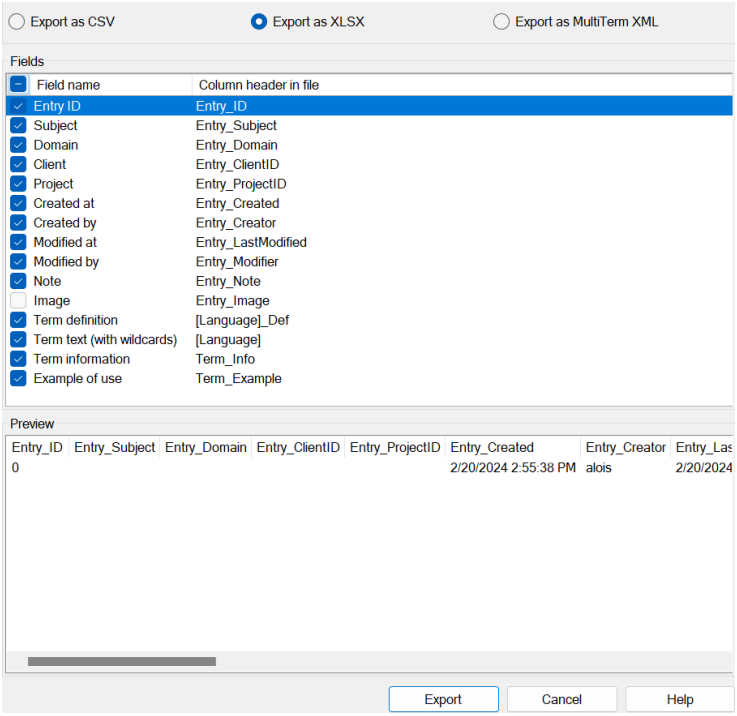

Figure 10. An image of memoQ's menu when exporting a TB as XLSX. Note how the image field cannot be ticked.

K	L	O	P
English_Def	**English**	**Portuguese_Portugal_Def**	**Portuguese_Portugal**
A pet	Cat	Um animal de estimação	Gato

Figure 11. A TB as an XLSX file.

Comma-separated values (CSV) files are not too dissimilar from Excel files. They are usually opened with Microsoft Excel as well but are far more difficult to interpret and read for the average user. As the name suggests, each property is separated by commas, instead of columns. The biggest selling point of this file format is its ability to be able to save and load images to memoQ natively, which is absent from every other file format. The images are stored in a folder named “attachment” and in a ZIP file with the TB.

	A	B
1	Entry_ID,Entry_Subject,Entry_Domain,Entry_ClientID,Entry_ProjectID,Entry_Created,Entry_Creator,Entry_LastModified,Entry_Modifier	
2	0,,,,,2/20/2024 2:55:38 PM,alois,2/20/2024 3:06:06 PM,alois,,A pet,Cat,CasePermissive	HalfPrefix,,Um animal de estimação,Gato,
3		

Figure 12. A TB as an XLSX file.

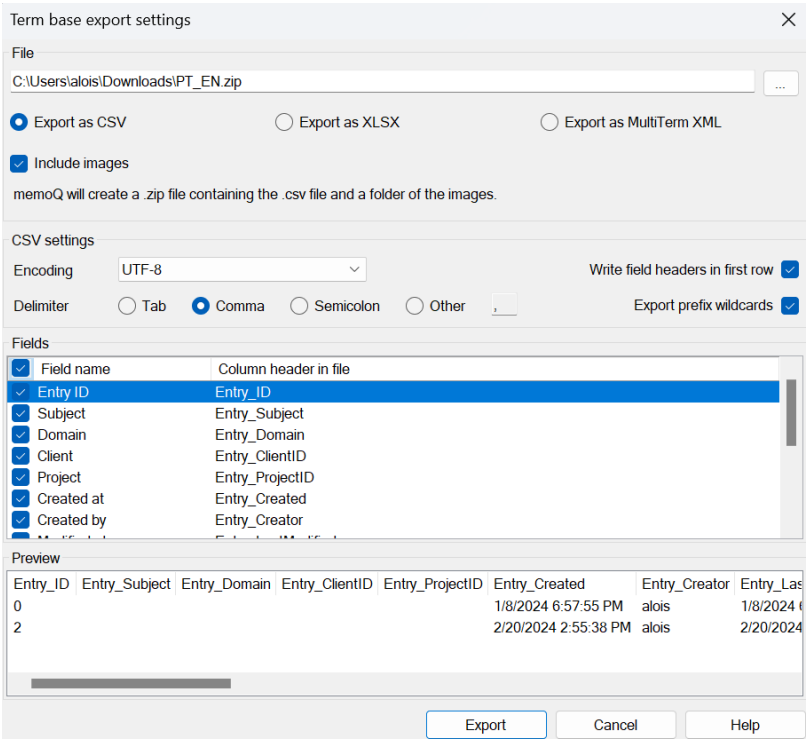

Figure 13. An image of memoQ's menu when exporting a TB as a CSV file. When the image field is ticked, memoQ will export the TB as a zip file with both the image folder and CSV file.

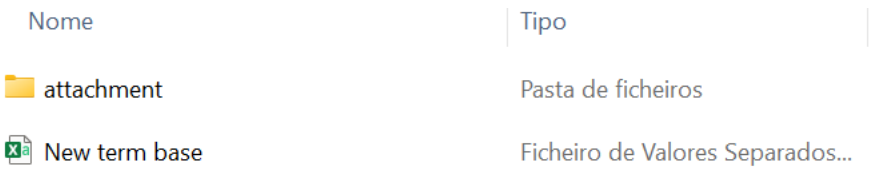

Figure 14. A CSV file with the folder containing the images in the TB inside the zip file.

MultiTerm XML is our third, and traditionally, last option. Despite the misleading name, this actually comprises more than just a simple XML file, as it also creates XDL and XDT files. The Multiterm format is used in SDL Trados, arguably the biggest CAT Tool outside of memoQ, which itself attempts to appease SDL's users by offering this compatibility. For the sake of keeping this section brief, I won't go into detail on how this works in that tool, rather I will use memoQ's own explanation of what XDL and XDT files are in the context of memoQ:

> XDL file is a layout definition file for SDL MultiTerm.(…)As memoQ term bases are of fixed structure (and custom fields from QTerm term bases are not exported),

XDL files always have the same format, and only language information can change according to the languages of the term base. (MemoQ4)

XDT is a definition file for SDL MultiTerm that describes the scheme of the term base(…) As memoQ term bases are of fixed structure (and custom fields from QTerm term bases are not exported), XDT files always have the same format, and only language information can change according to the languages of the term base. (MemoQ4)

Thus, the real meat of the TB lies in the XML files which are remarkably reliable powerful, and versatile, although alien to the average translator. However, it is not too difficult to grasp what is happening before their eyes upon closer inspection. The information appears between tags to categorize whatever is being described. For example, anything between <term> and </term> represents the term itself and anything between <descript> and </descript> represents the term's description.

```
<concept>1</concept>
<system type="entryClass">0</system>
<transacGrp>
        <transac type="origination">alois</transac>
        <date>2024-02-20T14:55:38</date>
</transacGrp>
<transacGrp>
        <transac type="modification">alois</transac>
        <date>2024-02-20T16:36:25</date>
</transacGrp>
<descripGrp>
        <descrip type="ID">0</descrip>
</descripGrp>
<descripGrp>
        <descrip type="Image">4938c7d3-1bad-47bf-aca4-77500996d2b2.bmp</descrip>
</descripGrp>
<languageGrp>
        <language type="Portuguese (Portugal)" lang="PT-PT"/>
        <descripGrp>
                <descrip type="Definition">Um animal de estimação</descrip>
        </descripGrp>
        <termGrp>
                <term>Gato</term>
                <transacGrp>
                        <transac type="origination">alois</transac>
                        <date>2024-02-20T14:55:38</date>
                </transacGrp>
                <transacGrp>
                        <transac type="modification">alois</transac>
                        <date>2024-02-20T16:36:25</date>
                </transacGrp>
        </termGrp>
</languageGrp>
<languageGrp>
```

Figure 15. A look inside a TB as an XML file.

Finally, there is an additional hidden fourth method. In the tool itself, each property is stored separately as MTX files for each language within the TB and a single MTB file. To access these files one must access the memoQ files inside the hidden folder "ProgramData" within the PC, which requires plenty of legwork from the user to access under normal circumstances as they are not usually meant to be accessible.

The MTX files are structured in hexadecimal, meaning that they cannot be read as plain text files. The MTB file serves as the key that determines how the remaining files should be saved and opened. In this state, while it is possible to edit the information, it is extremely difficult to do so efficiently without accidentally compromising the file itself. The only feasible solution is to utilize reverse engineering, which is against the terms of service of memoQ, and thus impossible to achieve.

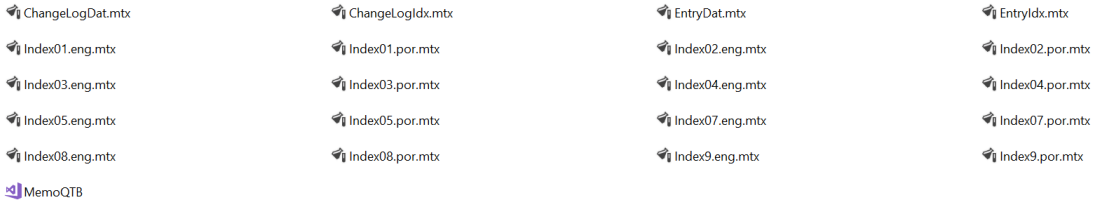

Figure 16. How a TB looks inside memoQs directory folders.

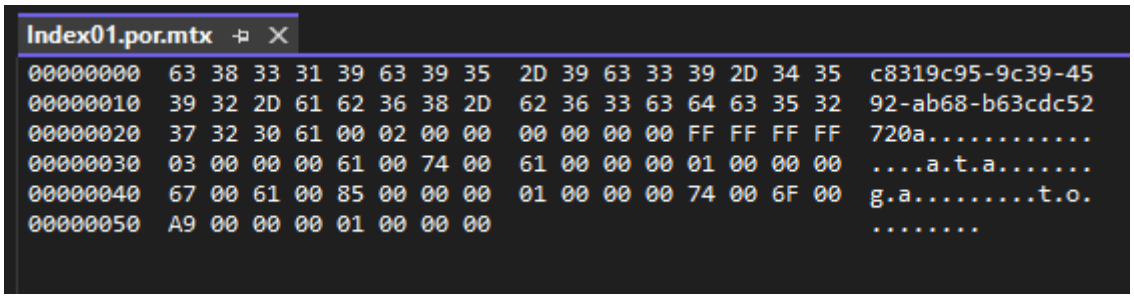

Figure 17. A look inside an MTX file pertaining to a TB field, namely the name of the terms in Portuguese.

b) Properties in a Term Base

As explained in the official documentation, the properties that make up a TB can be broken down into three separate categories. These include Entry-level properties (ID, Note, Project, Domain, Created by, Modified by, Client, Subject, Created at, Modified at, and Image), Language-level properties (which refers to the term itself and the definition given to each language), Term-level properties (Matching, Case sensitivity, Example, Part of speech, Gender, and Number) (MemoQ5). Although there is a lot of ground to cover

for each individual property, I will be limiting myself to explaining some of the more relevant and frequently used few whenever the occasion asks for.

Each property functions differently, and to find out how exactly, I tested thoroughly both how the CAT Tool created the TB file and how it reads it. From my findings, I was able to divide these properties into two distinct groups, which I labeled "Open Ended" and "Closed Ended" due to their nature for easier reference.

The "Open-ended" group is the easiest group to explain. This refers to the properties that store the user's input as a string, or in layman term's, the text as it is. The most obvious examples are the terms themselves and their definition, as they are displayed in the same fashion both inside the CAT Tool and in the files it produces. Most Entry-level and Langugae-level properties are part of this group.

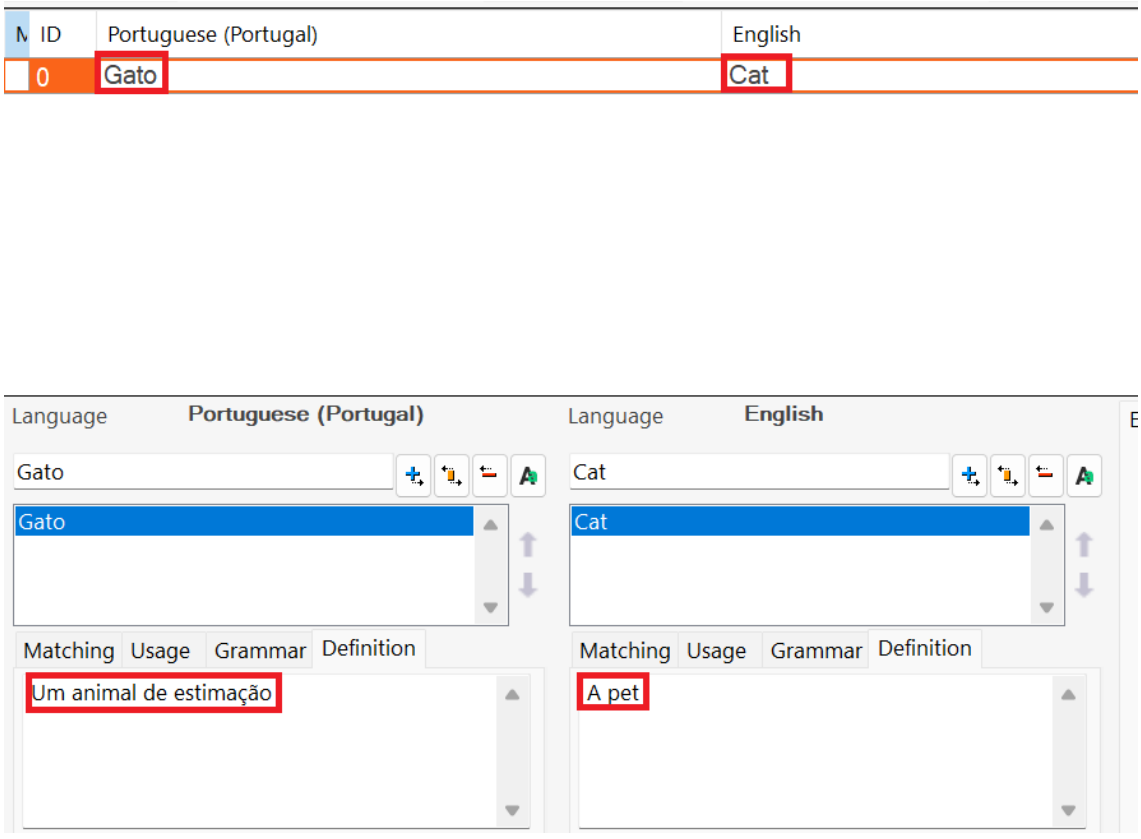

Figure 18. How properties are stored in a TB inside memoQ.

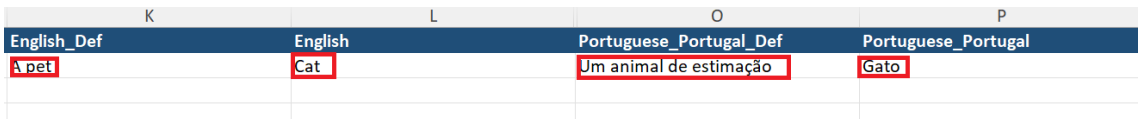

Figure 19. How those same properties are stored in a TB in an XLSX file

On the other hand, we have the “Close Ended” group which is not as straightforward as its counterparts. These properties are very much binaries and possess only one way to be represented both inside and outside the tool. Most Term-level properties are part of this group. For example, inside memoQ, the parameter of case sensitivity for a given term can *only* be “Yes”, “Permissive” or “No”, and “CaseSense”, “CasePermissive “, and “CaseInsense”, respectively when stored outside the tool. Any deviation in spelling or use of multiple options will result in an error as it will not be read by memoQ when importing the file.

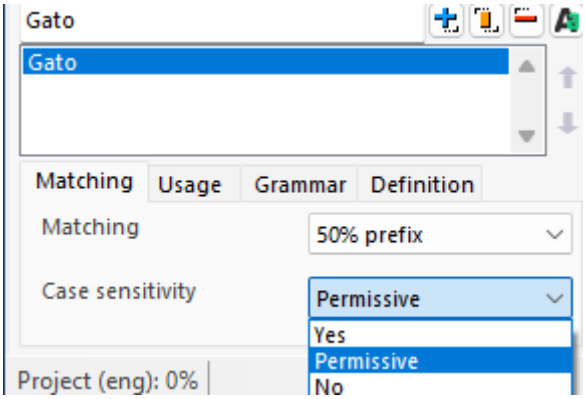

Figure 20. Case sensitivity in memoQ

P	Q
Portuguese_Portugal	**Term_Info**
Gato	CasePermissive;HalfPrefix

Figure 21. How data about case sensitivity is depicted in an XLSX file.

4.5. The initial design

After gathering all the information I could about how TBs work and deciding to limit myself to just memoQ, I began drafting my initial designs for how the application would look and operate.

I decided almost immediately that I wanted it to be an overlay app, or at least offer it as an option. An overlay application refers to an application whose window is constantly on top of every other window, regardless of cursor focus. I concluded that this would be a vital quality-of-life feature as users could browse and use any other application of their choosing to their heart's desire without the constant need to bring the application to the front and so they could easily monitor any changes they were doing to a given term or

TB. However, for the user's convenience, I decided to give the option to turn this property on or off, as I understand this could be something not everyone is used to and could alienate some users.

I also wanted the windows to be as compact as possible if it's going to be constantly on the forefront of the user's screen. However, due to the number of properties I needed to include for each term, I had to make a decision. I would have to either give up on some properties, or create more menus, or find another solution. In the end, I came up with the idea of dividing these properties into different panels and then having each panel slide and be able to hide behind each other. Not only did this solve my initial problem, but it also solved an issue later on in this project, as I worried that having these many properties presented to the user would result in an information overload and end up intimidating translators who may or may not be used to all memoQ's bells and whistles.

The layout of each panel and how properties would be split up came almost naturally. I decided that each panel would have a hierarchy, I wanted the main panel to be comprised of properties that were crucial to a TB and that almost every translator was familiar with. The second panel would have grammatical information that although not every user may know it's present in memoQ, they could certainly figure out what it means thanks to their knowledge of linguistics. Finally, the last panel would be reserved for memoQ-only data that was dispensable for the average user.

My study on how memoQ handles their TBs was extremely useful in this phase. As you may remember, I split every property into two different groups: "Open Ended" and "Close Ended". For "Open-ended" properties I used textboxes, a type of control element in GUIs that allowed users to type whatever they wished and store it as information. On the other hand, I used combo boxes for "Close Ended" properties, another type of control element in GUIs that grants users the ability to choose between various predetermined options.

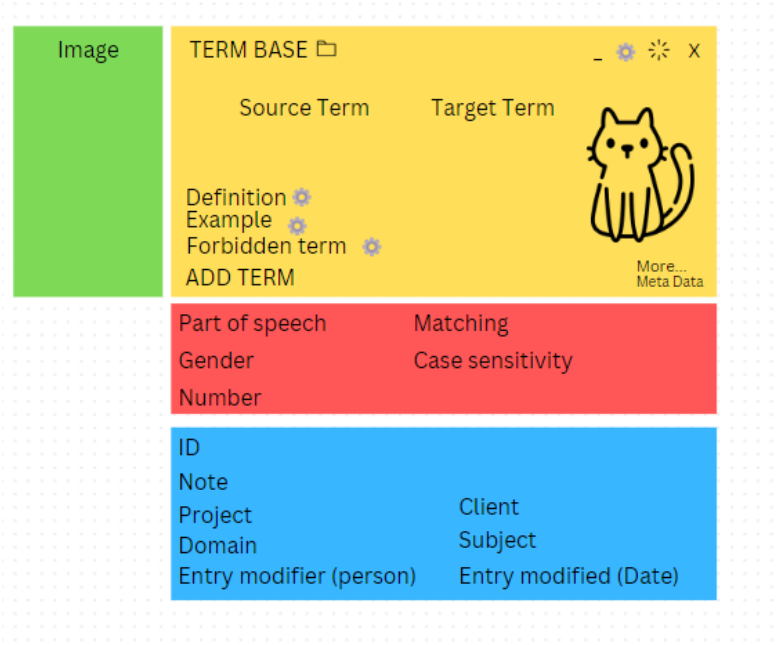

Figure 22. The very first visual draft for CAT Pal was done around December 2023

4.6. The evolution of the app

Once I had everything I wanted to do planned out, I started coding away. Along the way I made a few changes to some elements that either didn't work out as I hoped or because I came up with better solutions. Such an example would be how the forefront property was handled and how it communicated to the user that the application couldn't be minimized or put behind any other application. After many trials and errors, I ended up using a lock icon to communicate when the app is or is not acting as an overlay by showing it locked or unlocked.

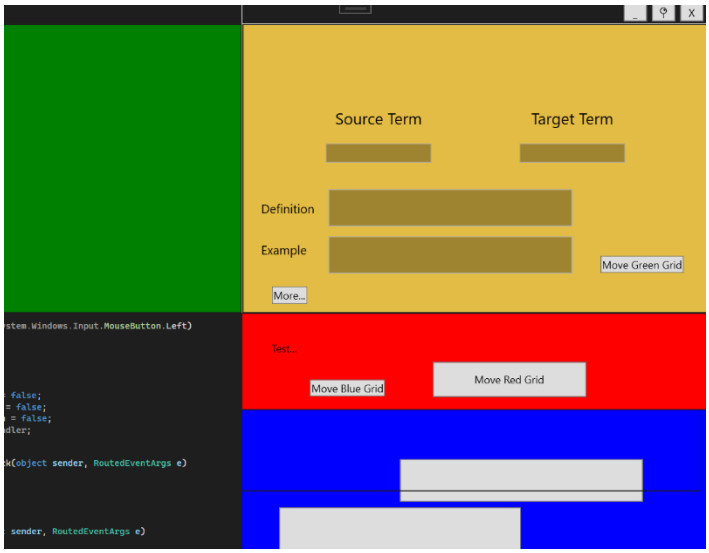

Figure 23. An extremely early build meant to test the moving grids feature. At this time, a pin was used to represent the overlay property.

Figure 24. A more recent build that uses a lock instead of a pin.

Firstly, I decided to limit myself to just working with XLSX files. When I first started to tackle this project, I envisioned using the Term Base SDK (Software development kit) developed and provided by memoQ themselves. The idea would be to be able to edit any local TB without the need to import or export them. However, after dabbling with the SDK, it came to light that I couldn't really fulfill my vision with this tool, as it is aimed at creating terminology plugins. In short, these terminology plugins are to be installed within your memoQ and would provide terminology from an external database (MemoQ, 2015). I even thought of still going through with this idea by having the terms be stored in that database, separate from any TB, but I concluded that not only would it defeat the purpose of the app, but it would also be very difficult to implement and for users to install.

In the same vein, I also briefly considered editing the TBs in memoQ, but it was short-lived as I realized it would require reverse engineering, thus breaking the terms of service of memoQ.

After exhausting my options, I decided to go for one of the three formats memoQ can handle. Since I could not edit or create TBs directly into memoQ, I figured I should go with the next best option and pick a format that translators could work with and

understand with ease. Thus, I picked the XLSX format due to how common and readable they are to even an untrained user. This also sadly meant that I had to abandon the idea of adding images as a feature since this was a CSV-exclusive feature which I concluded that it was not worth sacrificing user readability for an unpopular, albeit useful feature. With this said, it is a feature that I am willing to reconsider for a future version outside the scope of this current project.

After some more studying of how memoQ handles XLSX files and trial and error, I was able to recreate exactly how a file is produced by this tool. This ranged from simple aesthetic elements, such as the color of the first row and font used, to more meaningful aspects, such as the order in which each property typically appears.

For example, the column that represents a term is named [LanguageCode] (e.g. pt-pt or en), while its definition is named [LanguageCode_Def] (e.g. pt-pt_Def or en_Def). Unlike other columns whose names are static, these were dynamic and changed depending on whichever language the user chose, which posed an issue. As such, I simply created a dictionary within the app with all languages supported by memoQ as found on their official website (MemoQ3). This list comprised over one hundred entries that were linked to their respective 2-letter code to be used when creating the Excel files.

```
public readonly Dictionary<string, string> languageDictionary = new Dictionary<string, string>
{
  {"Afrikaans", "af"},
  {"Akan", "ak"},
  {"Albanian", "sq"},
  {"Albanian (Albania)", "sq-AL"},
  {"Albanian (Kosovo)", "sq-XK"},
  {"Albanian (Macedonia)", "sq-MK"},
  {"Albanian (Montenegro)", "sq-ME"},
  {"Amharic", "am"},
  {"Arabic", "ar"},
  {"Arabic (Algeria)", "ar-DZ"},
  {"Arabic (Bahrain)", "ar-BH"},
  {"Arabic (Egypt)", "ar-EG"},
  {"Arabic (Iraq)", "ar-IQ"},
  {"Arabic (Jordan)", "ar-JO"},
  {"Arabic (Kuwait)", "ar-KW"},
```

Figure 25. A peak inside CAT Pal's code depicting language support.

As for each property, the user could add as much or as little information as they wanted, which would be saved within the application and would be eventually transferred into the XLSX file whenever needed. Open ended properties would retain any text written down and close ended properties did a similar trick to the titles of columns where they would have had every option available in memoQ written down as such in combo boxes and would be saved as their counterpart in the final file. Special mention of the Forbidden

Term property should be made, as instead of being written down, it is highlighted in red, which works functionally the same when importing to memoQ and reads better to the user when reviewing their TB.

en	en_Def	pt-pt	pt-pt_Def
Dog	House pet	Cão	
Dog		Cachorro	Animal de estimação
Cat		Gato	
Horse		Cavalo	

Figure 26. A file produced by CAT Pal, mimicking how a similar file would be produced by memoQ.

Perhaps one of the more eye-grabbing details in the app was the inclusion of an actual cat as a design element. As seen in the first visual draft, this idea was implemented from the start. I wanted to have a visually striking element and decided on a cat due to its close affinity with CAT tools due to spelling. Originally, it was merely meant to cover a blank space that I struggled to find a purpose for, meant to be deleted sometime during development.

As time went on, I found myself attached to the digital pet, which struck me as a charming detail that users could also appreciate. At one point, I came to the realization that I could capitalize on this asset if I really wanted to keep it, as it can be used to convey information to the user. The cat's eyes could serve to indicate to the user that the application is running smoothly as it will follow the cursor. Its tail could communicate if the application is loading something or if a file was successfully created. Its nose could serve as a button for a possible feature.

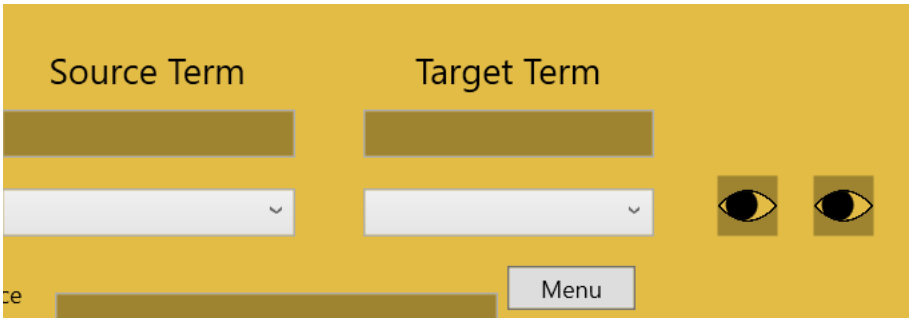

Figure 27. Slightly earlier debug build of CAT Pal testing the eye-moving feature.

Regardless of its function, I decided to put it on the back burner for the time being, as other features and debugging took priority, but I still wanted to go through with this idea since I believe it could create a good first impression on users and could be an

interesting way to pull attention to this app. As of the writing of this paper, only the eye-moving mechanic was implemented, with the rest to be done sometime in the future, alongside other visual assets.

Eventually, I was able to come up with a name for the application. I wanted to link both applications thematically, and with the inclusion of a cat in this application, I knew it had to have "CAT" somewhere too. Furthermore, I wanted to really dive into the idea of having a companion while using this application since terminology search can be a very time-consuming task. Not only that, but this app would also serve as a companion to memoQ in the form of a peripheral. Thus, the name CAT Pal was given rather naturally.

Finally, after some feedback and debugging cycles, the UI suffered a few natural changes. The main difference is the ID number field, as I found that despite being an entry-level property and largely ignored by most translators, it is still crucial to put it in the main pane. The reason being that it is directly linked to each term in a very fundamental way, as you cannot have one without the other. This became even more clear when it proved to be difficult and sluggish to delete or edit any term previously added, as there wasn't a readily available way to view terms within the tool. Being able to scroll through terms by their ID, made this task much easier and intuitive.

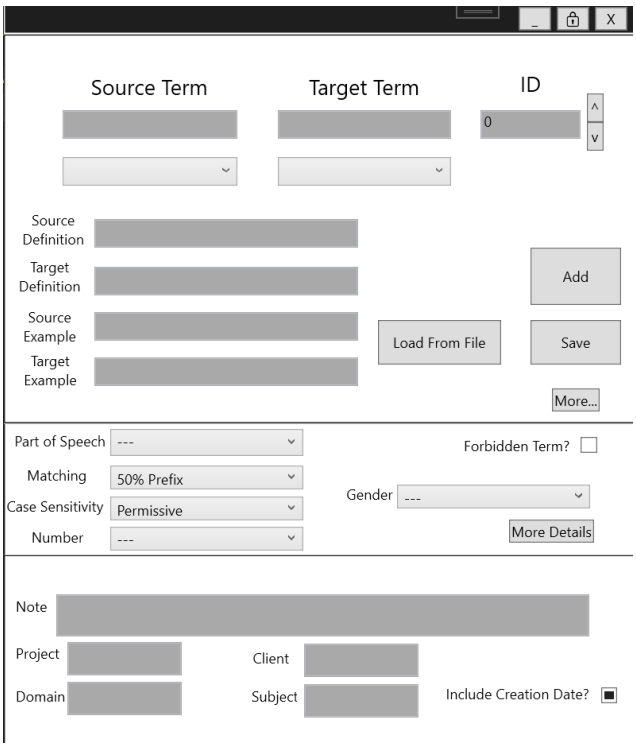

Figure 28. An image of the most recent CAT Pal build. The most noticeable change is to the ID Field, which is now given as much focus as the Source and Target Terms fields.

4.7. The future of the app

As mentioned, CAT Pal is much more ambitious than the CAT Nav, as it is, as far as I can tell, the only application like it. While having no direct application or feature to draw direct inspiration proved to be significantly more challenging than the previous app, it also proved to be the most rewarding and fun to work with.

Much like the CAT Nav, when it comes to its aesthetic and UI, there is still much to be achieved. I am content with its current design and layout from a mechanical standpoint, especially with the ability to slide each tab, but I also recognize that it still needs a big overhaul before I can even consider releasing this to the public.

For its features and content, there are also a few implementations to be made, as mentioned throughout the paper. The main implementations I have envisioned going forward involve more CAT tool compatibility, specifically Trados, although it is possible major revisions may need to be done for this to happen, as each CAT tool is different in how they handle their terminology, as explained extensively before.

Due to all these reasons, I don't really envision a release in the near future. Although there is no concrete plan or timeline, the idea is to test the waters with CAT Nav and hopefully garner some interest before releasing CAT Pal.

Furthermore, due to CAT Pal's nature and uniqueness, I believe it would benefit greatly from having native support from a CAT tool. It would be significantly more efficient if it could create and modify term bases from memoQ itself, skipping the need to create files to be imported into the tool. However, for this to happen, I would need to get in touch with memoQ and strike a deal, which I don't believe would be possible currently as I am nowhere near recognizable enough to be given the chance or time.

5. Conclusion

Translation technology plays a pivotal role in the world of translation, from the biggest translation company giant to the average freelance translator. As such, it is vital that these technologies evolve alongside its users.

Localization engineers are the missing link between translators and programmers as they hope to make the translation work as seamless and effortless as possible by offering tailored solutions to their problems.

Localization engineers also serve to steer translation technology in the right direction to obtain the desired results by either directly manipulating it or creating tools of their own. This sentiment, which still holds to this day, can be felt from as far back as the 1980's as Paillet (1988) claims:

> Generally speaking, the potential direction of CAT seems to lie in the development of a series of modular tools for the translator, tools that are not so complicated as to force the translator to become a programmer, but tools that perform those routine tasks that make the translation a drudge. This naturally involves close cooperation between practicing translators, applied linguists, and programmers. Termbase is the result of such cooperation. (1988, p. 12)

I hope that this paper along with my applications have shed some light on some aspects that are for the most part not talked about in the translation sphere, at least academically. The process of creating an application can be a gargantuan task with a multitude of moving parts to be tended to.

These applications represent significant advancements in the realm of translation technology, offering translators the means to enhance both the quantity and quality of their work not offered by any other tool in the market. The CAT Nav, for instance, allows users to save an unfathomable amount of time by launching multiple searches with a single keystroke. Meanwhile, CAT Pal provides a seamless and intuitive platform for crafting meticulous term bases, empowering and encouraging users to refine their terminology with precision.

In essence, these tools serve as natural extensions of the translators' existing CAT tool experience, broadening their interaction with software environments beyond traditional boundaries.

CAT Nav and CAT Pal stand as testament to the importance of tools designed by and for translators. They embody the notion that many of the challenges faced in translation are not insurmountable, and that keen observation and understanding of translators' daily struggles can lead to significant breakthroughs.

While I currently hold no quantitative data on the impact of the apps developed in this project, I can attest to the satisfaction of the team that served as test users. Their enthusiasm for each update, along with their invaluable input on improvements and bug fixes, underscores the tangible benefits these tools have brought to their workflow. It's been humbling to witness their gratitude and appreciation for the efficiencies gained through these technologies.

As development of these tools continues beyond the completion of this paper, my hope is that upon their official public release, they will serve as valuable assets to translation teams and freelancers alike, much as they have benefitted the team at Linguaemundi.

Looking ahead, I envision CAT Nav and CAT Pal as just the beginning of a larger journey. There remains ample room for further innovation and collaboration within the translation technology community. I am inspired by the contributions of individuals such as Michael Farrell, the creator of IntelliWebSearch, Loek van Kooten, the mind behind Cattitude, an independent CAT tool, Kevin Lossner, a giant in the translation technology industry, and countless others who are dedicated to building tools that empower translators to work more efficiently and effectively. These tools have the ability to push the boundaries of what is possible in the field of translation.

As time goes on, technology comes and goes, but translation stays. Technology is as powerful as it is ephemeral. New tools will emerge every day as older tools become obsolete. The most important thing is to learn from past and present tools to steer future translation technology into the right future. A future that frees translators from their burdens and allows them to blossom into the most productive and efficient professionals possible.

Figures

References

Berners-Lee, T. M. (1994). *Uniform resource locators (URL).*

Chan, S.-w. (2015). *Routledge Encyclopedia of Translation Technology: Towards a World without Babel.* London: Routledge.

Esselink, B. (2002). Localization Engineering: The Dream Job? *revista tradumàtica*.

Mayorcas, P. (1988). Proceedings of Translating and the Computer 10: The translation environment 10 years on. *ACL Anthology*, 10.

MemoQ. (2015). *Terminology SDK.*

MemoQ. (2017). *Discover memoQ 8.2*. Retrieved from memoQ: https://docs.memoq.com/9-4/en/Welcome/welcome-discover-memoq-8-2.html

MemoQ1. (n.d.). *Edit search provider*. Retrieved from memoQdocs: https://docs.memoq.com/current/en/Places/edit-search-provider.html

MemoQ2. (n.d.). memoQ *web search*. Retrieved from memoQdocs: https://docs.memoq.com/current/en/Places/memoq-web-search.html

MemoQ3. (n.d.). *Supported Languages.* Retrieved from memoQDocs: https://docs.memoq.com/current/en/Things/things-supported-languages.html

MemoQ4. (n.d.). *Term base in SDL MultiTerm XML format.* Retrieved from memoQ Server Docs: https://docs.memoq.com/current/api-docs/wsapi/memoqservices/tbservice.importexport.multiterm.html

MemoQ5. (n.d.). *Term bases - Inside an entry*. Retrieved from https://docs.memoq.com: https://docs.memoq.com/current/en/Things/things-term-bases-inside-an-entry.html

MemoQ6. (n.d.). *What us a Term Base?* Retrieved from memoQ: https://www.memoq.com/tools/what-is-a-termbase

O'Hagan, M. (2013). The impact of new technologies on translation studies: a technological turn? In *The Routledge handbook of translation studies* (pp. 503-518). Routledge.

Paillet, A. (1988). *Proceedings of Translating and the Computer 10: The translation environment 10 years on.* Retrieved from ACL Anthology: https://aclanthology.org/1988.tc-1.12.pdf

Pym, A. &. (2006). Technology and Translation. A pedagogical overview. In *Translation Technology and its Teaching (with much mention of localization)* (pp. 5-19).

Pym, A. (2010). *Exploring Translation Theories.* Abington: Routledge.

Somers, H. (2003). *Computers and Translation.* Amsterdam and Philadelphia: John Benjamins.

Vezzani, F. (2021). On the reusability of terminological data. *AlmaDL Journals*, 183-186.

Vieira, E. A. (2021). The impact of technology on the roke of the translator in globalized production workflows. In L. Bowker, *The Routledge Handbook of Translation and Globalization* (p. 393). London and New York: Routledge.