FACULTAT DE TRADUCCIÓ I D'INTERPRETACIÓ
GRAU DE TRADUCCIÓ I D'INTERPRETACIÓ

TREBALL DE FI DE GRAU
Curs 2016-2017

Localització web.
Anàlisi i estandardització de la traducció de text sense estructura dins de codi informàtic

Helena Garrido Terrats
1197077

TUTORA
PILAR SÁNCHEZ-GIJÓN

Barcelona, 4 de juny del 2017

Universitat Autònoma de Barcelona

Dades del TFG

Títol: Localització web. Anàlisi i estandarditzación de la traducció de text sense estructura dins de codi informàtic
Títol: Localización web. Análisis y estandarización de la traducción de texto sin estructura dentro de código informático
Títol: Web localization. Analysis and standardization of translating non-structured text within computer code
Autora: Helena Garrido Terrats
Tutora: Pilar Sánchez-Gijón
Centre: Facultat de Traducció i d'Interpretació
Estudis: Grau de Traducció i d'Interpretació
Curs acadèmic: 2015-2016

Paraules clau

tradumàtica, tecnologies de la traducció, localització web, eines TAO, llenguatges de programació, Python™, pretraducció, pseudocodi, estandarditzación, automatització
tradumática, tecnologías de la traducción, localización web, herramientas TAO, lenguajes de programación, Python™, pretraducción, pseudocódigo, estandarización, automatización
translation technologies, web localization, CAT tools, programming languages, Python™, pre-edition, pseudocode, standardization, automation

Resum del TFG

Aquest Treball de Fi de Grau pretén trobar una solució pràctica per a un encàrrec de traducció simulat que presenta una sèrie d'irregularitats. El text d'origen són comentaris escrits per i per a enginyers dins del codi informàtic d'una aplicació. El conjunt dels comentaris està mancat de consistència lingüística i de format, la qual cosa fa difícil poder tractar l'encàrrec de traducció com un de convencional. Per tal d'estandarditzar el text d'origen i donar-li el format necessari per a poder-hi treballar amb una eina TAO, s'han avaluat els diferents obstacles i solucions possibles. Finalment, s'ha dut a terme la solució més eficaç possible. Aquesta solució ha consistit en el disseny d'un petit programari informàtic que ens ajuda a extreure els comentaris del codi per a poder-los traduir amb una eina TAO i, posteriorment, tornar-los a inserir al codi. Per a dissenyar el programari, se n'ha redactat el pseudocodi.

Este Trabajo de Fin de Grado pretende encontrar una solución práctica para un encargo de traducción simulado que presenta una serie de irregularidades. El texto de origen son comentarios escritos por y para ingenieros dentro del código informático de una aplicación. El conjunto de los comentarios carece de consistencia lingüística y de formato, lo cual hace difícil poder tratar el encargo de traducción como uno convencional. Para estandarizar el texto de origen y darle el formato necesario para poder trabajar en él con una herramienta TAO, se han evaluado los diferentes obstáculos y soluciones posibles. Finalmente, se ha llevado a cabo la solución más eficaz posible. Esta solución ha consistido en el diseño de un pequeño software que nos ayuda a extraer los comentarios del código para poderlos traducir con una herramienta TAO y, posteriormente, volverlos a insertar en el código. Para diseñar el software, se ha redactado su pseudocódigo.

This thesis seeks to find a workable solution to the problems of a simulated translation project with very special features. The source text are comments written by programmers and intended to other programmers within the source code of an application. Such comments lack of format and linguistic consistency, which makes it hard to work with the text under ordinary conditions. In order to standardize the text and give it the right format, so that it can be translated using a CAT tool, we have analyzed all the obstacles of the text and the possible solutions to our translation problem. Finally, the most suitable solution has been to design a simple software able to extract comments from the code, in order to translate them with a CAT tool, and then reinsert them into the code. To design our software, we have written its pseudocode.

Avís legal

TAULA DE CONTINGUTS

ÍNDEX DE SIGLES

API: Application Programming Interface

CAA: Comunicació Augmentativa i Alternativa

CAT: Computer-Assisted Translation

CSV: Comma Separated Values

GILT: Globalization, Internationalization, Localization and Translation

ISO: International Organization for Standardization

MT: Machine Translation

PE: Postedició

QA: Quality Assurance

ST: Source Text*

TA: Traducció Automàtica

TAO: Traducció Assistida per Ordinador

TFG: Treball de Fi de Grau

TT: Target Text*

*S'ha preferit l'ús de la combinació *ST*/*TT*, i no pas TO/TM (Text d'Origen/Text Meta), perquè TM es pot relacionar amb el terme en anglès *Translation Memory*.)

ÍNDEX DE FIGURES

1. INTRODUCCIÓ, OBJECTIUS I MOTIVACIÓ PERSONAL

La paraula que obre el títol d'aquest Treball de Fi de Grau, 'localització', en anglès *localization* o *L10N*, descriu l'acció d'adaptar un producte, normalment una aplicació informàtica, a la llengua i cultura d'un mercat determinat (Pagans, 2002). És una de les disciplines més noves dins del camp de la traducció però, tot i la seva breu existència si la comparem amb altres àmbits de la traducció, en pocs anys ha obert un espai de mercat nou i immens per als traductors i traductores, que avui dia tenen l'opció d'enfocar la seva carrera professional cap a una pràctica molt demandada que es troba en plena expansió. Des de finals dels anys 80 (Inge, 2006), la localització de programari ha experimentat un creixement fort que ha dut a incloure aquesta disciplina entre els diversos vessants de la traducció professional. La consolidació de l'ús de les eines informàtiques ha provocat una forta demanda d'aquest servei i també de l'anomenada 'internacionalització', és a dir, l'adaptació del desenvolupament d'un programari amb relació a la seva futura localització. De fet, segons Inge (2006: 173), actualment la venda de productes localitzats suposa el 50 % dels ingressos de les principals empreses de programari.

En efecte, aquest Treball de Fi de Grau (en endavant, TFG) girarà entorn de la localització però, no obstant això, se centrarà en un camp d'aplicació poc tractat: la traducció dels comentaris incrustats dins del codi font d'un programari o un lloc web. Concretament, el cos principal del treball es basarà en l'elaboració d'un pseudocodi —és a dir, "la redacció en llenguatge informal, inspirat en l'estructura, sintaxi i vocabulari d'un llenguatge de programació, utilitzat per a descriure un algorisme o procediment sense entrar en els detalls específics de programació en un llenguatge concret" (TERMCAT, 2016)— per a dur a terme l'estandardització, revisió i traducció de comentaris escrits per programadors dins del codi font d'una aplicació. El fet és que ens trobem davant d'un problema tècnic de traducció i, per a proposar-hi una solució, caldrà aprendre a expressar-la d'una manera tècnica, perquè un programador la pugui llegir i entendre, sense ambigüitats (és a dir, amb pseudocodi). De fet, aquest aprenentatge també és part de les competències del traductor, tenint en compte el fort component tecnològic que té aquesta professió.

En el marc teòric del treball, a tall d'introducció als comentaris dins del codi informàtic, en primer lloc es definiran alguns conceptes generals de l'àmbit de les tecnologies de la traducció. També s'exposarà quina és la funció i la rellevància dels

comentaris dins del codi informàtic. Tot seguit, es farà un resum de l'estat actual de l'art en el camp de la programació informàtica i en el marc de la nova metodologia de programació *Agile*.

Pel que fa a l'encàrrec de traducció simulat que servirà d'exemple al llarg de tot el treball, en primer lloc es descriurà l'encàrrec de traducció real en què es basa i el seu context, englobat dins el camp de les aplicacions d'accessibilitat i la Comunicació Augmentativa i Alternativa (CAA). A continuació, es descriurà el *ST* i s'analitzaran els problemes que presenta, tant lingüístics com de format. També es descriurà el *TT* i s'establiran els objectius i criteris de l'encàrrec de traducció simulat, com ara la traducció dels comentaris amb una eina TAO (és a dir, fora del codi informàtic) i la introducció d'una etiqueta per a identificar-ne l'idioma, entre d'altres.

Pel que fa a la sintaxi d'introducció i tancament de comentaris dins del codi informàtic, se n'exposaran diferents tipus. La descripció del *ST* i del *TT* servirà per a definir, completament, quin és el nostre punt de partida i on volem arribar.

El següent pas serà establir un ordre de prioritats, com ara la rendibilitat del projecte i la comprensió del missatge per sobre de la correcció lingüística. Tenint en compte aquests criteris, es valoraran diverses solucions possibles per a solucionar el problema de traducció presentat i se n'escollirà una de manera raonada i argumentada. Seguidament, s'analitzaran diferents estratègies tecnològiques: la traducció humana, la traducció automàtica (TA), full *post-editing* and *light post-editing*.

El marc empíric del treball consistirà en l'elaboració del pseudocodi d'un programari que solucioni el problema de traducció presentat. Pel que fa a l'algorisme del programari, per ara només podrà ser utilitzat amb codi escrit en llenguatge PHP, HTML, JavaScript i CSS però, en canvi, no es limitarà únicament a un parell de llengües específic sinó que es podrà aplicar a qualsevol combinació d'idiomes. A l'últim apartat del marc empíric es descriuran algunes de les millores que es podrien fer al programari en el futur.

La temàtica d'aquest TFG i l'objectiu que es pretén assolir és una mena de repte personal basat en un cas real propi. Fa menys d'un any vaig rebre un encàrrec de traducció no professional que consistia a revisar, traduir i donar el format correcte als comentaris que es trobaven intercalats amb les línies del codi lliure d'una aplicació web anomenada Jocomunico[1]. Després d'aconseguir separar els comentaris amb text traduïble del codi

[1] PAHISA-SOLÉ, Joan. (2016). *Jocomunico. L'app de CAA que parla amb naturalitat*. [lloc web]. Disponible a: <http://jocomunico.com/#/home> [Consulta: 23 de desembre del 2016]

font gràcies a una eina TAO o de Traducció Assistida per Ordinador, vaig plantejar-me si hi devia haver alguna manera d'automatitzar el procés de revisió, traducció i formatació per tal de fer-lo més eficient i àgil. El text al qual m'enfrontava era extens (el codi estava format per més de 20.000 línies, encara que no totes eren comentaris) i, per si això no fos prou, alguns dels comentaris estaven escrits en català, d'altres en castellà i d'altres en anglès, la qual cosa dificultava la utilització d'una eina TAO. La feina que m'havia estat encarregada consistia a fer que tots els comentaris estiguessin escrits en català i en anglès perquè un programador extern al projecte pogués comprendre el codi i, si ho desitjava, contribuís en el seu desenvolupament.

La pregunta "com te'n sortiries si rebessis un encàrrec de traducció real i remunerat amb aquestes característiques?" va ser la que va em va dur a investigar sobre quin camí caldria seguir per a agilitzar el procés i com es podrien resoldre les necessitats traductològiques dins del context de les noves tecnologies i la programació informàtica. Es tractava d'un encàrrec complex i ja intuïa que caldria aprofundir-hi força si volia trobar una solució als reptes que es plantejaven. Així es va concebre la idea que ha donat lloc a aquest treball, l'objectiu fonamental i prioritari del qual és aconseguir elaborar un pseudocodi que resulti útil per a la revisió, traducció i formatació de comentaris escrits per programadors dins de codi informàtic, destinats a lectors especialitzats (normalment altres programadors) perquè puguin comprendre totes les accions que realitza el codi i puguin col·laborar en el seu desenvolupament, per a mantenir-lo o enriquir-lo.

2. MARC TEÒRIC

2.1. Localització, internacionalització i globalització

El procés que descriu aquest treball es troba emmarcat en l'àmbit de la localització, tal com hem vist a l'apartat d'introducció. La localització es considera un dels passos dins de la cadena de processos que es coneix amb les sigles GILT: *Globalization*, *Internationalization*, *Localization and Translation* (Anastasiou, Schäler, 2010). Tenint en compte que sovint els tres primers termes de la sigla s'utilitzen de manera sinònima erròniament i que existeix certa confusió al seu voltant, fugirem de qualsevol mena d'ambigüitat basant-nos en les definicions que en donen els experts, com ara Corte Fernández (2002) a l'article *Localización e Internacionalización de sitios web*:

> [La internacionalización] Consiste en la identificación de toda la información local que aparece en un sitio web, es decir, aquella información que viene dictada por el idioma y la cultura del país donde se diseñó originalmente. Por ejemplo fechas, números, moneda, información de contacto, etc. Estos elementos deberán aislarse y guardarse de forma independiente para que sea posible adaptarlos a las especificaciones de cualquier idioma. (Corte Fernández, 2002: 1)

> [La localización] Es el proceso de adaptar un sitio web a un idioma y una cultura diferente. Esto significa mucho más que simplemente traducir el contenido de las páginas. El contenido de una página web está formado por texto e imágenes, ambos deben ser traducidos y sometidos a una adaptación cultural. El usuario nunca debe notar que ese sitio fue originalmente creado en otro idioma. (Corte Fernández, 2002: 1)

> La globalización combina los procesos de internacionalización y localización. Consiste en el diseño de sitios web que pueden ser utilizados en diferentes países con un mínimo de cambios. Es un concepto que pertenece más al área del marketing que al área técnica. (Corte Fernández, 2002: 1)

Així, en projectes grans de localització, la internacionalització de llocs web o aplicacions suposa canvis en el disseny i la funcionalitat dels productes i una despesa econòmica i temporal addicional que es veu compensada a llarg termini, atès que facilita la localització del producte a la cultura dels països o mercats on té cabuda. Qualsevol empresa de programari o desenvolupament de llocs web que tingui intenció de treballar per a clients d'altres països haurà d'incloure, imperativament, la fase

d'internacionalització com a pas inicial dins del procés de localització dels seus productes.

Ara bé, Corte Fernández explica que, a l'hora de localitzar un lloc web o una aplicació, cal diferenciar clarament la funcionalitat de la interfície, atès que és només el segon d'aquests elements, la interfície, el que s'haurà de traduir, tenint en compte que és la part que veu l'usuari. La part funcional, és a dir, el codi font, queda oculta i, per aquest motiu, no és necessari sotmetre-la al procés de localització (Corte Fernández, 2002).

És precisament en aquest punt on rau la diferència entre un projecte de localització "estàndard" (tal com l'entenem d'acord amb les definicions que hem vist) i el nostre projecte en particular. Aquest TFG pretén localitzar, i fins i tot internacionalitzar, un dels elements d'aquesta 'part oculta' que esmenta Corte Fernández (2002). En concret, el nostre objectiu és crear un algorisme que permeti la traducció dels comentaris inserits dins del codi font d'aplicacions o llocs web i, alhora, que l'algorisme estigui preparat per a la introducció de qualsevol combinació d'idiomes en el futur. Aquesta última característica del nostre projecte és la que més ens recorda al concepte d'internacionalització. No es busca la localització del text traduïble que es troba entre l'hipertext de les aplicacions, sinó que es proposa la traducció del text que no té a veure amb la interfície, al qual l'usuari final no accedirà mai. La nostra missió és localitzar el text que no s'acostuma a localitzar i que el resultat contribueixi a millorar la comunicació entre desenvolupadors.

En el seu article, Corte Fernández (2002) fa una reflexió sobre els nous reptes de localització, els quals fan que cada dia apareguin al mercat eines noves que aporten solucions per a oferir la informació en altres idiomes. D'alguna manera, la cita següent podria ser un bon resum del cas proposat en aquest treball:

> Estados Unidos y el Reino Unido ya no son los únicos usuarios de Internet. El aumento de su uso en otros países presenta nuevos retos y requiere nuevas soluciones para ofrecer información en otros idiomas. Este proceso no sólo significa traducción sino que también implica adaptación cultural y la superación de varios problemas técnicos. Se debe adoptar una nueva forma de diseñar y crear sitios web que permita una fácil adaptación a otros idiomas. El proceso de localización de sitios web aún está desarrollándose y nuevas herramientas aparecen a diario en el mercado. En estos momentos cada proyecto de localización es único. (Corte Fernández, 2002: 7)

2.2. Funció i rellevància de la documentació del codi informàtic

Per a començar a abordar l'encàrrec de traducció a què ens enfrontem, primerament cal aclarir què són els comentaris, per a què serveixen i on rau la seva importància.

Encara que els comentaris de codi font són textos dels quals no sentim a parlar gaire, probablement pel seu registre tècnic sovint molt complex, *comentar* el codi informàtic és una pràctica absolutament normalitzada i estesa entre els programadors informàtics. En anglès, el conjunt de comentaris i anotacions fets al codi, juntament amb documents externs que descriuen detalladament el disseny i les funcionalitats del programari, s'anomena *documentation* (Knuth, 1992: 99) i, en endavant, farem servir la traducció d'aquest terme, 'documentació' o 'documentar', juntament amb els termes 'comentari' o 'comentar' per a referir-nos a aquest tipus de textos. A tall de definició, podem dir que els comentaris són missatges i anotacions que el programador o programadors insereixen al llarg del codi informàtic sense que el seu contingut afecti les accions que executa el codi (Dennett, 2015).

D'una banda, aquestes explicacions serveixen per a comunicar a altres lectors humans allò que el programador tenia al cap quan va escriure les línies de codi a què el comentari fa referència. Afegir comentaris al codi també és recomanable per al mateix programador que els escriu, atès que és fàcil que acabi oblidant el raonament que havia seguit per a escriure aquella part de codi. A més, en cas d'haver de tornar a una part del codi per a corregir-ne possibles errors (tant si ho fa l'autor del codi com un altre programador), els comentaris poden ser d'allò més útils (Dennett, 2015). Per a disposar d'una definició més tècnica, ens fixarem en la que recull Flaig (2011: capítol 6.1), el qual explica que un comentari és una part del codi font introduïda per una marca que el compilador interpreta com a text que no ha de ser executat. D'aquesta manera, es poden incloure anotacions que contribueixen a fer el codi més intel·ligible (per a altres programadors i pel mateix programador que l'ha escrit). Flaig assenyala que els comentaris no tenen res a veure amb les accions que genera el codi però, en canvi, són una característica destacada del codi ben escrit.

D'altra banda, Roturier (2015) opina que la documentació de codi font pot ser útil també per als traductors d'aplicacions i llocs web, especialment per a compensar la manca de context: "*Comments, however, can be extremely useful during the localization process,*

especially when translators do not have access to the context (i.e. access to the page of the application containing a particular string)." (Roturier, 2015: 65)

2.3. Els comentaris com a element central del codi

Si bé alguns autors, com ara Dennett, consideren que la *documentació* del codi és recomanable, n'hi ha d'altres que hi atribueixen una gran importància i, de fet, consideren que la *documentació* hauria de ser el cos principal del codi informàtic. Amb relació a aquest enfocament, trobem un programari anomenat CWEB[2], desenvolupat per Donald Knuth el 1981, que combina els llenguatges de programació C i C ++ amb el sistema de tipografia TeX. TeX és el llenguatge de programació nucli en què es basa el conegut sistema de composició de textos LaTeX, el qual s'ha convertit en l'estàndard tipogràfic dins dels àmbits acadèmic i científic. El programari CWEB ofereix una nova manera de programar, més ben *documentada*, en què el text redactat pels programadors queda en primer pla mentre que el codi informàtic queda ocult en segon pla. Knuth és un dels màxims representants del paradigma de programació anomenat *Literate Programming*, sota el qual es considera que el programador no escriu codi informàtic sinó més aviat un assaig i, per això, n'ha de cuidar bé l'exposició del contingut i l'estil (Knuth, 1992: 99). Per dir-ho d'una altra manera, el desenvolupador que adopta aquest paradigma, el *literate programmer* (Knuth, 1992: 99), no escriu codi informàtic acompanyat de comentaris, com és habitual, sinó al contrari. A la seva obra, titulada amb el mateix nom, *Literate Programming* (1992), Knuth afirma: "*Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.*" (Knuth, 1992: 99). Així, el paradigma de programació proposat per Knuth, encara que actualment no és el predominant en l'àmbit del desenvolupament de programari sinó que és un paradigma força minoritari, defensa una concepció de la programació que posa en relleu la importància dels comentaris dins del codi.

[2] Knuth, Levy (1993). *The CWEB System of Structured Documentation*. Disponible a: <http://www-cs-faculty.stanford.edu/~uno/cweb.html>

Un altre cas particular que exemplifica la rellevància de la *documentació* del codi en el procés de desenvolupament d'un programari és el llenguatge de programació HASKELL, el qual disposa d'un mode de programació, anomenat *literate HASKELL* (Flaig, 2011), que inverteix els papers dels comentaris i el codi. Així, parteix dels comentaris com a cos principal del text i és el codi informàtic el que ha d'anar introduït per una marca que el distingeixi de cara al compilador.

Per acabar, dins d'un discurs més radical, ens fixarem en l'argument que dóna Yourdon (1988), qui opina que no *documentar* el codi és un pecat imperdonable:

> In my opinion, there is nothing in the programming field more despicable than an uncommented program. A programmer can be forgiven many sins and flights of fancy [...]; however, no programmer, no matter how wise, no matter how experienced, no matter how hard-pressed for time, no matter how well intentioned, should be forgiven an uncommented and undocumented program. (Yourdon, 1988: 73)

Així, doncs, podem veure que els comentaris dins del codi no són simples anotacions complementàries que els programadors poden decidir afegir o no, sinó que, en alguns casos, són de gran importància per a esmenar possibles errors i perquè altres programadors puguin entendre com funciona un programa sense haver-hi treballat mai abans. A tall de conclusió d'aquest apartat, ens fixarem en l'afirmació de Zokaities (2002) a la seva obra sobre desenvolupament de programari *Writing Understandable Code*:

> […] As I gradually improved my in-code documentation, I realized that English is a natural language, but computer languages, regardless of how well we use them, are still "code." **Communication via natural language is a relatively quick and efficient process**. **Not so with computer languages**: They must be "decoded" for efficient human understanding. (Zokaities, 2002: 48-49)

2.4. La documentació del codi en el marc de la metodologia de programació Agile

Fins ara hem vist què són els comentaris del codi informàtic, per a què serveixen i la importància que té *documentar* el codi segons autors com Knuth (1992) o Yourdon (1988). No obstant això, aquests autors i altres dels esmentats en aquest treball van formar la seva visió de la *documentació* del codi ara fa més de vint anys. Aquest interval de temps, que pot ser curt des d'una perspectiva històrica, esdevé pràcticament una eternitat si el traslladem a l'àmbit de les tecnologies i la programació. Des dels anys 80 fins ara, i especialment a partir de l'any 2000, el camp de la programació ha experimentat canvis profunds en la manera de concebre el procés de desenvolupament de programari i en la relació que hi ha entre els diferents perfils professionals que intervenen en el procés. En aquest apartat, reflexionarem sobre el lloc que ocupen els comentaris del codi informàtic a la nova era de la programació, la qual tendeix cada vegada més cap al treball col·laboratiu que es realitza directament al núvol.

Avui dia, de manera paradoxal, ni tan sols l'enginyeria informàtica i el desenvolupament de programari no es troben exempts dels efectes del ritme trepidant amb què avancen les noves tecnologies. Fins fa poc més d'una dècada, la metodologia utilitzada de manera majoritària a la indústria del desenvolupament de programari ha estat el "model predictiu" (CollabNet Inc., 2011: 4), altrament conegut com a *Waterfall Method* (Cohen et al., 2003: 3). A grans trets, aquest procés està format per les fases següents:

1. Anàlisi dels requisits del programari: després de rebre l'encàrrec del client, els programadors defineixen un conjunt de requisits o funcionalitats que ha de tenir el futur programari. Per a fer-ho, hi ha un període previ d'interacció amb usuaris i consumidors en el qual expressen les seves necessitats i experiències.
2. Disseny: amb tota la informació recopilada a la fase anterior sobre les funcionalitats del programari, un equip de desenvolupadors dissenya fins a l'últim detall de l'arquitectura que tindrà i elabora una gran quantitat de documentació on se n'expliquen tots els detalls.
3. Desenvolupament: els programadors implementen informàticament tot allò que s'ha especificat a la fase anterior i "donen vida" al programari.
4. Proves: després de passar per un procés de proves o *testing*, el programari es lliura al client com un producte tancat que compleix els requisits inicials. És en

aquest punt, en una fase avançada del projecte, on el client veu, pràcticament per primera vegada, el resultat tangible de la seva inversió.

Aquest mètode, alhora, implica la traducció del programari com a un únic encàrrec, és a dir, l'agència de localització o traducció rep diferents paquets amb segments de text que, una vegada han estat traduïts, s'envien al client perquè els incorpori al programari.

Tot aquest procés, però, ha demostrat ser massa rígid i poc resistent als canvis i a l'evolució —característiques inherents a la indústria del desenvolupament de programari. Una de les fonts principals dels canvis, que sovint es produeixen a la fase de desenvolupament, respecte del que havia estat dissenyat és la voluntat del client, que depèn directament de les necessitats d'usuaris i consumidors. Aquestes necessitats o opinions canvien constantment a causa de la velocitat amb què sorgeixen nous aparells i tecnologia que els usuaris incorporen a la seva vida, remodelant la perspectiva del consumidor respecte d'altres productes del mercat. A més, el model predictiu es basa en l'anticipació de les variables que poden intervenir en el procés de producció de programari. És per això que les empreses de desenvolupament de programari tenen greus dificultats a l'hora d'introduir modificacions dins d'un pla de treball altament calculat i poc flexible:

> "In a typical Waterfall process, everything is fixed—scope, functionality, cost, time, and quality dimensions. Waterfall is also heavily reliant on accurately forecasting and anticipating all technical and business issues that will surface during the development process. The hope—a leap of faith—is that all will go exactly as planned. Of course, this is not possible." (CollabNet Inc., 2011: 4)

Així, un petit canvi en els requisits del programa comporta importants alteracions de la documentació on es recull la informació de les fases d'anàlisi i disseny i, posteriorment, una càrrega molt important de feina afegida per l'equip comercial i l'equip de programadors. En resum, la gestió ineficaç sumada a la complexitat d'aquest tipus de projectes es tradueix en temps —un temps "perillós" a causa de l'avantatge que dóna a les empreses competidores i del risc d'obsolescència que corre el projecte— i, en darrer terme, en diners:

> "Having put time and effort into the requirements […] becomes problematic for a couple of reasons. First, more time will elapse before coding production begins since the technical or design teams will proceed cautiously, and slowly, to analyse the requirements and develop better estimates. Second, what if, as most often happens, the requirements don't sync with real-world, unforeseen technical or business issues?" (CollabNet Inc., 2011: 4)

Tots aquests "desavantatges" del *Waterfall Method* han provocat que en els darrers 15 anys hagin sorgit iniciatives que defensen noves metodologies basades en l'anomenat "model adaptatiu" (CollabNet Inc., 2011: 4) de desenvolupament de programari. Aquest model *s'adapta* a les necessitats del mercat i busca la interacció amb el client i amb usuaris durant el procés de desenvolupament més que no pas el compliment d'un pla de treball establert, que pot acabar esdevenint totalment aliè a la realitat del mercat. A més, aquesta nova corrent estableix que el potencial de les empreses de desenvolupament de programari ve determinat per la seva capacitat de crear programes i aplicacions de gran qualitat en el menor temps possible. Per a fer-ho, els desenvolupadors i tots els equips humans que intervenen en el procés (com ara el departament de TI (Tecnologies de la Informació), el de disseny, el de qualitat, el comercial, el financer, etc.) han de treballar de manera paral·lela, deixant enrere el sistema tradicional, en el qual el producte ha de passar per diferents fases i diferents equips de treball d'un en un, *en cascada*.

Per tal de proposar una alternativa definitiva que encarnés la filosofia dels mètodes adaptatius, l'any 2001 va néixer l'*Agile Alliance*. L'objectiu d'aquesta aliança, formada per un grup d'enginyers als Estats Units, era trobar el nou camí que havia de prendre el desenvolupament de programari dins la filosofia de treball *Agile*, un terme que no només s'aplica al desenvolupament de programari sinó que defineix una nova concepció de la gestió de projectes en general.

Com a resultat de les aportacions de nombrosos experts, es va redactar el *Manifesto for Agile software development* (Agile Alliance, 2015), en què es recullen, a tall de resum, els principis següents:

- Persones i interaccions per sobre de processos i eines.
- Programari que funciona per sobre de documentació exhaustiva.
- Col·laboració amb el client per sobre de negociació de contractes.
- Resposta al canvi per sobre del seguiment d'una planificació.

El punt clarament més fort d'*Agile* és la comunicació amb el client, la qual cosa es tradueix en la interacció amb els usuaris, atès que el client basa les seves condicions i peticions en el *feedback* o retroacció que rep dels usuaris. Això ajuda els desenvolupadors de programari a tenir en compte possibles punts febles del projecte, sigui quin sigui el punt del procés en què es trobi. També proposa un nou mètode de treball, que consisteix en la divisió de la planificació del projecte en petites tasques o fites, anomenades *sprints* o iteracions (Dimes, 2015), les quals tenen un objectiu concret i una duració limitada. La feina resultant de cada iteració és revisada diverses vegades per l'equip de TI fins que s'obté el resultat desitjat. Aleshores, aquesta iteració deixa de formar part de la feina pendent i esdevé una part totalment operativa del producte final que s'entregarà al client. D'aquesta manera, el projecte pràcticament es revisa i s'actualitza alhora que es desenvolupa, cosa que fa que el temps de producció es redueixi. A això, cal sumar-hi la retroacció dels usuaris i la coordinació amb la resta de departaments, amb la qual cosa el resultat serà un producte desenvolupat en poc temps, a l'alçada de les expectatives dels futurs consumidors i que està preparat per a llançar al mercat. Dins del paraigua de la metodologia *Agile* s'inclouen altres mètodes de gestió de projectes que en pocs anys han pres una rellevància considerable, com ara *DevOps* (Mora Pérez, 2015) o *Scrum* (Dimes, 2015).

Una vegada hem entès en què consisteix la metodologia *Agile* i quins canvis implica respecte del model clàssic de desenvolupament de programari, ens sorgeixen diverses preguntes. D'una banda, si ens centrem en el camp principal en què s'emmarca aquest treball, ens preguntem: en quin punt té lloc la localització dins de la nova metodologia de treball paral·lel i col·laboratiu? Si s'actualitza el codi font del programa de manera freqüent, com ho fa l'equip de traductors per a aconseguir actualitzar la traducció amb els segments nous i no perdre el control dels segments traduïts i els no traduïts? A més, cal tenir en compte que aquesta manera de desenvolupar el programari, on professionals especialitzats en diferents camps treballen conjuntament, ha fet que, inevitablement, es tendeixi a traslladar el desenvolupament dels projectes al núvol:

> "[...] Agile also requires an automated development infrastructure to support continuous integration. This can be complex and difficult to maintain. Having a development platform in the cloud can simplify many of these issues in a cost-effective way." (CollabNet Inc., 2011: 4)

De manera resumida, dins de la metodologia *Agile,* el procés de localització del producte avança alhora que avancen els *sprints* o iteracions, de manera que l'empresa de localització rep amb freqüència segments nous d'un programari per traduir (probablement menys extensos que en el cas del *Waterfall Method),* els quals han de ser traduïts en un període curt de temps. Ara bé, en aquest treball no aprofundirem en com afronten les empreses de localització la nova *Agile localization*[3], atès que és un camp complex que encara té un llarg recorregut per endavant. Aquest tema, que està directament relacionat amb el nostre propòsit, podria ser el subjecte d'una futura investigació paral·lela a aquest TFG.

D'altra banda, ens sorgeixen preguntes respecte de la gestió dels comentaris al codi informàtic dins de la metodologia *Agile*. Si ens basem en el fet que aquest mètode posa en relleu la importància de la comunicació entre diferents equips de treball (encara que estiguin formats per professionals de diversos àmbits, sovint fins i tot situats a diferents llocs del món) i que bona part dels projectes es desenvolupa al núvol, no esdevenen encara més necessaris els comentaris al codi informàtic? Però, si ho pensem bé, tenint en compte que un dels aspectes de la programació tradicional que critica *Agile* és el temps que perden els desenvolupadors elaborant una gran quantitat de documentació dels programaris, els comentaris dins del codi també es consideren una 'pèrdua de temps'? En definitiva: com tracta la metodologia *Agile* aquesta part del codi?

La resposta a totes aquestes preguntes és que, probablement per la influència d'*Agile* i altres metodologies que comparteixen la mateixa essència, actualment es tendeix a considerar que els comentaris són una part redundant del codi que, en general, fa 'perdre el temps' als programadors. Encara que és important aclarir que aquest punt de vista no s'expressa al *Manifesto for Agile software development* (Agile Alliance, 2015), sí que es fa referència a la reducció de la gran quantitat de documentació que s'ha d'elaborar durant el procés de desenvolupament d'un programari. En general, però, els nous corrents dins de la programació podrien estar afavorint aquesta concepció entre la comunitat de desenvolupadors. De tota manera, aquest debat no ha sorgit arran de l'entrada en escena d'*Agile*, sinó que l'opinió respecte dels comentaris dins del codi informàtic sempre ha estat molt dividida entre la comunitat de desenvolupadors.

[3] Chereshnovska, Marta (2014). "Agile Localization: Myth or Reality?" [en línia]. Disponible a: <http://www.oneskyapp.com/blog/agile-localization-myth-reality/>

Un dels arguments en contra dels comentaris és que, si el codi està ben elaborat, si s'ha utilitzat una estructura clara, si els identificadors són útils i si existeix una correspondència lògica dels noms de les classes, les funcions i les variables, no és necessari comentar-lo. A més, hi ha mecanismes en programació que contribueixen a explicar el codi sense haver-lo de comentar, com ara els *unit tests*, que són proves que analitzen segments curts de codi per a trobar-hi possibles errors, o la 'refacció', que consisteix a reestructurar el codi sense modificar-ne la funcionalitat per a fer-lo més entenedor. Alguns professionals del sector opinen que aquests mecanismes, que s'acostumen a incloure dins del codi, donen lloc a l'anomenat '*self-documented code*', és a dir, codi que ja conté documentació. Així, doncs, es considera que ja aporten prou informació perquè un desenvolupador pugui realitzar el manteniment del codi d'un programari encara que no l'hagi escrit personalment.

Tanmateix, hi ha experts que, lluny de pensar que amb l'arribada d'*Agile* cal eradicar els comentaris del codi, opinen que els comentaris són importants, però no tant per a explicar què fa un segment del codi, sinó per a justificar la decisió que el desenvolupador ha pres en aquell punt. El perquè dels raonaments només es pot expressar mitjançant comentaris. A més, afirmen que els *units tests* i altres mecanismes semblants no només dificulten la comprensió ràpida del funcionament del codi, sinó que es limiten a destacar el que el codi no fa bé, sense evidenciar què li manca. En canvi, si un desenvolupador s'adona d'una mancança del codi, pot recollir la seva reflexió en un comentari.

En resum, podem pensar que els principis de la metodologia *Agile* van en detriment dels comentaris dins del codi. Si bé és cert que la nova tendència podria estar conduint els desenvolupadors en aquest sentit, en realitat, ens trobem davant d'un debat més antic que, tot i l'arribada d'*Agile*, sempre ha tingut aferrissats defensors i detractors.

3. ANÀLISI DE L'ENCÀRREC DE TRADUCCIÓ

3.1. Context de l'encàrrec de traducció

En aquest punt del projecte, ja hem comprovat el pes que té la *documentació* del codi per a alguns programadors i enginyers informàtics cèlebres i hem fet un repàs de l'estat actual de la qüestió. A banda d'aquests arguments, i per tal de justificar l'objectiu proposat en el treball, cal que considerem quin podria ser el context real i professional del nostre encàrrec de traducció simulat, en el qual una empresa ens demana la revisió, traducció i formatació dels comentaris del codi font d'un programari, aplicació o lloc web. Amb la combinació de tots els arguments entendrem el potencial que pot arribar a tenir un programari destinat a aquesta finalitat, del qual elaborarem el pseudocodi.

D'una banda, el pes innegable que han pres les noves tecnologies en les dues últimes dècades ha provocat que moltes empreses, grans i petites, i fins i tot les institucions, s'hagin hagut d'emmotllar al protagonisme absolut de la tecnologia en tots els àmbits de la vida. En aquest sentit, cal tenir en compte que no només podrien ser les empreses especialitzades en desenvolupament de programari les que requerissin la traducció dels comentaris del codi font d'un programari, sinó que el ventall de clients a qui podria interessar és força ampli. Per exemple, el govern d'un país, que desitja disposar de la traducció dels comentaris del codi font d'un programari informàtic d'ús governamental o per a la ciutadania en els idiomes oficials del país (podria ser el cas del català); o bé una empresa emergent de desenvolupament de programari, que treballa en col·laboració amb programadors d'altres països i necessita homogeneïtzar la *documentació* del codi pel que fa a la llengua i al format però no disposa de traductors a la plantilla i ha de subcontractar aquest servei; o bé un projecte sense ànim de lucre, en què un grup de programadors ha desenvolupat un *free and open-source software* (*FOSS*) o programari lliure de codi obert i es necessita traduir la *documentació* del codi a diversos idiomes perquè programadors d'altres països tinguin l'opció de llegir el codi font, comprendre'l i col·laborar en el projecte. No obstant això, la veritat és que podria ser un client potencial qualsevol empresa que encarregui l'elaboració d'un programari a una empresa de desenvolupament, atès que si posteriorment desitja modificar el programari que ha comprat, haurà d'encarregar aquesta tasca a una altra empresa que també pot necessitar la traducció dels comentaris del codi per a treballar-hi.

D'altra banda, ens trobem a l'era de la globalització, les conseqüències de la qual s'estenen a tots els àmbits, com ara les institucions i governs, la política, l'economia i també a l'àmbit professional (amb especial èmfasi dins del camp de la localització). En aquest escenari, no tindria cap sentit que el nostre pseudocodi es limités a una combinació d'idiomes concreta, atès que l'objectiu principal del projecte és que qualsevol traductor d'arreu del món pugui utilitzar-lo per a traduir la documentació del codi d'un programari perquè els desenvolupadors el comprenguin millor. Així, doncs, l'essència del projecte, el seu propòsit i el context actual són els que ens porten a fer d'aquest aspecte un requisit fonamental.

En resum, si sumem el pes que ha adquirit el camp de la localització dins de la traducció professional —tal com hem comentat a la introducció d'aquest treball— amb els nombrosos escenaris i necessitats de mercat que sorgeixen constantment arran del nou rumb que prenen moltes empreses i institucions, marcats per l'ús de la tecnologia, no és difícil d'imaginar la utilitat que pot arribar a tenir la traducció de comentaris dins de codi informàtic. Al cap i a la fi, no hem d'oblidar que s'emmarca dins d'un negoci, la localització, que mou grans xifres de diners cada any i que no ha fet sinó començar.

3.2. Descripció de l'encàrrec de traducció

Per a seguir el procés recomanable en qualsevol encàrrec de traducció, analitzarem el text d'origen (en endavant, *ST*, de l'anglès *Source Text*) amb què treballarem i també el text d'arribada (en endavant, *TT*, de l'anglès *Target Text*), és a dir, el text que volem aconseguir i que ens ha demanat el client. El *ST* sobre què treballarem és el codi font de l'aplicació web Jocomunico[4], que és el cas real que ha inspirat la idea d'aquest treball i el codi que prendrem de referència per a l'elaboració del pseudocodi. En aquest apartat, per tal d'il·lustrar les explicacions, utilitzarem fragments del codi font de l'aplicació web Jocomunico, escrits en llenguatge PHP. Tot i així, el codi font de Jocomunico combina els llenguatges PHP, HTML, JavaScript i CSS. Per aquest motiu, caldrà tenir en compte les variacions de format pel que fa a la introducció de comentaris en tots aquests llenguatges. Veurem aquest aspecte amb més detall a l'apartat 3.4.

[4] PAHISA-SOLÉ, Joan. (2016). *Jocomunico. L'app de CAA que parla amb naturalitat*. [lloc web]. Disponible a: <http://jocomunico.com/#/home> [Consulta: 23 de desembre del 2016]

Per a posar-nos en context breument, direm que Jocomunico és una aplicació gratuïta de Comunicació Augmentativa i Alternativa (CAA)[5] que s'emmarca en el camp de l'accessibilitat i que està pensada per a persones amb trastorns greus de la parla que es comuniquen amb pictogrames. Sovint aquests trastorns es deriven de paràlisis cerebrals (que en molts casos també impliquen deficiències motrius) i trastorns de l'espectre autista. L'aplicació expandeix de manera automàtica el llenguatge telegràfic, derivat de l'ús de pictogrames, a llenguatge natural en català i en castellà. Per exemple, un conjunt de pictogrames com ara "jo anar escola demà", es converteix en una frase natural: "Demà aniré a l'escola". (Pahisa-Solé, 2016). Per a fer-ho, després que l'usuari hagi introduït una sèrie de pictogrames (en el cas de la figura 1, els pictogrames introduïts són "jo", "agradar", "arròs" i "molt"), l'aplicació utilitza un motor intern de traducció basat en regles que converteix els símbols en elements d'una frase: els identifica (subjecte, verb, adjectiu, complement de lloc, complement de temps, etc.) i els ordena segons convingui en cada cas. D'aquesta manera, l'aplicació generarà la frase que veiem a la barra superior de la figura 1 i un sistema de síntesi de veu la llegirà en veu alta i es podrà sentir pels altaveus del dispositiu utilitzat: "L'arròs m'agrada molt". D'aquesta manera, s'aconsegueix donar veu a usuaris que tenen la parla afectada.

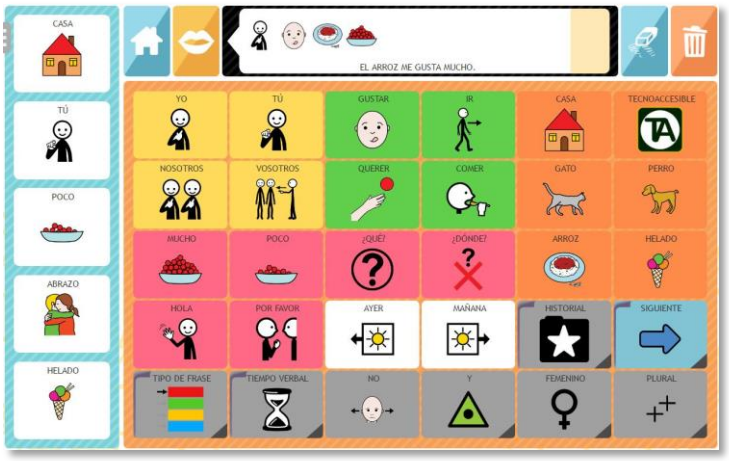


Fig. 1. Panell general de l'aplicació de CAA Jocomunico (Pahisa-Solé, 2016).

Tanmateix, les estratègies de CAA són un camp d'estudi molt ampli en què no entrarem, atès que ens allunyaríem del veritable propòsit d'aquest treball, que és l'automatització de processos per a la formatació d'un text i la traducció amb una eina TAO. En tot cas, Jocomunico és una aplicació gratuïta de codi obert que ha estat desenvolupada sense ànim de lucre de manera voluntària per un grup de programadors.

[5] Confederación ASPACE (2013). *Comunicación Aumentativa y Alternativa (CAA)* [en línea]. Disponible en: <http://aspacenet.aspace.org/main-menu/informacion-basica/item/153-comunicaci%C3%B3n-aumentativa-y-alternativa-caa>

Per les seves característiques i finalitats, és probable que en el futur sigui modificada i ampliada per desenvolupadors independents.

3.2.1. Descripció del *ST*

El nostre *ST* es troba dins de fitxers de codi informàtic. En aquests fitxers, hi trobem codi (instruccions, escrites en un llenguatge de programació concret, que executen accions) i comentaris escrits pels programadors (el contingut dels quals fa referència a les accions del codi o a les decisions preses pel programador). Així, el nostre *ST* no és pas el codi informàtic sinó els comentaris que l'acompanyen. Malauradament, aquests comentaris poden presentar una manca d'homogeneïtat lingüística i de format que dificulta poder-ne fer una traducció sistemàtica, coherent i professional, com és el nostre cas. Per a il·lustrar el nostre problema de traducció i veure'n exemples concrets, primerament veurem, a tall de mostra, dos fragments del codi font de l'aplicació. En els exemples, els comentaris dels programadors i, per tant, el nostre *ST*, són de color verd i van introduïts pel símbol de la doble barra (//). El primer fragment que veurem és una mostra de la manca d'homogeneïtat lingüística del *ST*:

```
.controller('historicCtrl', function ($scope, $rootScope, txtContent, $location, $http, dropdown
    // Comprobación del login   IMPORTANTE! HAY QUE PONERLO EN TODOS LOS CONTROLADORES
    if (!$rootScope.isLogged) {
        $rootScope.dropdownMenuBarValue = '/home'; //Dropdown bar button selected on this view
        $location.path('/home');
    }
    // Pedimos los textos para cargar la pagina
    txtContent("historicview").then(function (results) {
        $scope.content = results.data;
    });

    //Dropdown Menu Bar
    $rootScope.dropdownMenuBar = null;
    $rootScope.dropdownMenuBarValue = '/'; //Button selected on this view
    $rootScope.dropdownMenuBarButtonHide = true;
    $rootScope.dropdownMenuBarChangeLanguage = false;//Languages button available
    //SentenceBar button to open dropdown menu bar when hover
    $("#idSentenceBar").hover(function () {
        console.log('hover');
        $scope.dropdownMenuOpen = true;
```

Fig. 2. Exemple de documentació de codi amb manca d'homogeneïtat lingüística (Jocomunico, 2016).

A la figura 2 podem observar que el *ST* està escrit en més d'un idioma (en aquest exemple trobem el castellà i l'anglès), fet que representa un dels obstacles principals per a poder traduir el text amb una eina TAO, atès que aquest tipus d'eines només accepten una llengua de partida i una llengua d'arribada. A la figura 2, els comentaris són d'una

sola línia però, com veurem més endavant, a l'apartat 3.4, també podem trobar-ne de més d'una línia. A més, en aquest exemple observem que hi ha faltes ortogràfiques i, pel que fa al format, un ús abusiu de les lletres majúscules i l'omissió de signes de puntuació, com ara el punt final.

El segon fragment que veurem no presenta una manca d'homogeneïtat lingüística, atès que està escrit únicament en català, però, en canvi, és un exemple de manca de sistematicitat pel que fa al format:

```
}
// si no són complements, dependrà de quina categoria sigui l'slot
else {
    // si tenen un possessiu l'article és sempre determinat
    if ($auxstring[7] > 0) $definite = true;
    // si no hi ha possessiu
    else {
        // si és subjecte és determinat, excepte si és un lloc
        if ($this->category == "Subject") {
            // si és una ordre, el subjecte no porta article
            if ($tipusfrase == "ordre") $noarticle = true;
            else {
                if ($wordaux->isClass("lloc")) {
                    // mirar propietats
                    if ($wordaux->propietats->determinat == 'sense') $noarticle = true;
                    else $definite = true;
                }
                else $definite = true;
            }
```

Fig. 3. Exemple de documentació de codi amb manca d'homogeneïtat de format (Jocomunico, 2016).

En el cas de la figura 3, els comentaris han estat redactats de manera simple, esquemàtica, amb una finalitat purament pràctica on la prioritat és el missatge i no pas la forma. Tot i que això no és realment incorrecte, un dels nostres objectius com a revisors pot ser millorar la redacció de les frases esquemàtiques per a aportar intel·ligibilitat al *TT*. Pel que fa al format, també s'ha omès el punt final i la majúscula inicial de les frases. Per tot això, podem dir que el *ST* presenta una manca de sistematicitat lingüística i de format. Aprofundirem més en aquest aspecte als apartats 3.3 i 3.4, respectivament.

3.2.2. Descripció del *TT*

Ara que ens hem fet una idea de com és el text de què partim, descriurem quina és la voluntat del client d'aquest encàrrec de traducció simulat. D'una banda, el client ens ha demanat la revisió de la redacció, l'ortografia i el format del *ST*, sigui quin sigui el seu idioma. D'altra banda, vol que els comentaris estiguin redactats en català, castellà i anglès i que vagin introduïts per alguna mena de codi que identifiqui cadascun dels idiomes. Així, doncs, per a codificar els idiomes utilitzarem el sistema "idioma-PAÍS" que recomana Bergsten (2003) per a la identificació de *locales* seguint els estàndards internacionals ISO 639[6], de codi d'identificació d'idioma, i ISO 3166[7], de codi d'identificació de país:

> [The locale class] represents a particular geographical, political or cultural region, as specified by a combination of a language code and a country code. [...] The language code, a lowercase two-letter combination, is defined by the ISO 639 standard [...].The country code, a uppercase two-letter combination, is defined by the ISO 3166 standard [...]. (Bergsten, 2003: 235-236)

Aquesta combinació de tots dos codis és habitual dins de la programació informàtica. Així ho explica l'empresa de desevolupament de programari Oracle al seu lloc web[8]:

> A locale can be composed of a base language, country (territory) of use, and an optional codeset. Codeset is usually assumed. For example, German is de, an abbreviation for Deutsch, while Swiss German is de_CH, CH being an abbreviation for Confederation Helvetica. This convention allows for specific differences by country, such as currency unit notation. (Oracle, 2016)

[6] International Organization for Standardization (2016). *Language codes – ISO 639* [en línia]. Disponible a: <http://www.iso.org/iso/home/standards/language_codes.htm>
[7] International Organization for Standardization (2016). *Country codes – ISO 3166* [en línia]. Disponible a: <http://www.iso.org/iso/home/standards/country_codes.htm>
[8] Extret de: Oracle (2016). *Oracle. Integrated Cloud Aplications & Platform Services* [en línia]. Disponible a: <http://docs.oracle.com/cd/E19253-01/817-2521/overview-39/index.html>

Finalment, una vegada hem analitzat les característiques del *ST* i coneixem el resultat que vol obtenir el client, ja sabem cap a on ens dirigim. A continuació veurem el mateix fragment de codi que a la figura 3 —d'entre les dues figures, és la que presenta un major grau d'homogeneïtat— després d'haver estat revisat, traduït i editat seguint les indicacions del client. D'aquesta manera podrem comparar el *ST* amb el *TT* quant a contingut i aspecte, cosa que ens ajudarà a entendre completament d'on partim i què volem aconseguir. Una vegada més, en el fragment els comentaris apareixen de color verd introduïts pel símbol de la doble barra (//):

```
}
// en-US: If there is no complement, it will depend on the slot category.
// ca-ES: Si no són complements, dependrà de quina categoria sigui l'slot.
// es-ES: Si no son complementos, dependerá de la categoria del slot.
else {
    // en-US: If there is a possessive adjective, then we use no article (in Spanish).
    // ca-ES: Si tenen un possessiu, no porten article en castellà.
    // es-ES: Si tienen un posesivo, no llevan articulo en castellano.
    if ($auxstring[7] > 0) $noarticle = true;
    // en-US: If there is no possessive.
    // ca-ES: Si no hi ha possessiu.
    // es-ES: Si no hay posesivo.
    else {
        // en-US: If it is the subject, then it is definite, except for places.
        // ca-ES: Si és subjecte és determinat, excepte si és un lloc.
        // es-ES: Si es sujeto, es determinado, excepto si es un lugar.
        if ($this->category == "Subject") {
            // en-US: If the sentence is a command, no article will be attached to the subject.
            // ca-ES: Si és una ordre, el subjecte no porta article.
            // es-ES: Si es una orden, el sujeto no lleva artículo.
            if ($tipusfrase == "ordre") $noarticle = true;
            else {
                if ($wordaux->isClass("lloc")) {
                    // en-US: Features are checked.
                    // ca-ES: Comprovem les propietats.
                    // es-ES: Comprobamos las propiedades.
                    if ($wordaux->propietats->determinat == 'sense') $noarticle = true;
                    else $definite = true;
                }
```

Fig. 4. Exemple de documentació de codi revisada, traduïda i formatada (Jocomunico, 2016).

Així, doncs, aquest és el resultat que el client desitja obtenir per tal que desenvolupadors d'arreu del món puguin accedir al codi font, comprendre'l i modificar-lo si és necessari, per a afegir-hi funcionalitats o millorar les actuals. Si comparem les figures 3 i 4, pel que fa als aspectes lingüístics, observem que, en el cas de la figura 4, la documentació del codi disposa d'una redacció més extensa i completa que facilita la comprensió del missatge. Pel que fa al format, hem incorporat la codificació "idioma-PAÍS" abans de cada segment seguint l'exemple de Bergsten (2003). També hem corregit les majúscules i hem afegit el punt al final de les frases. Tanmateix, hem decidit ser coherents amb la terminologia tècnica (com ara el terme *slot*) perquè entenem que fa referència a elements del codi que porten el mateix nom i el

programador s'hi refereix en el seu comentari. Per tal d'evitar cap mena d'ambigüitat i mantenir la coherència entre el codi i de la seva documentació, hem pres la decisió d'adoptar els termes tècnics en anglès com a neologismes i fer-ne sempre un ús sistemàtic, atès que trobem que aquesta és l'opció més comprensible per al nostre lector final. El següent apartat aborda aquest aspecte amb més detall.

3.3. Anàlisi dels problemes lingüístics i terminològics del ST

A la figura 2, hem observat que la principal dificultat que presenta el *ST* és la barreja de més d'un idioma en un sol document. De fet, en alguns casos trobem més d'un idioma a la mateixa frase però en aquest cas el problema és terminològic. En altres paraules, al *ST* no tenim frases que estiguin escrites a mitges entre dos idiomes sinó que la terminologia tècnica és en anglès. Aquest fet té una explicació molt clara, que és la llengua anglesa com a idioma imperant en l'àmbit de la programació. El motiu d'aquest fet és que molts dels llenguatges de programació han estat creats a països de parla anglesa (Thompson, 2011), de manera que al llarg dels anys s'ha acabat establint, de manera inevitable, un únic idioma per al desenvolupament de programari. No obstant això, el conflicte sorgeix, normalment, en el cas dels desenvolupadors que no parlen la llengua anglesa. En aquest cas, dins del codi trobem que només les paraules clau han estat escrites en anglès però els comentaris o els noms de les variables i classes estan escrits en la llengua que parlen els programadors (Thompson, 2001):

> [...] we started by talking about programming by people whose language is not English. The keywords they use are, for almost all languages, in English. Comments, variables, user written classes and methods though are in their own language. How confusing might that be? (Thompson, 2001)

L'observació que fa Thompson reflecteix perfectament el cas del nostre projecte: comentaris escrits en català i en castellà, majoritàriament, que fan referència a elements del codi escrits en anglès. Així, doncs, com a revisors i traductors, cal que homogeneïtzem completament l'idioma del *ST* abans de començar a treballar-hi?

Davant d'un dilema d'aquest tipus, i deixant de banda el debat que pot existir arran de la imposició de l'anglès a la programació informàtica, cal que ens plantegem dues qüestions. La primera té a veure amb l'objectiu essencial del projecte, que en aquest cas

és facilitar la comunicació entre desenvolupadors de programari. Si homogeneïtzem l'idioma del *ST*, perjudicarem greument aquest principi, atès que, en el nostre cas, tots els termes en anglès dins de comentaris escrits en català o castellà fan referència a paraules clau o elements del codi informàtic. No ens trobem pas davant d'un desenvolupador bilingüe en català i anglès, o en castellà i anglès, que barreja tots dos idiomes intencionadament, sinó que ho fa per a prioritzar la funcionalitat dels comentaris.

La segona qüestió té a veure amb el temps que volem dedicar a la revisió del *ST*, especialment si, per a homogeneïtzar el text, la nostra intenció és buscar una traducció pels termes en anglès i afegir el terme original entre parèntesis a continuació per tal d'ajudar al desenvolupador a relacionar el terme traduït amb la paraula clau del codi. Per exemple, a la figura 3 veiem el terme en anglès *slot*: "// si són complements, dependrà de quina categoria sigui l'slot". Si considerem el terme *slot* com a incorrecte, a la fase de revisió hauríem de substituir-lo per un terme aproximat en català i afegir-hi el terme en la llengua d'origen entre parèntesis: "// Si són complements, dependrà de quina categoria sigui l'espai (slot)". El problema d'aquesta solució no només és que invertirem molt més temps del que havíem previst en la revisió, sinó que ens arrisquem a convertir els comentaris del codi en un text poc explicatiu en què hi ha un excés de terminologia entre parèntesis que dificulta la lectura i la identificació d'aquests termes dins del codi.

En conclusió, tenint en compte que el nostre *ST* inclou neologismes en anglès, tractarem aquest lèxic com a terminologia del client i el respectarem pel seu grau d'especialització (modificar-la podria fer el *TT* molt confús) i per la seva alta freqüència, és a dir, per la gran quantitat de terminologia tècnica en anglès que conté el *ST*. És molt important que, en establir un criteri de revisió o de traducció per a un text tan tècnic, siguem conseqüents amb la decisió, cosa que es relaciona, un cop més, amb la primera qüestió. Respecte d'aquest punt, com a traductors, ha calgut prendre una decisió i la respectarem al cent per cent al llarg de tot el text. En el nostre cas, s'ha decidit adoptar els termes en anglès que apareixen al text com a neologismes i fer-ne un ús completament sistemàtic. Per tant, cada vegada que trobem un terme en anglès dins del *ST* que faci referència a un element del codi que porta el mateix nom, el respectarem. A més, si en algun punt del text trobem que el mateix terme s'ha traduït al català o al castellà, el corregirem i el tornarem a escriure en anglès per a mantenir la coherència terminològica. Així, doncs, mantindrem la revisió de la frase d'exemple que ja hem pogut veure a la figura 3: "// ca-ES: Si són complements, dependrà de quina categoria sigui l'slot."

3.4. Anàlisi dels problemes de format del ST

En aquest apartat no només descriurem la manca d'homogeneïtat de format que presenta el *ST* i que cal que esmenem sinó que també enumerarem totes les variacions que hi trobem pel que fa als símbols d'introducció dels comentaris, és a dir, la seva sintaxi, en els diferents llenguatges de programació que utilitza el codi font de Jocomunico: PHP, HTML, JavaScript i CSS. Per a fer-ho, establirem una llista de criteris que ens servirà de guia a l'hora d'escriure el pseudocodi.

En primer lloc, als segments del *ST* els manca la majúscula al principi de frase i el punt final. Així, el primer criteri que establirem pel que fa a la revisió de format és l'esmena d'aquests dos errors.

En segon lloc, al *ST* trobem que els comentaris estan escrits en més d'un idioma dins d'un mateix document. Per a poder identificar-los, el segon criteri que establirem és la introducció del codi d'idioma i país (que s'explica a l'apartat 3.2.2) a principi de frase.

En tercer lloc, la distribució i posició física del text d'origen està marcat per tabulacions. En l'àmbit de la programació, és vital respectar el sagnat del codi (que en programació també es coneix com a *indentació*), atès que ajuda a fer-lo més llegible i manté l'estructura del contingut (Myatt, 2008: 36). Per aquest motiu, el tercer criteri que establirem té una gran importància si volem evitar desestructurar el *TT*. Així, caldrà respectar el sagnat del text original tot afegint la traducció de cada segment o grup de segments a la línia inferior al mateix nivell de tabulació. Aquest aspecte pot suposar un repte a l'hora d'elaborar el pseudocodi, atès que hem d'establir una llista de casos amb solucions que es puguin aplicar a la totalitat del *ST*.

Per últim, el quart criteri que establirem té a veure amb la sintaxi dels comentaris dins del codi, és a dir, els símbols que els introdueixen i que varien segons el llenguatge de programació (Rubin; Lloyd; Croft, 2007: 78). A continuació veurem quina és la sintaxi correcta pels comentaris en els quatre llenguatges de programació que conté el nostre *ST*. D'aquesta manera entendrem perquè alguns segments requereixen una nova formatació en aquest sentit.

- Llenguatge PHP (Davis, Phillips, 2007: 42) i JavaScript (Powers, 2008: 11):
  - Els comentaris d'una sola línia s'introdueixen amb la sintaxi "`//`", siguin dins o fora de la línia de codi. Per exemple:

```
$scope.languageList = [];
// Això és un comentari fora d'una línia de codi.
$scope.languageList = []; // Això és un comentari dins d'una
línia de codi.
```

  - Els comentaris de més d'una línia s'introdueixen amb la sintaxi "`/*`" i es tanquen amb "`*/`". La majoria d'aquests comentaris es fan fora del codi. Per exemple:

```
/*Això és un comentari
de més d'una línia.*/
```

O bé:

```
/*
Això és un comentari
de més d'una línia.
*/
```

- Llenguatge HTML (Irfanullah, 2014: 55):
  - Els comentaris d'una sola línia i de més d'una línia s'introdueixen amb la sintaxi "`<!--`" i es tanquen amb "`-->`". Per exemple:

```
<!--Això és un comentari d'una línia.-->
<!--Això és un comentari
de més d'una línia.-->
```

O bé:

```
<!--
Això és un comentari
de més d'una línia.
-->
```

- Llenguatge CSS (Rubin; Lloyd; Croft, 2007: 78):
  - Els comentaris d'una sola línia i de més d'una línia s'introdueixen amb la sintaxi "`/*`" i es tanquen amb "`*/`". La majoria d'aquests comentaris es fan fora del codi. Per exemple:

```
/*Això és un comentari
de més d'una línia.*/
```

O bé:

```
/*
Això és un comentari
de més d'una línia.
*/
```

Si ens basem en les descripcions de la sintaxi dels comentaris per a cada llenguatge que fan els diferents autors, al nostre *ST* els segments escrits en llenguatge HTML i CSS disposen de la sintaxi correcta. Els segments escrits en llenguatge PHP i JavaScript, en canvi, presenten algunes irregularitats. A continuació en podem veure dos exemples on, una vegada més, els comentaris dels desenvolupadors apareixen de color verd:

```
$scope.DenyOpenConfirmSize = function () {
    //reload the dropdown menus
    $scope.edit();
    $scope.showBoard('0');
};

//The user has clicked the cell. The cell can be:
//      SentenceFolder: The user will be redirected to the historic view
//      Sentence: The sentence will be readed
//      Picto: The picto is added to the sentece
//      Link to another board: the user will be redirected to this new board
//      Function: go to the historic, change the tipus or tme of the sentence...
// The last three can be together in the same cell
$scope.clickOnCell = function (cell) {

    if (!$scope.inEdit && cell.activeCell == 1) {
```

Fig. 5. Exemple de documentació de codi PHP amb manca d'homogeneïtat de format en comentaris de més d'una línia (Jocomunico, 2016).

```
            $preguntabona = true;
            $CI->session->set_userdata('preguntapattern', true);
            $this->errormessagetemp = null;
            $this->errorcodetemp = null;
            $this->errortemp = false;
            $this->readwithoutexpansion = false;
        }
    } // Fi tractament de pregunta

    // Si el verb és pseudoimpersonal o si hi ha una pregunta, invertim les preferències
    // d'aparèxier abans i després del verb, ja que ara el subjecte va darrere del verb
    // Les variables beforeverb només s'utilitzaran al codi si l'idioma té estructura SVO
    if ($auxpattern->pseudoimpersonal || $partpreguntaposada) {
        for ($j=0; $j<count($paraules); $j++) {
            $auxword = &$paraules[$j];
            $auxword->beforeverb = !$auxword->beforeverb;
```

Fig. 6. Exemple de documentació de codi JavaScript amb manca d'homogeneïtat de format en comentaris de més d'una línia (Jocomunico, 2016).

En els dos exemples que acabem de veure, trobem comentaris de més d'una línia introduïts incorrectament per "//". En tots dos casos, la solució correcta és l'ús de la sintaxi "/*" abans del text i "*/" després del text. Així, el quart criteri consisteix en l'aplicació de la sintaxi correcta al *TT* i, si és necessari, la correcció de la sintaxi del *ST* basant-nos en els tres models de sintaxi de comentaris que hem vist en aquest apartat. Tornarem a tractar tots els casos de sintaxi que trobem al *ST* a l'apartat 6.1.2,"Anàlisi de l'extensió i la codificació dels fitxers i extracció dels comentaris del codi".

4. AVALUACIÓ DE LES SOLUCIONS POSSIBLES

4.1. Ordre de prioritats

Abans de començar a valorar els avantatges i desavantatges dels mètodes que podrien solucionar més eficaçment els problemes del nostre encàrrec de traducció, és important que establim les característiques o condicions que ha de complir la solució escollida. Aquestes condicions seran com una mena de barem que ens servirà per a definir la funcionalitat de les opcions que explicarem als apartats següents i escollir la més escaient.

D'una banda, i com a requisit indispensable, volem que el mètode escollit inclogui, en algun punt del procés, una fase en què el text es pugui extreure del document d'origen per a, posteriorment, ser traduït mitjançant una eina de traducció assistida per ordinador o eina TAO (o *CAT*, per les seves sigles en anglès, tal com ho defineix Bowker, 2002). Utilitzar una eina TAO o *CAT* és indispensable al nostre projecte si volem dur a terme la traducció amb rigor professional. A més, ens ajudarà a treballar d'una manera eficient pel que fa a l'anàlisi dels fitxers que hem de traduir, la gestió i la coherència de la terminologia, l'estandardització d'un glossari que ens ajudi a ser precisos amb el lèxic que utilitzem, la reutilització de termes i segments gràcies a la memòria de traducció, la possibilitat de revisió, el control de qualitat i, per últim, la seguretat del projecte. Una altra de les característiques més pràctiques de les eines TAO és que respecten les etiquetes de format de l'arxiu original a l'hora de crear l'arxiu de destí. En el nostre cas, però, aquest no és un factor rellevant, atès que treballarem amb arxius d'extensió *CSV* (*Comma Separated Values*), els quals, en general, no emmagatzemen etiquetes de format. A tall de resum de les virtuts de les eines TAO, ens fixarem en un fragment de l'article "El papel de las herramientas TAO en la documentación técnica multilingüe" de Lidia Cámara (2001) a la *Revista Tradumàtica*:

> Las herramientas TAO ofrecen una optimización de la productividad en el proceso de traducción gracias a la automatización de los procesos repetitivos, lo que permite reducir considerablemente la velocidad del flujo de trabajo. Así mismo incrementan la productividad gracias al reciclaje (reutilización) de la información ya digitalizada obtenida mediante traducciones previas y por el reaprovechamiento de la estructura y el formato de los documentos originales, generados automáticamente en las versiones traducidas. Dependiendo del tipo de sistema TAO utilizado,

> puede optimizarse igualmente la calidad del producto final. Estos sistemas pueden integrar módulos y funciones cuya aplicación afecta directamente a la calidad de trabajo gracias a la mejora de la homogeneidad del estilo y la terminología en grandes volúmenes de documentos. (Cámara, 2001)

D'altra banda, atès que volem que el resultat d'aquest TFG es pugui aplicar a altres encàrrecs de traducció similars al que ens ocupa, és important que el mètode que escollim no estigui dissenyat exclusivament per a la combinació d'idiomes del nostre cas, sinó que, lingüísticament, volem que sigui transversal i aplicable a molts altres casos. Altrament, estaríem treballant en una solució feta a mida pel nostre encàrrec de traducció que no serviria de res en un altre encàrrec perquè, naturalment, tindria característiques diferents. Per tant, l'idioma d'origen i el de destí no seran un condicionant (és a dir, la nostra solució no valdrà només per a textos en català que calgui traduir al castellà i a l'anglès, per exemple) a l'hora d'utilitzar el pseudocodi que proposa aquest treball.

Així, doncs, per tal de trobar un mètode que s'adeqüi als criteris que hem establert fins ara i al nostre ordre de prioritats per a l'encàrrec de traducció, al proper apartat es proposen dues opcions possibles: d'una banda, la formatació i traducció directa del text i, d'altra banda, l'elaboració del pseudocodi. La primera opció, la qual s'ha posat en pràctica abans de l'elaboració d'aquest treball, consisteix a resoldre els problemes lingüístics (apartat 3.3) i de format (apartat 3.4) del text fent la traducció directament sobre una còpia del *ST* i tractant cada segment de manera individual. El punt fort d'aquesta primera opció és que el traductor pot solucionar per ell mateix els problemes del text de manera directa. Quan parlem de 'solucionar problemes de manera directa' ens referim a poder identificar els diferents idiomes que conté el *ST*, respectar el sagnat o corregir la sintaxi dels comentaris del *TT* sense haver d'emprendre la tasca d'elaborar un petit programari que ho faci per nosaltres. La segona opció, en canvi, consisteix a redactar, en un llenguatge no especialitzat, els passos que hauria de seguir un algorisme informàtic per a separar els comentaris del codi, respectar-ne o modificar-ne la sintaxi segons calgui i integrar els canvis de format i la traducció al *TT* de manera automàtica. Parlarem del mètode que seguirem per a traduir el text a l'apartat 5 del treball. A continuació, s'explicaran els avantatges i els inconvenients de les dues opcions amb més detall i s'escollirà la més adequada amb relació a l'ordre de prioritats.

4.2. Formatació manual i traducció directa del text

L'encàrrec de traducció simulat que descriu aquest TFG es basa en un de real, tal com s'explica a la introducció del treball. Per a parlar d'aquesta primera opció, és a dir, la formatació manual i la traducció directa del text, ens basarem en el procés que es va seguir en intentar aplicar aquesta opció. Després de rebre l'encàrrec, es va fer una pluja d'idees per a trobar la solució més adequada als problemes de format i traductològics del text. Aquestes idees, però, van quedar reduïdes principalment en dues.

Tenint en compte les característiques del *ST* pel que fa a la barreja d'idiomes, el sagnat dels comentaris dins del codi informàtic i la sintaxi que han de tenir els segments en funció del llenguatge de programació de què es tracti, la idea inicial (o més aviat el primer impuls) va ser formatar i traduir el text 'manualment'. A simple vista, trobar una solució que doni resposta a tots els interrogants i superi tots els obstacles que planteja la traducció és clarament una tasca enrevessada que requereix una investigació prèvia. Com que no es disposava de temps suficient, es va posar en pràctica aquesta primera idea. Amb el programa Notepad ++[9], es va començar a treballar directament sobre còpies dels fitxers que calia traduir, els quals tenien l'extensió .php, .html, .js i .css. Els avantatges i els inconvenients d'aquest mètode de seguida es van fer evidents.

Pel que fa als avantatges, tenen a veure principalment amb l'adequació del format, és a dir, el sagnat i la sintaxi dels comentaris en els diferents llenguatges de programació. En aquest sentit, el principal, i potser únic, punt a favor d'aquesta opció és que ens permet manipular el sagnat i la sintaxi lliurement i de manera específica per a cada segment.

En canvi, si parlem dels nombrosos desavantatges, la manca d'una memòria de traducció que asseguri la consistència lèxica és, possiblement, el més rellevant. En no tenir una memòria de traducció on s'emmagatzemin els fragments traduïts ni tampoc un glossari, cada vegada que es tradueix un segment cal comprovar si es repeteix al llarg del document. Si es repeteix, s'ha de substituir (amb el quadre de diàleg que apareix en prémer CTRL+F a Notepad ++ i a molts altres programaris) el text d'origen per la traducció tantes vegades com repeticions hi hagi del segment. Aquest mètode, que d'entrada pot semblar força senzill, requereix bastant de temps si treballem amb fitxers molt extensos. Al cas real en què es basa aquest TFG es treballava amb 11 fitxers que, en total, contenien al voltant de 20.000 línies de codi informàtic. Tanmateix, tenint en

[9] Don Ho, 2016. *Notepad ++ (GPL License)*. Disponible a: <https://notepad-plus-plus.org/>

compte que es tractava d'un projecte amateur, és fàcil d'imaginar el problema que suposaria fer servir aquesta tècnica en projectes més grans i professionals, amb una quantitat molt més elevada de segments per traduir.

La segona opció valorada, tot i que no va arribar a posar-se en pràctica amb èxit, va ser seleccionar el codi informàtic de dalt a baix i de text. A continuació, calia fer una macro que permetés seleccionar el text entre el símbol '//' i el final de la línia (o el següent punt que anés seguit d'un salt de línia) i copiar els resultats en un segon document de text. De fet, calia fer diverses macros per a corregir els diversos tipus de sintaxi dels comentaris que hi ha al text. D'aquesta manera, es podien aïllar tots els comentaris en un sol document i, així, realitzar la traducció d'una manera més controlada i còmoda. Per a integrar la traducció dels comentaris i les modificacions de la sintaxi dins del primer document de text, es podia fer servir una altra macro. Un cop fets aquests dos passos amb èxit, només calia copiar el text altra vegada al document original. Un dels problemes d'aquesta opció, però, és el perill que suposa executar línies de codi informàtic que han passat prèviament per un document de text, atès que aquest programari sovint afegeix etiquetes de format i altres elements que l'usuari no veu. Qualsevol símbol que hagués estat omès o afegit pel programari de processament de text podia perjudicar greument la funcionalitat del codi.

Si deixem de banda la pèrdua d'agilitat que suposa per al traductor el fet d'haver de cercar manualment les repeticions d'un segment, un altre dels inconvenients de tots dos mètodes és la manca d'una revisió exhaustiva de la traducció. Quan traduïm sense fer servir una eina TAO (que disposa de recompte de segments traduïts i per traduir, així com de control de qualitat o *QA* i un corrector ortogràfic), és probable que, en algun punt de la traducció, ometem un segment de manera involuntària o ens oblidem de picar un símbol, una lletra o un accent. Encara que un error de picatge que afecti una lletra o un accent és un error greu pel que fa a la correcció lingüística, quan traduïm comentaris dins de codi informàtic encara és més greu ometre un símbol, atès que, inevitablement, modificarem les ordres del codi i això tindrà conseqüències negatives a l'hora d'executar-lo. A les dues opcions que hem descrit, una vegada hàgim acabat de traduir el text, fer el control de qualitat de la traducció suposarà rellegir, un per un, tots els segments. En fer-ho, tornarem a arriscar-nos a ometre segments o errades.

Per acabar, si decidim formatar i traduir la totalitat dels segments sense fer servir una eina TAO, estarem descartant la possibilitat d'automatitzar algunes de les tasques del

procés, com ara la creació d'un glossari de termes freqüents. Això vol dir que, la pròxima vegada que rebem un encàrrec de traducció similar, haurem de repetir tot aquest procés com si fos el primer cop i, per tant, també tornarem a córrer els mateixos riscos.

4.3. Elaboració del pseudocodi d'un algorisme informàtic

En aquest apartat descriurem què és un algorisme, què és el pseudocodi i per a què serveix. A més, en veurem alguns exemples i proposarem una eina en la seva fase de disseny, una solució per al nostre problema tècnic de traducció. Per a dissenyar l'eina i parlar de les accions concretes que ha de realitzar, utilitzarem la sintaxi del llenguatge de programació Python™, per tal d'acostar l'explicació del disseny del programari al llenguatge tècnic que utilitzen els programadors.

Un algorisme és un "procediment de càlcul que consisteix a acomplir un seguit ordenat i finit d'instruccions amb unes dades especificades per tal d'arribar a la solució del problema plantejat." (TERMCAT, 2017). En el nostre cas, el problema plantejat és l'automatització de tasques i l'extracció i posterior reintroducció del text traduïble d'un codi informàtic. Pel que fa al pseudocodi, tal com ja hem vist a la introducció de treball, es tracta d'un "llenguatge informal, inspirat en l'estructura, sintaxi i vocabulari d'un llenguatge de programació, utilitzat per a descriure un algorisme o procediment sense entrar en els detalls específics de programació en un llenguatge concret." (TERMCAT, 2016). A més, podem escollir el grau de naturalitat o complexitat amb què escriurem el pseudocodi, que pot incloure més o menys elements de l'estructura, sintaxi i vocabulari del llenguatge de programació a què el vulguem traduir posteriorment.

El pseudocodi és molt útil si el que volem és comprovar que el nostre algorisme aconsegueix l'objectiu proposat abans de passar a la fase de programació. En fer-ho, és més probable que identifiquem errors i punts febles o dubtosos del codi. A més, una vegada el pseudocodi ha estat redactat i revisat, la traducció a un llenguatge de programació és més ràpida, atès que no cal que el programador reflexioni sobre quin camí ha de seguir el codi perquè aquest camí ja està dissenyat. Així, només s'haurà de concentrar a trobar la manera d'implementar el disseny en el llenguatge de programació escollit.

El pseudocodi pot allunyar-se en major o menor grau de la sintaxi del llenguatge per a què ha estat escrit. Per tal d'il·lustrar quin aspecte té el pseudocodi, a continuació

es mostren diversos exemples de pseudocodi amb diferent grau de proximitat a un llenguatge de programació. Els exemples 1 i 2 han estat extrets del llibre *Fundamentos de informática y programación en C* de l'autor Diego Rafael Llanos Ferraris (2010: 53). El tercer exemple ha estat extret del llibre *Problem Solving with Flowcharts and a Little Flavor of Programming with Python*, d'Agarwal et al. (2010: 12).

Exemple 1:

```
Leer primer número
Leer segundo número
Leer tercer número
Sumar los tres números
Escribir resultado
```

Exemple 2:

```
Leer dividendo
Leer divisor
si    dividendo es distinto de 0
      Calcular división
      Escribir resultado
si no Escribir No se permite la división por 0
```

Exemple 3:

```
Start
      Num1=5
      Num2=10
      Num3=15
      Sum=num1+num2+num3
      Average=sum/3.0
      Print average
End
```

Independentment del grau de complexitat amb què escollim elaborar el pseudocodi, és aconsellable que utilitzem alguns elements de la sintaxi del llenguatge en què s'escriurà el codi informàtic. D'aquesta manera, no només ens obligarem a pensar de manera similar a com ho fa un programador sinó que facilitarem la posterior traducció del pseudocodi al codi. Abans de veure quins són aquests elements de la sintaxi, però,

parlarem breument del codi de programació escollit per a posar en pràctica el pseudocodi que proposa aquest treball: el llenguatge Python™[10].

El llenguatge de programació Python™ s'utilitza en àmbits molt diversos, com ara el desenvolupament web i de programari, la programació en el camp científic, l'educació, etc. (Python Software Foundation, 2017). Aquest llenguatge de programació disposa d'una sintaxi més senzilla si la comparem, per exemple, amb C++ o amb Java. Per tant, el codi en Python™ és relativament més fàcil d'escriure i de llegir. També és més dinàmic i concís que aquests altres llenguatges, en el sentit que es necessiten poques línies de codi per a aconseguir executar una acció mitjanament complexa. Un altre dels seus avantatges és la transparència de la sintaxi, és a dir, el nom específic que reben els condicionals, les variables, els objectes, les classes, etc. Per últim, suporta nombrosos sistemes operatius i plataformes i, a més, és un llenguatge de programació de codi obert amb una gran comunitat de desenvolupadors al darrere.

Si parlem dels desavantatges de Python™, un dels més coneguts és la seva baixa velocitat de compilació, és a dir, el temps que triga a transformar el codi informàtic en el programa final. Tanmateix, aquesta no és una característica rellevant per al nostre projecte, atès que el codi informàtic del programari que es proposa dissenyar aquest treball serà relativament senzill i poc extens.

4.3.1. Sintaxi bàsica del llenguatge de programació Python™

A continuació veurem, de manera superficial, alguns dels elements bàsics de la sintaxi de Python™. No tindrem en compte la sintaxi completa d'aquest llenguatge tal com ho faria un programador professional sinó que únicament utilitzarem els elements més bàsics perquè el nostre pseudocodi tingui l'estructura adequada. Tota la informació que s'exposa a continuació ha estat extreta de la guia d'ús per a principiants *Learn Python Programming: The Definitive Guide* (Programiz, 2017):

- Tipus de dades (*Data Types*):
    - Nombres: poden ser enters (*int*), decimals (*float*) o complexos (*complex*).
    - Lletres: cadenes de caràcters (un *string*).

[10] Python Software Foundation (2017). *Python*™ [en línia]. Disponible a: <https://www.python.org/>

Aquestes dades es poden emmagatzemar en llistes (*list*) o bé emmagatzemar-les en la memòria del programa i assignar-los una variable (*var*) amb el símbol o operador "=".

- Operadors:
  - Suma (+), resta (-), multiplicació (*) i divisió (/).
  - Assignació (=), igual que (= =) i diferent de (!=).
  - Més gran que (>), més petit que (>).

- Condicionals:
  - *If... elif... else*: Si es compleix una condició, es realitzarà una acció posteriorment (*If*). Si la condició no es compleix, se'n poden proposar d'altres (*elif*). Si cap de les condicions anteriors no es compleix, se'n realitzarà una última (*else*). El condicional *if* funciona dins de la combinació *If... elif... else* però, si l'utilitzem tot sol, farà la seva funció igualment. En canvi, *elif* i *else* han d'anar precedits obligatòriament d'*if*.
  - *for* / *Loop*: Realitza una acció de manera repetida (*Loop*) tantes vegades com la condició *for* ho especifiqui. Serveix per a realitzar accions en bucle, com, per exemple, revisar una per una les línies d'un arxiu fins a trobar un *string* (cadena de caràcters) concret.
    - *break and continue:* interromp el *for* / *Loop.*

- Comparacions entre dades o entre variables (var), que representen dades emmagatzemades (s'utilitzen especialment en els condicionals):
  - Si *a* equival a *b* → if a = = b
  - Si *a* és diferent de *b* → if a !=b
  - Si *a* és més petit que *b* → if a>b
  - Si *a* és més gran que *b* → if a<b
  - Si *a* és més gran o igual a *b* → if a>=b
  - Si *a* més petit o igual a *b* → if a<=b

- Concatenació de condicionals (amb els operadors lògics o operadors booleans *and* i *or*):
    - Si *a* equival a *b* i *b* és més gran que *c* → if a= =b and b>c
    - Si *a* equival a *b* i *b* és més petit que *c* → if a= =b and b<c
    - Si *a* equival a *b* o *b* és més gran que *c* → if a= =b or b>c
    - Si *a* equival a *b* o *b* és més petit que *c* → if a= =b or b<c

- Introduir dades, ordenar-les, combinar-les, veure-les a la pantalla o extreure-les:
    - Cridar l'element o fitxer que volem introduir (*input*).
    - Crear una taula interna amb files i columnes on es poden emmagatzemar dades (*array*).
    - Ordenar elements (*sort*).
    - Unir dos o més elements en un de sol (*merge*).
    - Veure a la pantalla un resultat (*print*).
    - Guardar dades en una ubicació externa al programa (*save*).
        - Separar els valors d'aquestes dades amb el símbol que vulguem (*sep*).
    - Introducció de comentaris (#).
    - Salt de línia (\n).

5. AVALUACIÓ DE LES ESTRATÈGIES TECNOLÒGIQUES POSSIBLES

En aquest apartat s'exposaran les diferents estratègies tecnològiques de traducció que s'han tingut en compte a l'hora d'abordar aquest projecte. S'avaluaran els principals punts forts i febles de cada estratègia, així com tots els factors que hi intervenen, com ara el temps, els diners, la fluïdesa de la llengua, etc. D'aquesta manera, es pretén establir l'equilibri entre automatització o suport digital i esforç humà per tal d'aconseguir el millor resultat possible en les condicions més favorables. D'altra banda, si realment duguéssim a terme l'encàrrec aquest encàrrec de traducció simulat, abans de començar la traducció pròpiament dita, hauríem de realitzar diverses tasques, com ara el recompte de paraules, el recompte d'arxius, l'elaboració del pressupost, l'anàlisi terminològica, l'anàlisi del format i una planificació temporal amb relació al termini de lliurament del projecte, entre d'altres (Martín-Mor, Piqué, Sánchez-Gijón, 2016: 46). Per a començar a aproximar-nos a les estratègies tecnològiques, ens fixarem en l'esquema que plantegen Hutchins i Somers (1992: 148), atès que ens servirà per a il·lustrar els diversos camins que s'obren davant nostre quan ens disposem a fer una traducció:

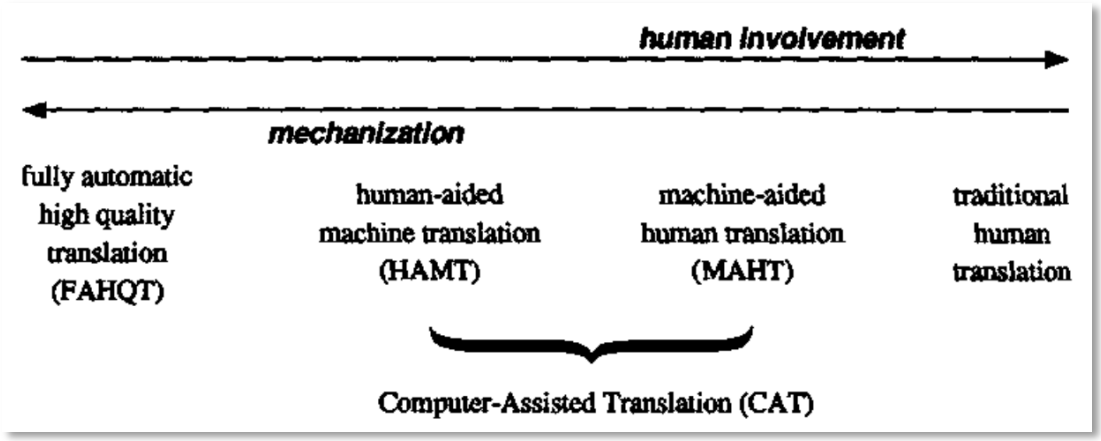


Fig. 7. *The practical use of MT Systems* (Hutchins, Somers, 1992: 148).

En aquest esquema s'exposa de manera clara la relació inversament proporcional que hi ha entre el grau d'intervenció del traductor humà, o *human involvement,* i el grau d'automatisme, o *mechanization,* de la traducció. Com més automàtic és el procés de traducció, menys parts del procés requereixen la implicació humana. Tanmateix, això també es tradueix en més o menys costos i qualitat, com veurem als apartats següents.

5.1.Traducció humana

Quan parlem de 'traducció humana', ens referim al concepte '*traditional human translation'* que Hutchins i Somers inclouen a l'extrem inferior dret del seu esquema. Això equivaldria al fet que el traductor traduís a mà tot el text amb un programari de processament de text com ara Microsoft Word[11]. A continuació, tornarem a veure l'esquema (Hutchins, Somers, 1992: 148), però és important destacar que **l'esquema original ha estat modificat** per tal d'indicar l'equivalència esmentada:

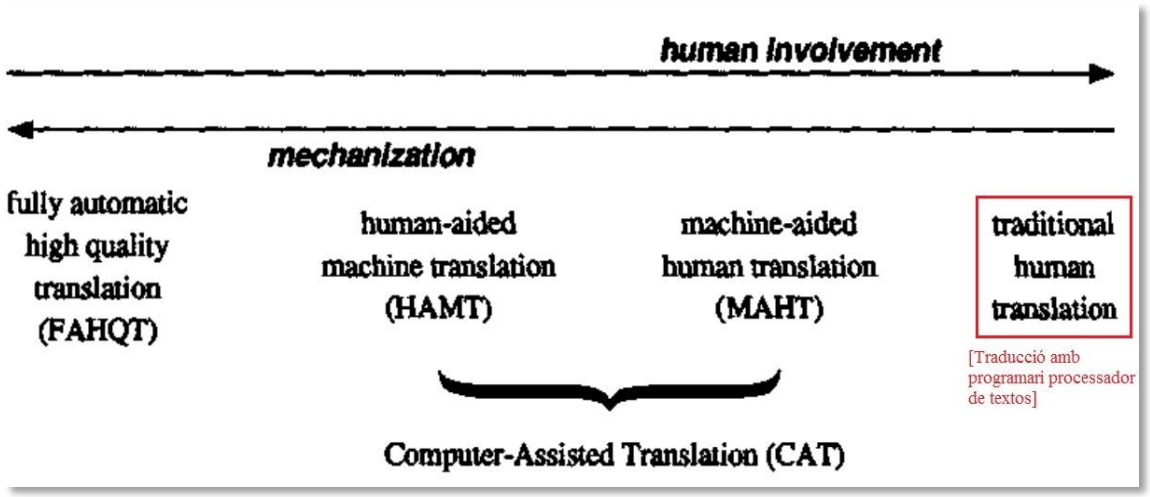


Fig. 8. *The practical use of MT Systems* (Hutchins, Somers, 1992: 148) [**ORIGINAL MODIFICAT**]. *Traditional human translation*: equivalència amb traducció humana.

Aquest mètode, d'una banda, tindria com a resultat una traducció d'alta qualitat, 100 % humana, escrita i revisada per un traductor. Ara bé, el cost de la traducció s'elevaria desmesuradament, la qual cosa descartaria tota rendibilitat. D'altra banda, aquesta estratègia compta amb nombrosos inconvenients, dels quals ja hem parlat a l'apartat 4.2, "Formatació manual i traducció directa del text". Per a recordar-ne els detalls, si us plau, vegeu la pàgina 35.

Els desavantatges innegables i de pes d'aquest mètode, que són fonamentalment els diners i el temps, però també el perill que comporta no utilitzar una eina TAO i una memòria de traducció que garanteixi la consistència terminològica dels segments, ens obliguen a descartar-lo directament.

[11] Microsoft (2017). *Microsoft Word* ©, 2016 Office.

5.2. Traducció automàtica amb postedició

Una vegada hem descartat la traducció humana, principalment per la seva baixa rendibilitat, ens fixarem en les opcions que tenim quant a la traducció automàtica combinada amb la postedició (en endavant, PE). Aquest mètode parteix d'una primera versió de la traducció generada per un sistema de Traducció Automàtica (TA) o *Machine Translation* (*MT*), pel seu nom en anglès. Aquest text "en brut", és posteriorment revisat per un traductor. Cal destacar que aquest procediment busca, per sobre de tot, la productivitat a partir de l'estalvi de temps i diners que suposa no haver de traduir manualment el text. Vegem com descriuen Martín-Mor, Piqué i Sánchez-Gijón l'ús de la TA:

> Quan s'opta per l'ús de la TA [...] el sistema tradueix el text original oració per oració, i a vegades fins i tot a partir d'elements inferiors a l'oració. El text resultant és, per tant, un conjunt de les millors opcions de traducció que ha trobat el sistema de TA, sigui basat en regles (TABR), estadístic (TAE) o híbrid [...]. El text que s'obté per mitjà d'un sistema de TA es coneix en anglès com a *raw translation*. (Martín-Mor, Piqué, Sánchez-Gijón, 2016: 62)

Aquesta estratègia, seguida de postedició, equivaldria a la '*Human-Aided Machine Translation*' de l'esquema de Hutchins i Somers (1992: 48), que es troba en un punt mitjà amb relació a la traducció tradicional manual i la traducció automàtica sense intervenció humana. Ara bé, si pensem en la qualitat de la traducció final, ens podem plantejar: quins són els límits de la PE? En quina mesura el traductor ha de modificar el text en brut que ha obtingut de la TA? Fins on ha d'arribar la intervenció humana en el text per tal que no invertim més temps del necessari i, de nou, el projecte esdevingui poc rendible?

Aquestes qüestions tenen a veure, principalment, amb el propòsit de la traducció, és a dir, l'ús a què es destinarà el text, el qual va directament lligat amb la qualitat que donarem a la traducció. En funció de la visibilitat i funcionalitat que tindrà la traducció, optarem per una revisió estricta del text que ha produït la TA o bé per una revisió més superficial on prioritzarem el contingut i no pas la forma. En altres paraules, tot dependrà de si necessitem un text lingüísticament acurat que ha de ser llegit per un públic nombrós (de manera que ha de semblar que hagi estat escrit per un traductor humà) o si, en canvi,

únicament volem que el lector rebi el missatge (per motius determinats, com ara perquè ens dirigim a un públic molt reduït o especialitzat) sense distorsions ni ambigüitats.

Alhora, hem d'afegir a la balança el temps que volem dedicar a la revisió i, en conseqüència el cost de la feina humana que hi invertirem. A més, cal recordar que la qualitat de la TA condicionarà, juntament amb la finalitat de la traducció, el temps que invertirem en la PE del text. En global, la suma de tots aquests factors determinarà la productivitat de la traducció i el mètode que més ens convé.

Les dues opcions de PE que hem descrit fins ara es coneixen en anglès, entre altres noms, com a *full post-editing* i *light post-editing:*

> La indústria de la traducció diferencia fonamentalment entre dos nivells de qualitat en funció de l'ús i de la visibilitat que tindrà el text final. D'una banda, trobem textos d'alta visibilitat, pensats per ser completament funcionals, que han de complir els mateixos requisits de precisió i de fluïdesa que qualsevol traducció humana. Aquest tipus de PE [postedició] es coneix com a *PE d'alta qualitat* (*full post-editing* o *high quality post-editing*), i es mesura amb els mateixos estàndards de visibilitat que la traducció humana. D'altra banda, també trobem textos d'una visibilitat més petita, o bé amb una funcionalitat limitada [...] que poden presentar alguna mancança en termes de fluïdesa. Aquest tipus de PE es coneix com a *PE de qualitat suficient* (*light post-editing* o *good enough post-editing*), i es mesura amb estàndards de qualitat més laxos que els d'una traducció humana en termes de fluïdesa però no en termes de precisió. (Martín-Mor, Piqué, Sánchez-Gijón, 2016: 69-70)

5.2.1. *Full post-editing*

Tal com hem vist fins ara, la PE d'alta qualitat o *full post-editing* té l'objectiu de convertir el text que ha generat la TA en un text que respongui als estàndards d'una traducció humana. Per tant, volem aconseguir un text que sigui acurat i comprensible i que reprodueixi amb un alt grau de fidelitat el significat del text original. Pel que fa a l'estil, mai serà tan correcte com si hagués estat escrit per un traductor nadiu, però ha de tenir un bon nivell. La puntuació ha de ser correcta, així com la gramàtica i la sintaxi. El lloc web de TAUS (*Translation Automation User Society*)[12] proporciona una guia de consells on s'especifiquen quines edicions hem de dur a terme com a revisors en funció del resultat que vulguem obtenir de la nostra PE. Aquestes són les indicacions que recull per a assolir una PE d'alta qualitat (TAUS, 2017):

[12] TAUS, 2017. *Translation Automation User Society* [en línia]. Disponible a: <https://www.taus.net/>

- Cal que corregim errors de gramàtica, sintaxi i semàntica.
- Cal que ens assegurem que la terminologia essencial ha estat traduïda de manera correcta i que els termes que no han estat traduïts són els que pertanyen a la llista de termes que el client ens ha demanat que no traduïm.
- Cal que ens assegurem que no s'ha omès ni afegit informació.
- Cal que editem tot aquell contingut que considerem ofensiu, inapropiat o inacceptable des d'un punt de vista cultural.
- Cal que fem servir la major part de la traducció en brut que puguem.
- Cal que corregim errors d'ortografia, ortotipografia i puntuació.
- Si és necessari, cal que ajustem el format del text.

Així, doncs, la PE d'alta qualitat o *full post-editing* busca la qualitat humana, amb un grau òptim de comprensió, fluïdesa i correcció lingüística del text en llengua d'arribada.

5.2.2. *Light post-editing*

Si la traducció amb què treballem, com dèiem, és un text destinat a tenir poca visibilitat o funcionalitat, sovint és preferible invertir-hi menys temps i recursos (malgrat que el resultat no sigui perfecte), que no pas invertir-hi tant o més temps que si haguéssim traduït el text manualment. Ara bé, mentre que establir els límits de la PE d'alta qualitat no sembla gaire complex (només cal que comparem la traducció amb un text escrit per un nadiu per a saber si ens acostem a la mateixa qualitat o no), això no resulta tan senzill quan es tracta de PE de qualitat suficient *(light post-editing o good enough post-editing*). Amb aquest tipus de PE, només es busca corregir errors de fluïdesa, si dificulten o impossibiliten la comprensió del missatge, i tots els errors d'adequació que hi pugui haver. D'igual manera, la correcció de la gramàtica i la sintaxi passa a un segon pla, sempre que no es posi en perill la intel·ligibilitat del text. També cal corregir les ambigüitats, però no corregirem els errors que no afectin el missatge, encara que siguin d'allò més evidents. Per exemple, si en el text generat pel sistema de TA trobéssim l'oració "La menú d'opcions serveix per a...", no invertiríem temps a corregir la manca de concordança de gènere entre l'article 'la' i el substantiu 'menú', atès que no afecta el sentit i transmet el missatge sense problemes.

Una vegada més, ens fixarem en les indicacions que recull el lloc web de TAUS (2017) per a assolir una PE de qualitat suficient:

- Cal que corregim la traducció des del punt de vista semàntic.
- Cal que ens assegurem que no s'ha omès ni afegit informació.
- Cal que editem tot aquell contingut que considerem ofensiu, inapropiat o inacceptable des d'un punt de vista cultural.
- Cal que fem servir la major part de la traducció en brut que puguem.
- Cal que corregim errors d'ortografia.
- No farem cap correcció de caràcter estilístic.
- No reformularem les oracions per a guanyar fluïdesa.

Per tant, la PE de qualitat suficient o *good enough post-editing* busca la precisió quant al contingut però no pas la correcció gramatical, sintàctica ni estilística. Tampoc no prioritza la fluïdesa del text sinó que només pretén suavitzar els errors de traducció que s'interposin amb la comprensió del missatge.

En qualsevol cas, existeixen altres recomanacions que es poden aplicar a qualsevol mena de PE pel que fa a errors de coherència terminològica i lèxica (que determinaran en gran mesura la qualitat de la traducció final), gramaticals, d'estil i ortotipogràfics, així com aspectes relacionats amb el producte on apareixerà la traducció final. D'entrada, segons Martín-Mor, Piqué i Sánchez-Gijón, "existeix una màxima en PE segons la qual només s'ha d'editar allò que sigui imprescindible canviar" (2016: 71). A més, aquests autors indiquen que cal parar atenció als casos següents (Martín-Mor, Piqué, Sánchez-Gijón, 2016: 73-76):

- Ús apropiat de la terminologia fixada pel client o pròpia del projecte.
- Tractament dels noms propis o comercials, que caldrà traduir o no en funció de l'estratègia comercial del client i la visibilitat del text.
- Revisió de les abreviatures i acrònims.
- Llista d'equivalents recomanats, especialment en programaris, per tal d'aconseguir la màxima homogeneïtat i formalitat i no caure en fórmules inadequades o ambigües en la llengua d'arribada.
- Llista de falsos amics que cal revisar.
- Solucionar errors de concordança en la mesura que considerem necessària.

- Procurar que el text sigui formal i tingui un estil impersonal (si això s'adequa a la finalitat de la traducció).
- Tractament de les frases subordinades complexes.
- Llista de traduccions literals, especialment quant a les estructures sintàctiques de les oracions.
- Traducció adequada dels títols d'apartats.
- Equivalència dels signes de puntuació entre la llengua d'origen i la d'arribada.
- Equivalència correcta entre xifres (decimals, milers, milions, etc.)
- Ús de majúscules als títols i a les llistes.
- Aspectes de localització regional com ara la data, la moneda, les unitats de mesura, etc.
- Tractament d'elements reemplaçables que poden donar lloc a errors de concordança de gènere i nombre.

5.3. Fully Automatic Unattended Machine Translation

Per últim, parlarem del que es coneix com a *fully automatic unattended machine translation.* Consisteix a traduir el text amb un sistema de TA i eliminar la fase de revisió posterior o postedició, característica que fa que rebi el nom d'*unattended translation* o traducció *desatesa*, on no intervé mai el traductor humà. En aquesta estratègia de traducció, la *raw translation* de què parlàvem a l'apartat 5.2 es considera, doncs, la traducció final. Aquest mètode s'utilitza fonamentalment amb textos que són consultats amb poca freqüència (és a dir, tenen poca visibilitat) o amb una finalitat molt específica, com ara manuals d'ajuda de programaris i llocs web o missatges de correu electrònic generats automàticament que no admeten resposta (Muzii, 2016: 20).

Per exemple, quan tenim un dubte respecte del funcionament d'un programari, una de les opcions que tenim per a trobar la resposta al nostre dubte (a banda de cercar la solució directament a Google) és el manual d'ajuda del programari, que sovint es troba en línia. És molt habitual que, en aquest tipus de manuals, al final de cada entrada trobem la pregunta: "T'ha semblat útil aquesta resposta?", acompanyada de dos botons que ens deixen escollir entre el 'sí' i el 'no'. En aquests casos, és molt probable que el text sigui una traducció automàtica *desatesa*. En cas que la resposta 'no' tingui un gran nombre de

votacions o clics, el text serà revisat per una persona però, en canvi, les respostes que no rebin un nombre significatiu de queixes no es revisaran mai perquè, tenint en compte la visibilitat i funcionalitat del text, no és rendible.

En definitiva, aquesta estratègia de traducció s'utilitza amb textos que només pretenen transmetre el missatge essencial, la informació rellevant i res més, ignorant per complet la fluïdesa i la correcció lingüístiques.

Una vegada ja hem vist algunes de les opcions principals entre les quals hem d'escollir a l'hora d'abordar el projecte de traducció que planteja aquest TFG, tornarem a veure l'esquema de Hutchins i Somers (1992: 148) tot indicant l'equivalència aproximada de les categories que estableix l'autor respecte de les estratègies tecnològiques que hem explicat en aquest apartat. Novament, és important destacar que **l'esquema original ha estat modificat** per tal d'indicar les equivalències esmentades, així com el factor econòmic (que depèn immediatament del temps que invertim en la traducció) i el factor qualitatiu:

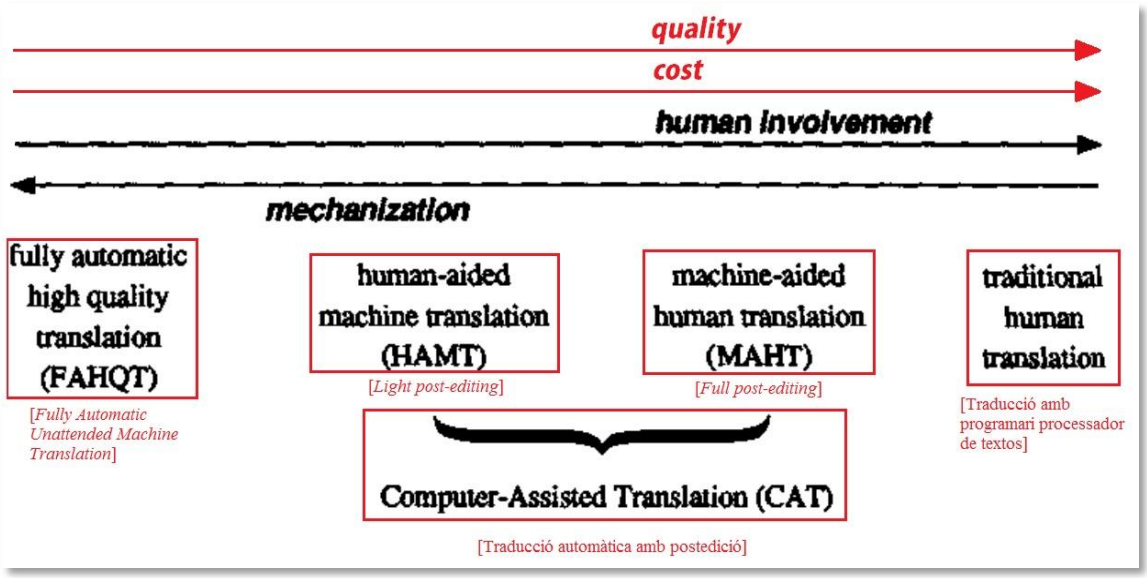


Fig. 9. *The practical use of MT Systems* (Hutchins, Somers, 1992: 148) [**ORIGINAL MODIFICAT**]. *Traditional human translation*: equivalències, factor preu i factor qualitat.

Tal com hem explicat a l'apartat 5.1, la traducció humana queda descartada pel seu cost elevat i l'alta probabilitat d'inconsistència terminològica. L'estratègia *fully automatic unattended machine translation* també la descartarem, atès que no podem oblidar que l'objectiu principal de la traducció de comentaris de text dins de codi informàtic és contribuir que enginyers d'arreu del món es puguin comunicar d'una manera àgil i puguin resoldre qualsevol dubte sobre fragments de codi que han estat escrits per altres programadors. Tenint això en compte, no volem que una revisió nul·la del resultat de la TA dificulti l'assoliment d'aquest requisit. Les característiques complexes del nostre *ST* ens indueixen a pensar que la traducció automàtica *desatesa* podria perjudicar el missatge fins al punt de fer-lo incomprensible. Així, doncs, les opcions entre les quals hem de decidir es redueixen a dues: *full post-editing* o *light post-eding*, atès que són les més escaients amb relació a la visibilitat i funcionalitat que tindrà el *TT*.

Per a decidir-ho, però, és necessari avaluar els resultats que ofereix la traducció automàtica en català, castellà i anglès. A l'apartat 6.1.5, "Fase de preedició del text", s'ofereix una mostra del resultat de la TA de 5 segments del *ST* en català, castellà i anglès abans i després de l'aplicació de criteris de preedició de textos per a TA. Amb aquests mateixos resultats, a l'apartat 6.1.6, "Fase de traducció automàtica i postedició del text", es decidirà quina és l'estratègia idònia per a aquest encàrrec de traducció.

6. MARC EMPÍRIC: REALITZACIÓ DE L'ENCÀRREC DE TRADUCCIÓ

6.1. Pseudocodi de l'algorisme informàtic

En el marc empíric d'aquest TFG, dissenyarem l'algorisme informàtic del programari. Per a fer-ho, expressarem les instruccions en format de pseudocodi. El nostre objectiu és descriure, de la manera més tècnica i entenedora possible, el producte que volem obtenir. Així, hauríem de poder entregar el pseudocodi final a un programador professional, que l'entengués i el programés. Per a la redacció del pseudocodi, ens basarem en la sintaxi del llenguatge de programació Python™, d'acord amb els motius que s'exposen a l'apartat 4.3, "Elaboració del pseudocodi d'un algorisme informàtic". Així, doncs, al final d'aquest apartat, haurem dissenyat tots els passos que ha de seguir un programari que ens permeti extreure els comentaris del codi informàtic i emmagatzemar-los en fitxers externs que puguem preeditar i introduir en una eina TAO. Una vegada traduïts i posteditats, volem tornar-los a incorporar dins del *ST*, és a dir, els fitxers de codi, sense perjudicar-ne el sagnat i respectant la posició on es trobava el comentari original. A més, afegirem davant dels comentaris una etiqueta amb el format "idioma-PAÍS: " de quatre caràcters separats per un guionet (-) (vegeu apartat 3.2.2, pàg. 26). Per tant, com a resultat, allà on abans hi havia un comentari en un idioma, hi haurà el mateix comentari en un altre idioma o en més d'un (l'usuari ho podrà escollir) correctament identificat amb l'etiqueta d'idioma corresponent. A més, es corregirà el format dels comentaris antics i dels nous, sempre que calgui, quant a la sintaxi d'introducció de comentaris, la majúscula inicial i el punt final.

Per tal de no carregar l'explicació del procés amb captures de pantalles del pseudocodi que ocuparien, en la majoria de casos, més d'una plana, vegeu el pseudocodi complet del programari a l'annex 9.8 (pàg. 91). Així, doncs, cada subapartat del punt 6 contindrà la descripció del procediment lògic que segueix el programari acompanyada de figures que per a il·lustrar-la.

A continuació, farem un breu resum de la concatenació d'accions que ha d'executar el programari que dissenyarem. D'aquesta manera, entendrem millor tot el procés i les fases que descriurem en els propers apartats:

1. **Obtenció de l'idioma i la variant geogràfica dels comentaris.**

Demanarem a l'usuari de quin idioma/es tradueix i cap a quin idioma/es per tal d'obtenir l'etiqueta d'idioma que precedirà els comentaris al *TT*.

2. **Anàlisi de l'extensió i la codificació dels fitxers i extracció dels comentaris del codi.**

L'usuari introduirà els fitxers de treball i en comprovarem l'extensió i codificació. D'aquesta manera, sabrem quina sintaxi d'introducció de comentaris hem de buscar a cada fitxer per a extreure el text que hi hagi a continuació (és a dir, els comentaris, el nostre *ST*).

3. **API de reconeixement d'idioma.**

En cas que el *ST* contingui comentaris en més d'un idioma, enviarem els comentaris que hem extret a una interfície de programació d'aplicacions o API de reconeixement d'idioma que ens retornarà l'idioma de cada comentari.

4. **Obtenció dels fitxers en format CSV.**

Una vegada sapiguem l'idioma de cada comentari, extraurem tots els comentaris en fitxers CSV separats per idiomes. En cas que només hi hagi un idioma al *ST*, no haurem de separar els fitxers CSV sinó que extraurem un únic fitxer de treball.

5. **Fase de preedició del text.**

En aquesta fase del procés, fora del programari, preeditarem el *ST* seguint una sèrie de directrius específiques per a textos que han de ser traduïts amb TA. D'aquesta manera, invertirem una mica de temps extra en la preparació del text però optimitzarem els resultats de la TA i estalviarem temps a la postedició.

6. **Fase de traducció automàtica i postedició del text.**

En aquest punt continuarem treballant fora del programari. Després de traduir el text amb TA, escollirem quina estratègia tecnològica ens interessa més pel que fa a la postedició (amb relació a les característiques del *ST*): *light postediting* o *full postediting*.

7. **Substitució del *ST* pel *TT*.**

Una vegada hàgim traduït tots els fitxers de treball, introduirem els segments traduïts (*TT*) al programari. Finalment, el programari substituirà el *ST* pel *TT*, tenint en compte la sintaxi adequada de cada comentari i la posició que ocupaven originalment.

6.1.1. Obtenció de l'idioma i la variant geogràfica dels comentaris

Per començar, mostrem a l'usuari dues llistes: una basada en l'estàndard internacional ISO 639, de codi d'identificació d'idioma, i una altra basada en l'estàndard internacional ISO 3166, de codi d'identificació de país. A continuació podem veure un fragment de totes dues llistes (els annexos 9.3 i 9.4 contenen les llistes completes):

ISO-639 Language Codes

Language Name	Language Code
Abkhazian	ab
Afar	aa
Afrikaans	af
Albanian	sq
Amharic	am
Arabic	ar
Armenian	hy
Assamese	as
Aymara	ay
Azerbaijani	az
Bashkir	ba
Basque	eu
Bengali (Bangla)	bn
Bhutani	dz
Bihari	bh
Bislama	bi
Breton	br
Bulgarian	bg
Burmese	my
Byelorussian (Belarusian)	be
Cambodian	km
Catalan	ca

ISO-3166 Country Codes

Country Name	Country Code
AFGHANISTAN	AF
ALBANIA	AL
ALGERIA	DZ
AMERICAN SAMOA	AS
ANDORRA	AD
ANGOLA	AO
ANTARCTICA	AQ
ANTIGUA AND BARBUDA	AG
ARGENTINA	AR
ARMENIA	AM
ARUBA	AW
AUSTRALIA	AU
AUSTRIA	AT
AZERBAIJAN	AZ
BAHAMAS	BS
BAHRAIN	BH
BANGLADESH	BD
BARBADOS	BB
BELARUS	BY
BELGIUM	BE
BELIZE	BZ
BENIN	BJ

Fig. 10. Fragment de les llistes ISO 639 i 3166 mostrades a l'usuari.

Després, li demanem que, mirant la llista d'idiomes, indiqui quin és l'idioma d'origen del seu *ST* amb el codi de dos caràcters en minúscules. Tot seguit, li demanem que introdueixi la variant geogràfica de l'idioma del seu *ST* amb el codi de dos caràcters en majúscules. Tot seguit, li preguntem si hi ha més idiomes d'origen al seu *ST*. Si la resposta és afirmativa, com és el nostre cas, cal que torni a indicar el codi de l'idioma i de la variant geogràfica segons apareixen a la llista. Repetim la pregunta fins que la resposta sigui negativa. Fem exactament la mateixa operació per a la llengua o llengües de destí.

Un cop l'usuari ha introduït tota aquesta informació, emmagatzemem les dades en una taula amb tres columnes: una per al nom de l'idioma (que obtenim després de cercar a la llista ISO 639 el codi de dos caràcters d'idioma que ens ha facilitat l'usuari), una altra per al codi de dos caràcters en minúscula de l'idioma que ha introduït l'usuari, i una altra

per al codi de dos caràcters en majúscula de la variant geogràfica de l'idioma, que també ha introduït l'usuari. Utilitzarem les dades d'aquesta taula més endavant per a crear l'etiqueta d'idioma amb el format idioma-PAÍS, que hem descrit a l'apartat 3.2.2. Vegeu el fragment de pseudocodi que descriu el pas 6.1.1 a l'annex 9.8 (pàg. 91).

6.1.2. Anàlisi de l'extensió i la codificació dels fitxers i extracció dels comentaris del codi

Ara, demanem a l'usuari que introdueixi l'adreça URL on es troba el *ST*, és a dir, el fitxer de codi informàtic que conté els comentaris en la llengua d'origen. Una vegada tenim l'URL del *ST*, necessitem detectar-ne l'extensió per a saber quins caràcters (o sintaxi) hem de cercar per a extreure els comentaris del codi. Definim dos tipus de sintaxi d'obertura i de tancament de comentaris (vegeu l'apartat 3.4): la de tipus A (per a arxius .php, .html, .js i .css) i la de tipus B (per a arxius .php i .js). Més endavant, a la fase final 6.1.7, "Substitució del *ST* pel *TT*", tornarem a seguir aquest mateix procediment per a saber quina sintaxi cal aplicar per a introduir altra vegada els comentaris traduïts dins del codi.

En aquest punt, a més, extraiem els comentaris del codi —és a dir, el text que es troba entre la sintaxi d'introducció i de tancament de comentaris. Per a fer-ho, hem de donar al programari un conjunt de condicions que equivalen al tipus de sintaxi que trobem als fitxers .php, .html, .js i .css, la qual ve determinada pel llenguatge de programació en què han estat escrits, tal com hem vist a l'apartat 3.4 d'aquest treball. Per tant, el programari ha de cercar aquests símbols, extreure el text que hi ha a continuació i emmagatzemar-lo en una taula interna o *array* que contingui tres columnes: una per a l'identificador del comentari, una per al text del comentari i una per a l'idioma. No obstant això, no omplirem la columna destinada a l'etiqueta d'idioma fins al pas 6.1.3, i només ho farem en cas que hi hagi més d'un idioma al ST i el reconeixement d'idioma de l'API sigui necessari. Si no és així, aquesta columna quedarà buida fins al final.

A l'hora d'emmagatzemar cada comentari del *ST*, hi afegim, després de la sintaxi d'introducció, un identificador únic. Aquest identificador és fonamental si tenim en compte que el número de línia dels comentaris (és a dir, la seva posició) variarà a mesura que anem substituint el text dels comentaris originals per la seva preedició o traducció, atès que els segments corregits o traduïts poden ser més extensos que els originals. Per

aquest motiu, no podem prendre de referència el número de línia on es trobaven els comentaris dins del codi inicialment, sinó que els hem de donar un identificador únic (per cada comentari, independentment de l'idioma) que no variï mai i que ens ajudi a mantenir la posició original del comentari de cara a la seva substitució en el *TT*. Per tal de fer servir una combinació de símbols que no es pugui confondre de cap manera amb el contingut del comentari, hem donat a l'identificador el format "$$"+número+"$$", on el número incrementa per 1 a cada comentari, és a dir: "$$1$$", "$$2$$", "$$3$$", "$$4$$", etc.

En acabar, preguntarem a l'usuari si té més *STs* per traduir. Si la resposta és afirmativa, repetirem tot el procés, fins que sigui negativa. Vegeu el fragment de pseudocodi que descriu el pas 6.1.2 a l'annex 9.8 (pàg. 92).

6.1.3. API de reconeixement d'idioma

A continuació, el programari demana a l'usuari si el seu *ST* conté comentaris escrits en més d'un idioma. Si la resposta és negativa, s'exporta automàticament, en format CSV, la taula o *array* que hem descrit a l'apartat anterior i que conté tres columnes: una per a l'identificador del comentari, una per al text del comentari i una per a l'idioma, malgrat que aquesta última columna està buida perquè, en haver-hi un sol idioma d'origen, no cal diferenciar i marcar els comentaris amb una etiqueta d'idioma. El fitxer CSV resultant serà, doncs, *ST* per traduir (és a dir, els comentaris sense el codi), preparat per a ser introduït en una eina TAO.

Ara bé, si la resposta és afirmativa, entrarem en el pas més delicat del procés que segueix el programari. Tal com hem vist al llarg del treball, els comentaris del codi informàtic (especialment si ha estat elaborat per programadors amateurs o de parla no anglesa) poden presentar una falta d'homogeneïtat pel que fa a l'idioma, com en el cas real en què es basa aquest treball. Així, doncs, necessitem detectar l'idioma dels comentaris amb l'ajuda d'una interfície de programació d'aplicacions o API. Un dels motius que fan que aquest sigui un pas complex és el factor econòmic, atès que la majoria d'API de reconeixement d'idioma, com ara *Google Translation API*[13] o *Microsoft Translator Text API*[14], són de pagament a partir de cert nombre de peticions. El segon

[13] Google Cloud Platform (2017). *Google Translation API* [en línia]. Disponible a: <https://cloud.google.com/translate/docs/detecting-language>

[14] Microsoft (2017). Microsoft Translator Text API [en línia]. Disponible a: <https://www.microsoft.com/en-us/translator/translatorapi.aspx>

motiu és que no podem afirmar que siguin 100 % infal·libles a l'hora de reconèixer els idiomes, atès que, malgrat que la probabilitat d'encert dels resultats acostuma a ser elevada, hi ha un petit marge d'error en el cas dels idiomes minoritaris.

A les proves dutes a terme per a l'elaboració d'aquest treball, s'ha utilitzat *Language Detection API* de LanguageLayer[15], un lloc web que ofereix una API que permet realitzar fins a 5.000 peticions de manera gratuïta, amb un grau considerable d'encert en català. Cal tenir en compte, però, que el nombre de peticions realitzades per a aquest treball és molt reduït si el comparem amb els milers de peticions que podríem arribar a fer per a traduir tots els comentaris del codi de qualsevol programari professional.

A tall de demostració, veurem quina informació ens retorna l'API de LanguageLayer després d'introduir-hi una frase en català:

- Indica si la petició ha estat resposta amb èxit o no.
- L'idioma escollit expressat en un codi de dos caràcters.
- El nom complet de l'idioma.
- La probabilitat d'encert.
- El percentatge de seguretat o *confidence* de l'idioma detectat.
- Indica si es considera que la resposta és fiable o no.


```
{
   "success": true,
   "results": [
        {
          "language_code": "ca",
          "language_name": "Catalan",
          "probability": 38.50637972414,
          "percentage": 100,
          "reliable_result": true,
        },
    ]
}
```


Fig. 11. Retorn de *Language Detection API* de LanguageLayer.

Com dèiem, aquest és un pas delicat en el procés que, en certa mesura, complica la programació del pseudocodi. No obstant això, seguirem exposant el disseny del nostre programari.

Una vegada hem fet les peticions a l'API i n'hem obtingut els resultats, ens tornem a centrar en la taula o *array* descrita anteriorment, la qual conté tres columnes: una per a l'identificador del comentari, una altra per al text del comentari, i una altra per a l'etiqueta d'idioma en el format idioma-PAÍS. Per a obtenir l'etiqueta d'idioma, comparem el nom de l'idioma que ens ha retornat l'API amb el nom de l'idioma que hem emmagatzemat a

[15] Apilayer (2017). *LanguageLayer* [en línia]. Disponible a: <https://languagelayer.com/>

la taula descrita al final de l'apartat 6.1.1. Una vegada hàgim identificat de quin idioma es tracta dins la taula, prendrem el codi de dos caràcters en minúscula de l'idioma i el de dos caràcters en majúscula de la variant geogràfica que ha introduït l'usuari per a aquell idioma i els unirem amb un guionet (-).

Per tant, obtenim una taula d'un aspecte similar al de la figura 12. En endavant, els colors de les figures seran purament il·lustratius, per tal que sigui més fàcil identificar els idiomes a primera vista. A la figura, l'idioma català està marcat amb el color vermell, l'idioma castellà amb el verd i l'idioma anglès amb el blau.

	A	B	C
1	$$1$$	Hola món.	ca-ES
2	$$2$$	Este es el segundo comentario.	es-ES
3	$$3$$	Aquest és el tercer comentari.	ca-ES
4	$$4$$	This is the fourth comment.	en-US
5	$$5$$	This is the fifth comment.	en-US
6	$$6$$	Este es el sexto comentario.	es-ES
7	$$7$$	Aquest és el setè comentari.	ca-ES
8	$$8$$	Este es el octavo comentario.	es-ES
9	$$9$$	This is the ninth comment.	en-US
10	$$10$$	Aquest és el desè comentari.	ca-ES

Fig. 12. Exemple il·lustratiu de taula amb identificadors, *ST* i etiquetes d'idioma.

A continuació, exportem la taula en format CSV, un format que no emmagatzema etiquetes de format, com ara la negreta, el subratllat, la cursiva o el color de la casella, entre d'altres. Escollim aquest format de fitxers per a no córrer el risc d'introduir etiquetes invisibles no desitjades al codi quan tornem a incorporar-hi els comentaris, una vegada hagin estat traduïts. Tal com veurem a l'apartat següent, l'objectiu de l'exportació de la taula (fig. 12) en aquest punt del procés és donar a l'usuari l'oportunitat de revisar i, si cal, corregir la identificació d'idioma que ha produït l'API. Vegeu el fragment de pseudocodi que descriu el pas 6.1.3 a l'annex 9.8 (pàg. 93).

6.1.4. Obtenció dels fitxers de treball en format CSV

En aquesta part del procés, primer demanem a l'usuari si desitja tornar a importar el fitxer que hem generat amb els resultats de l'API. D'aquesta manera, si l'API ha detectat erròniament l'idioma d'algun dels comentaris o segments, l'usuari té l'oportunitat de modificar l'etiqueta de l'idioma i tornar a importar tots els comentaris amb l'etiqueta correcta dins del programari. De fet, aquest pas podria formar part de la fase de preedició, atès que es tracta d'un pas posterior a la introducció dels documents en un programari, però anterior a la traducció. En cas que totes les deteccions hagin estat correctes, l'usuari simplement haurà d'introduir l'adreça URL on es troba emmagatzemat el fitxer que tot just acaba de generar el programari.

A continuació, el nostre objectiu és filtrar i dividir el fitxer CSV de la figura 12 i obtenir-ne tants fitxers independents com idiomes d'origen diferents s'hagin detectat. En el cas que es presenta en aquest TFG, obtindrem 3 fitxers diferents que seran els nostres *STs*: un que contingui els segments en català, un altre amb els segments en castellà i un altre amb els segments en anglès.

Una vegada hem realitzat aquesta operació, ja tenim els fitxers CSV que finalment preeditarem, traduirem amb una eina TAO i posteditarem. Podem imaginar-nos els segments que tenim i que ens manquen en cada idioma com una si fossin un trencaclosques.

	A	B	C
1	$$1$$	Hola món.	ca-ES
2	$$3$$	Aquest és el tercer comentari.	ca-ES
3	$$7$$	Aquest és el setè comentari.	ca-ES
4	$$10$$	Aquest és el desè comentari.	ca-ES

	A	B	C
1	$$4$$	This is the fourth comment.	en-US
2	$$5$$	This is the fifth comment.	en-US
3	$$9$$	This is the ninth comment.	en-US

	A	B	C
1	$$2$$	Este es el segundo comentario	es-ES
2	$$6$$	Este es el sexto comentario	es-ES
3	$$8$$	Este es el octavo comentario.	es-ES

Fig. 13. Separació del *ST* per idiomes en fitxers independents.

Per exemple, en el cas dels segments en català, a continuació marcarem els segments que tenim de color vermell i els que no tenim els deixarem de color negre. Farem el mateix en castellà i en anglès amb els seus colors respectius:

ca-ES:	**1**	2	**3**	4	5	6	**7**	8	9	**10**
es-ES:	1	**2**	3	4	5	**6**	7	**8**	9	10
en-US:	1	2	3	**4**	**5**	6	7	8	**9**	10

El que volem és traduir els segments que tenim en català al castellà i a l'anglès, traduir els que tenim en castellà al català i a l'anglès i traduir els que tenim en anglès al català i al castellà. D'aquesta manera els acabarem tenint tots en els tres idiomes. Cal recordar que només hem de realitzar aquesta part del procediment en cas que tinguem més d'un idioma d'entrada o de sortida (és a dir *STs* o *TTs* multilingües). Vegeu el fragment de pseudocodi que descriu el pas 6.1.4 a l'annex 9.8 (pàg. 93).

6.1.5. Fase de preedició del text

A la fase de preedició i a la de traducció i postedició del text, treballarem fora del programari que estem dissenyant i no editarem el pseudocodi ni hi afegirem informació nova. Com que hem establert que la traducció dels segments es farà amb TA + PE, una vegada ja tenim els fitxers CSV en idiomes diferents, o un únic fitxer en un sol idioma, és molt recomanable, si no imprescindible, preeditar el text seguint una sèrie de directrius específiques per a textos que han de ser traduïts amb TA. D'aquesta manera, invertirem una mica de temps extra en la preparació del text però optimitzarem els resultats de la TA i, en conseqüència, estalviarem temps a la fase de postedició. En reduir el nombre d'errors predictibles que genera habitualment la TA, n'augmentarem l'eficiència. Vegem alguns dels consells que recull el blog de KantanMT (2013) i el lloc web de l'empresa de localització Sajan (2014), en un ordre de més a menys rellevància:

1) Correcció ortogràfica: accents, lletres canviades d'ordre, etc. Per a fer la revisió ortogràfica, podem copiar el contingut del document CSV en un corrector en línia

o bé en un programari de processament de text per tal de poder utilitzar el seu corrector ortogràfic.

2) Revisió de possibles errors de codificació, com ara accents o majúscules no reconegudes.
3) Simplificació de les oracions que tinguin una estructura complexa.
4) Segmentació amb punts d'oracions que siguin excessivament llargues.
5) Introducció de comes i signes de puntuació allà on manquin.
6) Substitució de termes amb més d'un significat.
7) Substitució de les abreviatures per substantius complets.
8) Transformació del llenguatge col·loquial en un de més formal.
9) Substitució de frases fetes o expressions per sintagmes clars i senzills.
10) Reutilització de segments per a missatges equivalents.
11) Introducció d'articles i conjuncions allà on manquin.
12) Transformació de les oracions passives en oracions actives.
13) Substitució dels pronoms per noms.
14) Substitució de dobles espais per espais únics.

En aquesta fase, també hem de valorar si necessitem introduir termes prohibits a l'eina TAO que utilitzarem. Per exemple, si no volem que el verb en anglès "*get*" es tradueixi en cap cas per "aconseguir", en català, o "*conseguir*", en castellà, sinó que preferim que el tradueixi d'alguna altra manera, com ara "obtenir", prohibirem aquesta traducció per tal de no haver-la de corregir a la postedició cada vegada que aparegui. Pel que fa al cas en què es basa aquest treball, i amb relació al raonament que s'exposa a l'apartat 3.3, "Anàlisi dels problemes lingüístics i terminològics del *ST*", sobre el tractament de bona part del lèxic en anglès del text com a terminologia especialitzada del client, també podem introduir termes en el glossari perquè la traducció (o la no-traducció) sigui sistemàtica. En el nostre cas, per exemple, volem que sempre tradueixi el nom en anglès "*print*" per "printar" en català i en castellà, encara que no sigui correcte normativament, perquè considerem que s'adequa al llenguatge que utilitzen els lectors a qui està adreçat el text i que és part de la terminologia del client. A més, volem que deixi sense traduir els termes en anglès "*slot*", "*string*", "*board*" o "*modal*" pel mateix motiu.

Per tal d'aportar una prova dels efectes de la preedició de textos per a TA, als annexos 9.5, 9.6 i 9.7 (vegeu les pàgines 88, 89 i 90) s'ofereix una comparació del resultat de la TA de 5 segments del *ST* en català, castellà i anglès abans i després d'haver-los

preeditat. En aquestes tres proves podem observar que ometre la preedició del text posa en perill la transmissió del missatge, atès que la TA genera frases estranyes que són difícils de comprendre per a algú que no ha vist mai el *ST*. Alhora, això implica una postedició més extensa. No obstant això, els resultats de la TA amb preedició també contenen bastants errors que necessiten ser revisats per a un traductor humà que pugui comprar la TA amb el *ST*. Aquesta prova confirma que, per al nostre encàrrec de traducció, no podem prescindir de la postedició i que, malgrat que el nostre *TT* tindrà poca visibilitat, haver descartat la *fully automatic unattended machine translation* ha estat una decisió encertada.

Tornant als passos que segueix el nostre programari, una vegada hàgim preeditat el *ST* en tots els idiomes que tinguem, guardarem una còpia de cadascun d'ells per a diferenciar-los dels fitxers que no han estat preeditats. A partir d'aquest moment, treballarem sobre les còpies i descartarem els fitxers originals que contenen segments que no han passat per la fase de preedició.

6.1.6. Fase de traducció automàtica i postedició del text

En aquest apartat, continuem treballant de manera externa al pseudocodi del programari. Ha arribat el moment d'introduir el *ST* en una eina TAO per a traduir-lo amb TA i després posteditar-lo.

L'elecció de l'eina TAO l'ha de fer el traductor basant-se en el seu criteri personal, experiència i pressupost. Per al cas que ens ocupa, s'han comparat dues eines TAO que disposen d'editors web que permeten treballar al núvol: Memsource Cloud[16], la versió al núvol de Memsource, i Lilt[17]. Malgrat que totes dues eines són d'allò més intuïtives, completes i pràctiques, s'ha descartat l'eina Lilt perquè no inclou el català entre les opcions de llengua de partida ni d'arribada. A més, la versió de prova de Memsource Cloud és gratuïta de manera indefinida, encara que limita a 2 el nombre de projectes que es poden tenir actius simultàniament, mentre que Lilt ofereix una versió de prova de 21 dies però després esdevé de pagament.

Una vegada hem escollit l'eina TAO amb què treballarem, crearem un projecte nou per al català i habilitarem l'opció de 'pretraducció', que genera un text traduït amb

[16] Memsource (2017). *Memsource Cloud* [en línia]. Disponible a: <https://cloud.memsource.com/>
[17] Lilt Inc. (2017). *Lilt* [en línia]. Disponible a: <https://lilt.com/>

un sistema de traducció automàtica o TA ('Microsoft with Feedback' en el cas de Memsource Cloud). El sistema de TA que utilitzem dependrà de l'eina TAO que utilitzem i dels motors de TA que ofereixi, o bé dels nostres coneixements per a integrar una API externa de TA dins l'eina TAO. La preedició que hàgim fet del text i l'API que escollim per a la TA, determinaran la qualitat de la traducció que posteditarem i, per tant, el temps que dedicarem a aquesta fase.

Pel que fa a la postedició, basant-nos en els resultats obtinguts a la comparació del resultat de la TA de 5 segments del *ST* en català, castellà i anglès abans i després d'haver-hi aplicat criteris de preedició (vegeu els annexos 9.5, 9.6 i 9.7), sembla més escaient aplicar una postedició *light* (vegeu l'apartat 5.2.2, "*Light post-editing*"). D'aquesta manera, el temps invertit en la preedició quedarà compensat per una revisió de la TA més ràpida i superficial. En el nostre cas, doncs, la prioritat de la postedició serà transmetre el missatge però obviarem els errors de fluïdesa sempre que no impedeixin la comprensió. Hem de recordar que els fitxers de codi informàtic acostumen a ser molt extensos, per la qual cosa tampoc no ens interessa que una postedició exhaustiva (*full post-editing*) allargui el termini de lliurament de l'encàrrec de traducció en excés i, novament, el projecte perdi rendibilitat.

6.1.7. Substitució del *ST* pel *TT*

Després d'haver traduït automàticament el fitxer en llengua d'origen a la llengua de destí i haver posteditat la TA, és hora de tornar-nos a centrar en el pseudocodi per a introduir els segments traduïts al codi informàtic, amb la sintaxi adequada segons l'extensió del fitxer i en la posició original.

En el nostre cas, com que partim de tres idiomes d'origen i volem arribar a tres idiomes de destí, hem d'introduir un total de 9 fitxers CSV: l'original en català més la traducció al castellà i l'anglès; l'original en castellà més la traducció al català i l'anglès; per últim, l'original en l'anglès més la traducció al català i al castellà. A la figura 14, podem veure petits fragments del que trobaríem a cadascun d'aquests 9 fitxers CSV. La columna B mostra els textos originals que hem extret del codi mentre que les columnes F

i J en mostren les traduccions. Novament, els colors serveixen només per a fer els exemples més explicatius:

	A	B	C	D	E	F	G	H	I	J	K
1	$$1$$	Hola món.	ca-ES		$$1$$	Hola mundo.	es-ES		$$1$$	Hello world.	en-US
2	$$3$$	Aquest és el tercer comentari.	ca-ES		$$3$$	Este es el tercer comentario.	es-ES		$$3$$	This is the third comment.	en-US
3	$$7$$	Aquest és el setè comentari.	ca-ES		$$7$$	Este es el séptimo comentario.	es-ES		$$7$$	This is the seventh comment.	en-US
4	$$10$$	Aquest és el desè comentari.	ca-ES		$$10$$	Este es el décimo comentario.	es-ES		$$10$$	This is the tenth comment.	en-US
5											
6											
7	$$2$$	Este es el segundo comentario.	es-ES		$$2$$	Aquest és el segon comentari.	ca-ES		$$2$$	This is the second comment.	en-US
8	$$6$$	Este es el sexto comentario.	es-ES		$$6$$	Aquest és el sisè comentari.	ca-ES		$$6$$	This is the sixth comment.	en-US
9	$$8$$	Este es el octavo comentario.	es-ES		$$8$$	Aquest és el vuitè comentari.	ca-ES		$$8$$	This is the eighth comment.	en-US
10							ca-ES				
11											
12	$$4$$	This is the fourth comment.	en-US		$$4$$	Aquest és el quart comentari.	ca-ES		$$4$$	Este es el cuarto comentario.	es-ES
13	$$5$$	This is the fifth comment.	en-US		$$5$$	Aquest és el cinquè comentari.	ca-ES		$$5$$	Este es el quinto comentario.	es-ES
14	$$9$$	This is the ninth comment.	en-US		$$9$$	Aquest és el novè comentari.	ca-ES		$$9$$	Este es el noveno comentario.	es-ES

Fig. 14. Resultat de la fase de traducció dels *STs* cap als *TTs*.

Així, doncs, en primer lloc, demanem a l'usuari que introdueixi tots els fitxers que tingui com a *TTs*, que pot ser el document que ha traduït a partir del fitxer que hem generat a l'apartat 6.1.3 i que només conté una llengua de destí, o bé els documents que hem generat a l'apartat 6.1.4 per a dividir els segments en fitxers segons el seu idioma. Si l'usuari només introdueix comentaris en un idioma, el programari entén que l'usuari desitja substituir els comentaris originals pels traduïts (o pels corregits o posteditats, si l'idioma del *ST* i del *TT* és el mateix). Així, si al *ST* hi ha un únic idioma, els comentaris en el *TT* no aniran introduïts per l'etiqueta d'idioma, atès que, tal com hem vist a l'apartat 6.1.3, no haurà estat necessari identificar-los i diferenciar-los (la columna destinada a l'etiqueta d'idioma ha quedat buida). En canvi, si l'usuari introdueix *TTs* que contenen els mateixos comentaris en idiomes diferents, el programari entén que l'usuari desitja afegir els comentaris en tots els idiomes nous sota els comentaris en l'idioma d'origen. Podem saber si els *TTs* que ha introduït l'usuari contenen els mateixos comentaris en diferents idiomes gràcies a l'identificador únic de comentari que els hem donat, atès que és el mateix per a tots els idiomes.

Dels fitxers que ens ha proporcionat l'usuari, agafem tots els comentaris en tots els idiomes i els emmagatzemem en una taula o *array* amb tres columnes: una per a l'identificador, una altra per al text del comentari, i una altra per a l'etiqueta d'idioma. Tot seguit, ordenem aquesta taula alfabèticament per idiomes amb l'objectiu de tenir, en els fitxers de codi finals, els comentaris traduïts sempre en el mateix ordre (per exemple, primer els comentaris en català, després els comentaris en castellà i després els comentaris en anglès).

A continuació, entrem a l'última fase del procés. Primer tornem a comprovar l'extensió del *ST* o *STs* que l'usuari ha introduït a l'inici, per a saber quina sintaxi d'obertura i tancament de comentaris hem d'aplicar als segments a l'hora d'introduir-los al codi, i també la seva codificació, per a saber quina codificació de caràcters hem de fer servir.

Per acabar, recorrem totes les línies de cada *ST* i, allà on hi hagi un comentari, ens aturem i comprovem quin identificador té. Tot seguit, busquem el mateix identificador dins de la taula que hem creat amb el contingut de tots els fitxers amb els comentaris traduïts que ens ha proporcionat l'usuari. Quan trobem l'identificador a la taula, seleccionem i copiem el comentari que té associat i sobreescrivim el comentari original del *ST*, tot esborrant el text del comentari, però també l'identificador que li havíem donat ("$$+número+$$") i la sintaxi d'introducció de comentaris original que tenia el segment. El nou comentari anirà precedit de la sintaxi de comentari que li pertoqui, segons l'extensió del fitxer on es trobi, i de l'etiqueta "idioma-PAÍS: ". A continuació, seguirem recorrent la taula, atès que si hem traduït els comentaris a més d'un idioma, trobarem altres comentaris que tindran el mateix identificador. Quan els trobem, afegirem cada comentari traduït just a sota de l'anterior, fent servir la mateixa sintaxi i l'etiqueta d'idioma que li correspongui. En acabar de recórrer la taula, seguirem recorrent el *ST* i, quan trobem el següent comentari, repetirem tot el procediment.

Una vegada tots els comentaris antics hagin estat substituïts pels nous, generarem el fitxer final, és a dir, el *TT* (el qual tindrà la mateixa extensió que el *ST)*. Després, tornarem a fer la mateixa operació des del principi amb la resta de *STs* que ens hagi proporcionat l'usuari.

Vegeu el fragment de pseudocodi que descriu el pas 6.1.7 a l'annex 9.8 (pàg. 94).

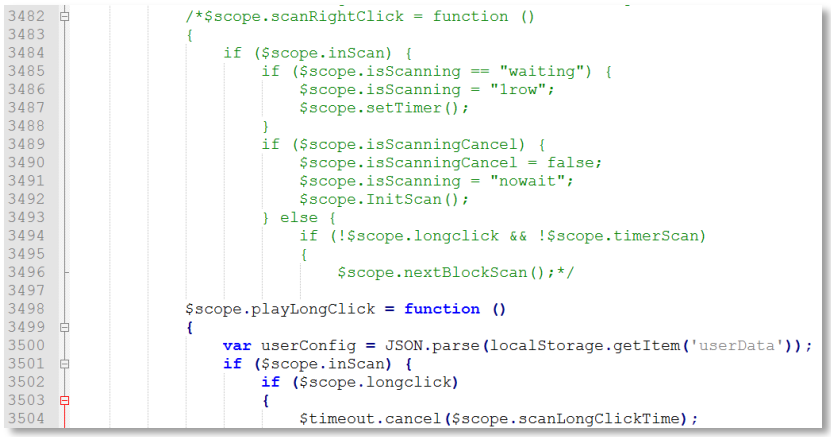
6.2. Ampliacions i millores de l'algorisme

Malgrat que l'algorisme que hem explicat a l'apartat 6.1 intenta cobrir totes les possibilitats i donar solució a diversos casos, hi ha certs punts febles o parts que caldria millorar dins del plantejament d'algunes parts del procés. Els annexos 9.1 i 9.2 contenen una mostra de 100 línies, a tall d'exemple, del *ST* i el *TT*, respectivament, per tal de comparar el text abans i després d'haver passat pel procés que porta a terme el nostre programari.

Algunes de les millores de què parlarem són detalls específics del funcionament del programari que no li impedeixen assolir el seu objectiu, però que caldria perfeccionar. Per exemple, podem parlar de l'elecció de l'ordre en què apareixeran els comentaris en el cas dels comentaris multilingües. Ara per ara, l'ordre és alfabètic i l'usuari no pot escollir si vol que l'ordre de l'idioma dels comentaris sigui un o un altre. Un altre detall que caldria millorar és l'aparició de missatges d'error en cas que l'usuari hagi modificat els noms dels fitxers *ST* que introdueix a l'inici del procés. És important que els noms dels fitxers no pateixin cap modificació, atès que és la referència que té el programa per a saber on ha de col·locar els comentaris traduïts. Per últim, seria interessant que, en un pas posterior a l'exportació de l'arxiu traduït final (és a dir, el *TT*) el programari alineés, en una taula, els comentaris originals amb els traduïts, mitjançant l'identificador que els afegim al principi. Després, es podria exportar aquesta taula perquè l'usuari tingués l'oportunitat d'introduir-la en una eina TAO i crear una memòria de traducció amb els parells de segments. En el futur, podria reutilitzar les traduccions dels comentaris i afegir-ne més a la memòria de traducció.

Ara bé, si parlem d'aspectes més generals, que afecten la lògica que segueix el programari, cal que parlem d'una pràctica molt habitual entre els programadors que es coneix com a 'codi comentat'. Sovint els programadors utilitzen la sintaxi d'introducció de comentaris no per a escriure un comentari realment sinó només per a inhabilitar una part del codi. Recordem que tot el que hi ha entre la sintaxi d'obertura i tancament de comentaris no afecta les operacions del codi. D'aquesta manera, els programadors poden mantenir un fragment de codi inactiu sense haver-lo d'esborrar, per a poder-lo recuperar si el tornen a necessitar més endavant. A la figura 15, podem veure que, en escriure la sintaxi d'obertura de comentaris "/*" (línia 3.482) i la de tancament "*/" (línia 3.496), el fragment de codi ha quedat inhabilitat i ha pres el color verd dels comentaris en aquest

fitxer. Igualment, pel color podem veure que les línies de codi que hi ha a continuació sí que es mantenen actives:



```
/*$scope.scanRightClick = function ()
{
    if ($scope.inScan) {
        if ($scope.isScanning == "waiting") {
            $scope.isScanning = "1row";
            $scope.setTimer();
        }
        if ($scope.isScanningCancel) {
            $scope.isScanningCancel = false;
            $scope.isScanning = "nowait";
            $scope.InitScan();
        } else {
            if (!$scope.longclick && !$scope.timerScan)
            {
                $scope.nextBlockScan();*/

$scope.playLongClick = function ()
{
    var userConfig = JSON.parse(localStorage.getItem('userData'));
    if ($scope.inScan) {
        if ($scope.longclick)
        {
            $timeout.cancel($scope.scanLongClickTime);
```

Fig. 15. Fragment de codi inhabilitat mitjançant la sintaxi de comentari (Jocomunico, 2016).

Això pot suposar un problema pel que fa al nombre de peticions que enviem a l'API, atès que invertiríem recursos en peticions innecessàries. Si recollim tot el text que hi ha a continuació dels símbols d'introducció de comentari, també recollirem inevitablement els fragments de codi inhabilitat. Una possible solució per a aquesta qüestió pot ser exportar un fitxer que contingui totes les peticions que s'enviaran a l'API abans d'enviar-les, perquè el traductor humà comprovi que tots els segments són realment comentaris i no fragments de codi. Tenint en compte que a simple vista l'aspecte d'un fragment de codi i un de text són fàcilment diferenciables, fer aquesta revisió no suposaria una gran dificultat, però requeriria un temps que allargaria el procés de traducció en general. A més, si contempléssim aquesta estratègia al pseudocodi, caldria ampliar-lo i incloure-hi un pas en què poguéssim exportar un fitxer i tornar-lo a importar una vegada revisat, tal com ho fem, ara per ara, amb els resultats que ens retorna l'API (vegeu apartat 6.1.4). Si no volem incloure aquest pas en el procés per a no allargar-lo més i trobem que al *ST* s'han filtrat fragments de codi per error, quan treballem amb l'eina TAO, tractarem el segment com a un comentari més que no modificarem. Així, copiem íntegrament el segment d'origen (columna esquerra de l'eina TAO) a la casella corresponent del segment d'arribada (columna dreta de l'eina TAO). En el cas de Memsource, que és l'eina que hem fet servir durant les proves per al marc empíric d'aquest treball, només hem de

prémer la tecla F8. A la fase de substitució del *ST* pel *TT*, el fragment de codi inhabilitat es tornarà a introduir al codi sense patir cap alteració.

Com a segon aspecte bàsic per millorar, parlarem dels resultats de l'API. En funció de l'API que fem servir, podem tenir problemes de detecció de l'idioma, especialment quan es tracta de llengües minoritàries. Si valorem la funcionalitat de l'algorisme, podem dir que aquest és un dels seus punts dèbils, atès que encara que un traductor humà podria revisar els resultats que ens retorna l'API, amb un volum molt gran de fitxers aquesta revisió comportaria una inversió de temps afegit a la traducció pròpiament dita. Així, encara que puguem automatitzar la major part del procés de traducció de comentaris dins de codi informàtic, si traduïm una llengua minoritària, perdrem part d'aquesta automatització i la intervenció humana serà major. Evidentment, això repercutirà en el cost de l'encàrrec. Tanmateix, el preu per paraula que es paga per la traducció de llengües minoritàries acostuma a ser més alt, i això s'aplica també als encàrrecs de traducció objectius del nostre programari, independentment del tipus de text de què es tracti (codi informàtic o text pla).

Per acabar, des d'un punt de vista més ampli, la millora més substancial que podria experimentar el programari que hem dissenyat seria la seva integració en una eina TAO. Treballar amb *STs* que continguin més d'un idioma és un escenari possible que les eines TAO, ara per ara, no contemplen. Si existís aquesta opció, no caldria que el *ST* passés per diverses interfícies i programaris i es simplificaria el procediment enormement. A més, seria interessant que les eines TAO incloguessin un pas intermedi després de la introducció del *ST* i abans de la seva traducció. En aquest pas, podria tenir lloc el reconeixement de l'idioma i la seva revisió, així com la preedició del text de cara a la TA. Per tant, encara que existeixen programaris de preedició de textos, seria interessant que els programaris tinguessin en compte aquesta funció dins del circuit pel qual passa un text des que rebem l'encàrrec de traducció fins que enviem el paquet final al client. Ara mateix, les eines TAO ja faciliten aquest procediment en gran mesura, però no ofereixen cap solució per a casos com el nostre.

7. CONCLUSIONS

Per a tancar la recerca teòrica i pràctica duta a terme en aquest treball de Fi de Grau, farem un repàs del punt de partida del projecte, els objectius establerts i els assolits, les decisions que s'han pres al llarg del treball i les conclusions que podem treure dels coneixements adquirits durant el procés d'aprenentatge.

L'objectiu principal d'aquest treball ha estat proposar una solució pràctica per a un problema de traducció. Aquesta solució pràctica ha consistit en l'elaboració del pseudocodi d'un programari que permet la traducció dels comentaris incrustats dins del codi font d'una aplicació web.

En el marc teòric del treball, a tall d'aproximació introductòria a l'element principal del treball —els comentaris dins del codi informàtic—, en primer lloc s'han definit alguns dels conceptes generals de l'àmbit en què s'emmarca el treball: les tecnologies de la traducció. Així, s'ha parlat de conceptes com ara localització, internacionalització i globalització. També s'ha exposat quina és la funció i la rellevància dels comentaris dins del codi informàtic i s'han comparat les opinions de diversos autors, per tal d'il·lustrar la importància que poden arribar a tenir aquesta mena de textos. Tot seguit, s'ha fet un resum de l'estat actual de l'art en el camp de la programació informàtica i la metodologia *Agile*, per tal de veure quin paper tenen els comentaris en aquest nou model de programació que, a poc a poc, s'ha anat imposant també en la indústria de la traducció.

Pel que fa a l'encàrrec de traducció simulat que ha servit d'exemple al llarg de tot el treball, en primer lloc s'ha descrit l'encàrrec de traducció real en què es basa i el seu context, englobat dins el camp de les aplicacions d'accessibilitat i la Comunicació Augmentativa i Alternativa (CAA).

A continuació, s'ha descrit el *ST* i s'han analitzat tots els problemes que presenta, tant lingüístics com de format. També s'ha descrit el *TT* i s'han establert els objectius de l'encàrrec de traducció simulat, com ara la traducció dels comentaris amb una eina TAO (és a dir, fora del codi informàtic) i la introducció d'una etiqueta per a identificar l'idioma dels comentaris, entre d'altres. Pel que fa a la sintaxi d'introducció i tancament de comentaris dins del codi informàtic, s'han exposat els diferents tipus de sintaxi que havia d'incloure el *TT*. La descripció del *ST* i del *TT* ha servit per a definir quin era el nostre

punt de partida i on volíem arribar. A tall d'exemple, els annexos 9.1 i 9.2 contenen una mostra de 100 línies del *ST* i el *TT* respectivament.

Una vegada l'objectiu del treball ha quedat completament clar, s'han establert algunes prioritats, com ara la rendibilitat del projecte i la comprensió del missatge del text per sobre de la correcció lingüística. Tenint en compte aquests criteris, s'han valorat dues solucions: la formatació manual del text i l'elaboració del pseudocodi d'un algorisme informàtic. Després d'haver tingut en compte els punts forts i febles de totes dues opcions, s'ha optat per l'elaboració del pseudocodi. Això ha requerit l'aprenentatge d'elements bàsics del llenguatge de la programació informàtica (en concret del llenguatge Python™), per tal de poder escriure el pseudocodi amb la major precisió possible i amb la terminologia pròpia de la disciplina en què s'emmarca: la programació informàtica.

A continuació, s'han analitzat diferents estratègies tecnològiques: la traducció humana, la traducció automàtica (TA), full *post-editing* and *light post-editing*. En aquest punt, encara no s'ha escollit una opció, atès que primer calia fer algunes proves i analitzar-ne els resultats.

El marc empíric del treball ha consistit en l'elaboració del pseudocodi d'un programari que solucionés el problema de traducció presentat. El pseudocodi ha estat dividit en set passos: obtenció de l'idioma i la variant geogràfica dels comentaris; anàlisi de l'extensió i la codificació dels fitxers i extracció dels comentaris del codi; API de reconeixement d'idioma; obtenció dels fitxers de treball en format CSV; fase de preedició del text; fase de traducció automàtica i postedició del text (on, després de valorar el resultat de la TA amb i sense preedició, s'ha escollit l'opció *light post-editing*); per últim, substitució del *ST* pel *TT*.

Al començament, s'havia establert que el programari només acceptaria fitxers de codi escrits en llenguatge PHP, HTML, JavaScript o CSS, però que, en canvi, no es limitaria únicament a un parell de llengües específic sinó que es podria aplicar a qualsevol combinació d'idiomes. Malgrat que aquest objectiu s'ha assolit en gran mesura, cal tenir en compte que, si treballem amb *STs* multilingües en llengües minoritàries, el grau d'intervenció del traductor humà serà més elevat. Això té a veure amb el reconeixement d'idioma de l'API, tal com hem explicat a l'apartat 6.1.3.

A l'últim apartat del marc empíric s'han descrit algunes de les millores que es podrien fer al pseudocodi de l'algorisme en el futur.

D'una banda, si parlem de les conclusions a què hem arribat quant al marc teòric d'aquest treball, primerament cal destacar que ha significat un gran aprenentatge més que no pas un assoliment d'objectius o de resultats concloents. En el marc teòric s'han explorat diversos camps que estan relacionats amb la temàtica principal del treball de manera més general (com ara els conceptes de localització, internacionalització i globalització) o més específica (com ara la funció i rellevància dels comentaris dins del codi informàtic i el paper que prenen amb el paradigma de programació emergent conegut com a *Agile*). A més a més, una altra part fonamental del marc teòric ha estat l'anàlisi del text d'origen i de destí, perquè ha permès establir objectius concrets i començar a definir quines funcionalitats havia d'assolir el programari que es volia dissenyar. El recull d'informació realitzat en el camp dels llenguatges de programació i la sintaxi del llenguatge Python™ també ha suposat un aprenentatge d'allò més interessant, atès que, malgrat que es tracta de conceptes que s'engloben en una disciplina que s'allunya molt de la traducció, continuen sent competències pròpies de la nostra professió, la qual té un alt component tecnològic. En aquest mateix àmbit, l'elaboració del pseudocodi ha estat una aproximació molt enriquidora a la lògica que cal seguir per a dissenyar un programari informàtic. Alhora, amb relació a la traducció, la recerca de les diferents estratègies tecnològiques ha estat molt útil per a tenir una idea general de quins són els procediments més habituals dins l'àmbit professional, atès que, mentre som estudiants, les condicions dels treballs que realitzem a la universitat no reprodueixen les de l'entorn professional que trobarem després d'obtenir la titulació, sinó que se centren més en nocions lingüístiques i traductològiques. Per aquest motiu, ha estat sorprenent saber que la traducció automàtica combinada amb la postedició és un procediment tan habitual a les empreses de traducció. Com a futurs traductors, cal que coneguem, encara que sigui d'una manera superficial, les metodologies de treball que més s'utilitzen a la indústria actualment.

D'altra banda, si parlem de les conclusions que podem treure del marc empíric del treball, hem de parlar de la solució que hem creat per a uns problemes de traducció que eren molt concrets. A l'apartat d'introducció s'explicava que la idea que va donar lloc a aquest treball va ser la d'esbrinar de quina manera es podia automatitzar, en la major mesura possible, un encàrrec de traducció real amb unes característiques molt particulars. Bé, podem afirmar que hem trobat una solució que resol bona part dels problemes que presentava la traducció del text. Malgrat que hi ha aspectes de la solució que caldria

millorar i perfeccionar, hem pogut elaborar el pseudocodi d'un programari que ens permetria extreure els comentaris que s'intercalen amb les línies del codi informàtic d'una aplicació o programari per a poder traduir-los amb una eina TAO, cosa que garantiria la consistència i l'homogeneïtat terminològiques. A més, ens donaria l'oportunitat de corregir el text d'origen, si fos necessari, i preeditar-lo per a poder traduir-lo traduir amb un sistema de TA posteriorment. D'aquesta manera, agilitzaríem el procés de traducció i estalviaríem temps i recursos.

Ara bé, si reflexionem sobre el següent pas lògic que caldria fer envers la culminació del programari que s'ha dissenyat en aquest treball, podem parlar de la integració del programari en una eina TAO. Treballar amb *STs* que continguin més d'un idioma és un escenari possible que les eines TAO, ara per ara, no contemplen. Si ho fessin, no caldria que el *ST* passés per diverses interfícies i programaris i es simplificaria el procediment enormement. A més, seria interessant que les eines TAO incloguessin un pas intermedi després de la introducció del *ST* i abans de la seva traducció. En aquest pas intermedi, podria tenir lloc el reconeixement de l'idioma i la seva revisió, així com la preedició del text de cara a la TA. Així, encara que existeixen alguns programaris de preedició de textos, si aquesta funció estigués integrada en el procés que gestionen les eines TAO, la feina del traductor davant d'un projecte amb un *ST* multilingüe es facilitaria molt, perquè no es veuria obligat a dur a terme cada fase del procés en entorns, interfícies i programaris diferents. Per tant, com a conclusió, podem dir que la integració del programari que hem dissenyat en una eina TAO és l'últim pas del procés, tot i que aquest TFG no hi ha arribat. Tanmateix, la porta a la innovació queda oberta i és possible que, en el futur, la idea de base del programari que hem proposat esdevingui una prestació més de les eines TAO.

Paral·lelament als resultats tangibles, teòrics i pràctics obtinguts, aquest TFG també ha comportat una reflexió sobre el paper de la tecnologia en la nostra professió. Mentre estudiem el grau a la universitat, l'única referència que tenim del món professional que ens espera en acabar els estudis són els professors, atès que la majoria d'ells combinen la traducció amb la docència o han exercit la professió en algun punt de la seva vida. Així, els professors ens acosten al món professional mitjançant els seus consells i experiència però, a banda d'això, els estudiants sovint no sabem del cert fins a quin punt les condicions de traducció que tenim durant el grau s'assemblen a les del mercat de treball (pel que fa al temps, a la qualitat exigida, etc.). Tanmateix, després

d'haver cercat informació sobre les diferents estratègies tecnològiques que existeixen, sobre la preedició, la traducció automàtica i la postedició, he après que la TA, tan poc present al grau de Traducció i d'Interpretació, és, en realitat, una eina molt utilitzada en el dia a dia dels traductors professionals, especialment si treballen per a grans empreses de traducció. Així, doncs, em pregunto si en el futur immediat no caldria replantejar-se l'enfocament que es dona a la formació dels traductors i reorientar-lo cap a l'ús de la TA com a recurs vàlid i molt útil del procés de traducció. Això no vol dir, però, que s'hagin de deixar de banda les competències lingüístiques tradicionals del traductor.

En aquest sentit, ara que em trobo al final del camí d'aquest TFG, no puc evitar tenir la sensació de no haver estat conscient de la importància de la tecnologia en la nostra professió fins ara. Partint del fet que reduir costos sempre serà la prioritat de les empreses, he deixat de veure la tecnologia com a una eina al servei dels traductors i he començat a veure-la com a l'element que, ara per ara, dicta les regles del joc. La tecnologia no només provoca canvis constants en les eines de treball del traductor amb la introducció de funcionalitats noves —com podria ser, en el futur, la integració del programari que es proposa en aquest TFG— que fan que cada vegada siguin més completes i que el procés estigui més automatitzat, sinó que el paper del traductor mateix arriba a veure's afectat per tots aquests canvis.

L'estratègia de traducció 'TA amb postedició' o 'preedició amb TA' és una fórmula cada vegada més estesa. Davant d'això, en comptes de deixar-nos endur per la por a una reducció de la demanda de traductors, no tenim més remei que adaptar-nos a aquest nou paradigma i trobar el punt del procediment en què és necessària la nostra intervenció. Tanmateix, és evident que l'ús de la TA pot fer desaparèixer alguns perfils professionals o, si més no, reduir-ne la demanda. Davant d'aquesta situació, no serveix de res sucumbir a la idea que la tecnologia acabarà fent obsoleta la figura del traductor, sinó que, pel bé de la llengua a què traduïm, hem de treballar per a trobar l'espai on podem ser útils, on una màquina mai no podrà substituir-nos (o trigarà a fer-ho). Aquest espai possiblement té a veure amb la gestió dels projectes, que està directament relacionada amb la voluntat del client, però que també està relacionada amb la creació d'un equip de revisors altament qualificats i eficients que s'encarreguin d'impedir que la TA deixi morir les expressions, el parlar familiar, l'argot i, en definitiva, totes les subtileses que passen per alt a les màquines i que conformen la personalitat d'una llengua.

Per tot això, aquest TFG, en realitat, apunta cap a un reciclatge en diversos àmbits, malgrat que ho fa des del punt de vista d'una estudiant i no d'una professional coneixedora de la situació actual del sector. Un reciclatge de les eines TAO, o, si més no, una actualització que incorpori funcionalitats com la que es proposa en aquest treball, d'acord amb el tipus de textos que presenten una necessitat de traducció avui dia. Un reciclatge de l'enfocament de la formació que reben els traductors a les universitats, que hauria de reorientar-se envers la inclusió de les tecnologies i de la TA, com a recurs indispensable en el procés de traducció, i l'assoliment de competències relacionades amb la preedició i la postedició. Per últim, un reciclatge del paper del traductor dins del projecte de traducció, que, d'una banda, ha de treballar conjuntament amb les eines informàtiques i, d'altra banda, amb la part humana i de gestió del projecte, sense limitar-se a la figura de l'expert en idiomes però vetllant per la protecció de les llengües. L'objectiu de tot plegat només ens beneficia a nosaltres, els traductors, tot seguint aquell refrany encunyat per Unamuno que diu: "renovar-se o morir". No podem defugir la necessitat de realitzar canvis en la nostra visió del món, el nostre comportament i els nostres actes com a professionals, si volem compassar-nos al ritme del progrés, trepidant i imparable.

8. BIBLIOGRAFIA

Obres citades:

AGARWAL ET AL. (2010). *Problem Solving with Flowcharts and a Little Flavor of Programming with Python*. 1st ed. (No place), (No Publisher). 197 p. ISBN 9780557276349.

AGILE ALLIANCE (2015). *Manifesto for Agile Software Development*. [Online] Available: https://www.agilealliance.org/agile101/the-agile-manifesto/ [Accessed February 25 2017]

ANASTASIOU, Dimitra, SCHÄLER, Reinhard (2010). "Translating Vital Information: Localisation, Internationalisation, and Globalisation". *Syn-thèses Journal*, volume 3. [Online] The Netherlands: Springer International Publishing AG. Available: http://www.d-anastasiou.com/Publications/Syntheses.pdf [Accessed December 2 2016]

BERGSTEN, Hans (2003). *JavaServer Pages: Help for Server-Side Java Developers*. 3rd ed. California: O'Reilly Media, Inc. 768 p. ISBN 9781449378974.

BOWKER, Lynne (2002). *Java Computer-aided Translation Technology: A Practical* Introduction. Didactics of translation series. 1st ed. Ottawa: University of Ottawa Press. Ill., 185 p. ISBN 9780776605388.

CÁMARA, Lidia. (2001). "El papel de las herramientas TAO en la documentación técnica bilingüe". *Revista Tradumàtica: Traducció i tecnologies de la informació i la comunicació* [en línia]. Núm. 0 (octubre) Disponible a: <http://www.fti.uab.es/tradumatica/revista/num0/articles/lcamara/central.htm> [Consulta: 13 de gener del 2016]

COHEN, David; LINDVALL, Mikael; COSTA, Patricia (2003). "Agile software development". *DACS SOAR Report* [Online]. Available: http://users.jyu.fi/~mieijala/kandimateriaali/Agile%20software%20development.pdf [Accessed February 25 2017]

COLLABNET INC. (2011). *Reinforcing Agile Software Development in the Cloud* [Online]. Available: https://www.open.collab.net/media/pdfs/CollabNet%20Whitepaper_Reinforcing%20Agile%20Dev%20in%20the%20Cloud.pdf?_=d [Accessed February 25 2017]

CONFEDERACIÓN ASPACE (2013). *Comunicación Aumentativa y Alternativa (CAA)* [en línea]. Disponible en: <http://aspacenet.aspace.org/main-menu/informacion-basica/item/153-comunicaci%C3%B3n-aumentativa-y-alternativa-caa> [Consulta: 2 de enero de 2017]

CORTE FERNÁNDEZ, Noelia (2002). "Localización e internacionalización de sitios web". *Revista Tradumàtica: Traducció i tecnologies de la informació i la comunicació* [en

línia]. Núm. 1 (octubre); Londres (City University): Electronic Publishing. Disponible en: <http://www.fti.uab.es/tradumatica/revista/articles/ncorte/art.htm> [Consulta: 2 de diciembre de 2016]

DAVIS, Michele E.; PHILLIPS, Jon A. (2007). *Learning PHP & MySQL: Step-by-Step Guide to Creating Database-Driven Web Sites*. 2nd ed. California: O'Reilly Media, Inc. 432 p. ISBN 9780596551650.

DENNETT, Daniel C. (2015). *Bombas de intuición y otras herramientas del pensamiento.* 1.ª ed. Traducción de Laura Lecuona. Méjico: Fondo de Cultura Económica. 337 p. ISBN 9786071631398.

DIMES, Troy (2015). *Conceptos Básicos De Scrum: Desarrollo De Software Agile Y Manejo De Proyectos Agile*. 48 p. ISBN 9781507102732.

FLAIG, Ruediger-Marcus. (2011). *Bioinformatics Programming in Python: A Practical Course for Beginners*. 1st ed. Germany: Wiley-VCH Verlag GmbH & Co. 428 p. ISBN 9783527644902.

HUTCHINS, W. John; SOMERS, L. Harold. (1992). *An introduction to machine translation*. London: Academic Press. 362 p. ISBN 0-12-362830-X.

INGE, Barbara K. (2006). "Terminology workflow in the localization process". Keiran J. Dunne (ed.) *Perspectives on Localization*. Amsterdam: John Benjamins Publishing Company, p. 173-191.

IRFANULLAH, Mohammed (2014). *HTML.* (No place), Sanria Books. 342 p. ISBN 9781503389304.

KANTANMT BLOG. (2013). *How to Write for Machine Translation* [Online]. Available: https://kantanmtblog.com/2013/07/03/how-to-write-for-mt/ [Accessed April 6 2017]

KNUTH, Donald E. (1992). *Literate Programming*. 1st ed. *Perspectives on Localization*. Stanford: CSLI Publications. 384 p. ISBN 9780937073803.

LLANOS FERRARIS, Diego Rafael (2010). *Fundamentos de informática y programación en C.* 1.ª ed. Madrid: Editorial Paraninfo. 392 p. ISBN 9788497327923.

MARTÍN-MOR, Adrià; PIQUÉ, Ramón; SÁNCHEZ-GIJÓN, Pilar (2016). *Tradumàtica: Tecnologies de la traducció*. 1a ed. Vic: Eumo Editorial (Biblioteca de Traducció i Interpretació; 21). 141 p. ISBN 9788497665704.

MORA PÉREZ, José Juan (2015). *DevOps y el camino de baldosas amarillas*. 1.ª ed. [s.l.]: Createspace Independent Publishing Platform. 294 p. ISBN 9781512191974.

MUZII, Luigi (2016). *Post-editing of Machine Translation for Project Managers*. The Netherlands: TAUS and sQuid. ISBN 9781365595844.

PAGANS, Marta. (2002). "Localització, ens ubiquem?" [en línia]. *Revista Tradumàtica: Traducció i tecnologies de la informació i la comunicació*. Núm. 1 (octubre) Disponible a: <http://www.fti.uab.es/tradumatica/revista/articles/mpagans/art.htm> [Consulta: 2 de desembre del 2016]

PAHISA-SOLÉ, Joan. (2016). *Jocomunico. L'app de CAA que parla amb naturalitat*. [lloc web]. Disponible a: <http://jocomunico.com/#/home> [Consulta: 23 de desembre del 2016]

POWERS, Shelley (2008). *Learning JavaScript: Add Sparkle and Life to Your Web Pages.* 2nd ed. California: O'Reilly Media, Inc. 398 p. ISBN 9780596554378.

PROGRAMIZ (2017). *Learn Python Programming: The Definitive Guide* [Online]. Available: https://www.programiz.com/python-programming [Accessed February 25 2017]

ROTURIER, Johann (2015). *Localizing Apps: A Practical Guide for Translators and Translation Students* (1st ed.). New York-London: Routledge. 222 p. ISBN 9781317621676.

RUBIN, Dan; LLOYD, Ian; CROFT, Jeffrey (2007). *Pro CSS Techniques.* 1st ed. California: Apress. 408 p. ISBN 9781430203353.

SAJAN. (2014). *6 expert writing tips for better machine translation results* [Online]. Available: https://www.sajan.com/6-expert-writing-tips-better-machine-translation-results/ [Accessed April 6 2017]

TAUS (2017). *MT Post-editing Guidelines* [Online]. Available: https://www.taus.net/academy/best-practices/postedit-best-practices/machine-translation-post-editing-guidelines [Accessed April 3 2017]

TERMCAT (2017). "Algorisme" [en línia]. Barcelona: TERMCAT, Centre de Terminologia. Disponible a: <http://www.termcat.cat/ca/Cercaterm/Fitxa/algorisme/MTUyMDQzMw==/#.WMBaPvP_06U.link> [Consulta: 2 de febrer del 2016]

TERMCAT (2016). "Pseudocodi" [en línia]. Barcelona: TERMCAT, Centre de Terminologia. Disponible a: <http://www.termcat.cat/ca/Cercaterm/Fitxa/pseudocodi/MjgwMzAzOQ==/#.WMBZzaTEQ40.link> [Consulta: 8 de desembre del 2016]

TERMCAT, Centre de terminologia (2008) [en línia]. Barcelona: TERMCAT, Centre de Terminologia. Disponible a: <http://www.termcat.cat/ca> [Consulta: 8 de gener del 2017]

THOMPSON, Alfred. (2011). "Computer Science Teacher – Thoughts and Information from Alfred Thompson: Why are all programming languages in English?". *MSDN Magazine* – Blogs [Online]. Microsoft Corporation (ed.). Available: https://blogs.msdn.microsoft.com/alfredth/2011/07/21/why-are-all-programming-languages-in-english/ [Accessed January 5 2017]

YOURDON, Edward (1988)."Flashes on Maintenance From Techniques of Program Structure and Design". *Techniques of Program and System Maintenance*. 2nd ed. Massachussets: QED Information *Sciences, Inc. Wellesley*. 461 p. ISBN 0-89435-231-8.

ZOKAITIES, David (2002). "Writing Understandable Code". *Software Development*. Volume 10 (January), p. 48-49. University of Michigan: Miller Freeman, Incorporated.

9. ANNEXOS

9.1. Mostra del *ST*, 100 línies de codi (Jocomunico, 2016)

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Myslot {


    // PROPIETATS
    var $category; // Tipus d'slot -> Theme, Subj... No té els 1|2 del final que tenen les keys de l'slotarray del pattern si hi ha subverbs
    var $grade; // Si és obligatori, opt o no pot ser-hi
    var $type = null; // Si vol qualsevol nom, joguines, adverbis, verbs... (si és obligatori o optatiu)
    var $defvalue = null; // Valor per defecte si és de grade obligatori
    var $prep = null; // Preposició que precedeix a l'slot
    var $art = null;
    var $defvalueused = false; // indica si l'slot s'ha omplert amb el def value
    var $verbless = false; // per quan no han introduït verb i s'afegeix després amb les columnes
                           // defaultverb o els verbless patterns

    var $full = false; // Si l'slot ja està ple/bloquejat
    var $paraulafinal = null; // Paraula que acaba omplint l'slot (classe Myword)
    var $puntsfinal; // punts segons com de bo és el fit de la paraula a l'slot
    var $indexclassfinalword = 0; // L'index de la classe per saber quina classe s'ha agafat si una paraula en tenia vàries
    var $puntsguanyats = -1000;

    var $paraulestemp = array(); // Array de les paraules/myword [0], els seus punts [1] que poden omplir l'slot i l'index de la classe [2]

    var $level = 1; // Nivell on es troba l'slot, si és d'un slot de subverb serà 2
    var $parent; // Si l'slot és de nivell 2, aquí hi ha el nom de l'slot original que substitueixen
                 // en general el que era subverb

    var $complements = array(); // ARRAY de slots pels complements de nom (NC) que siguin noms.
    var $NCassigned = false;
    var $NCassignedkey = null;

    var $cmpAdjs = array(); // ARRAY de slots pels complements de nom (NC) que siguin adjectius.
    var $CAdjassigned = false;
    var $CAdjassignedkey = null;

    var $cmpAdvs = array(); // ARRAY de slots pels complements de nom (NC) que siguin adverbis (com ara pels de lloc, "davant" la taula).
    var $CAdvassigned = false;
    var $CAdvassignedkey = null;

    var $cmpMod = array(); // ARRAY de slots pels complements de nom (NC) que siguin modificadors (quantificadors...).
    var $CModassigned = false;
    var $CModassignedkey = array();

    /*
     * VARIABLES PEL GENERADOR
     */

    var $slotstring = array(); // array amb la representació escrita de l'slot on a cada
                               // posició hi té una tupla amb [0] la forma escrita de la paraula
                               // i [1] una referència a la paraula (myWord) si no és una
                               // prep, un article,  conj... [2] si el nucli és un nom [3] si masc [4] si plural
                               // [5] si no necessita article -> quan hi ha un numero davant del nom o un quantificador
                               // [6] si el nom és un complement de nom [7] si el nucli té un possessiu, que l'article
                               // anirà davant del possessiu

    var $isInfinitive = false; // only for verbs. If after conjugating them, they are in infinitive form.


    function __construct() {}

    public function isFull()
    {
        return $this->full;
    }

    public function nounFitsSlot($word, $keyslot)
    {
        $CI = &get_instance();
        $CI->load->library('Mymatching');

        $matching = new Mymatching();

        $langnouncorder = $CI->session->userdata('uinterfacelangncorder');

        $numclasses = count($word->classes);

        $matchscore = 1000;
        $matchindexclass = -1;
        $output = 0;
        $isPronoun = false;

        for ($i=0; $i<$numclasses; $i++) {

            if ($word->classes[$i] == "pronoun") $isPronoun = true;
            // comprovem que la classe de nom existeixi i que el type de l'slot sigui de nom
            if ($matching->isSetKeyNoun($word->classes[$i]) && $matching->isSetKeyNoun($this->type)) {
```

```
88                  $tipusx = $matching->nounsFitKeys[$this->type]; // agafem l'index del tipus de nom de l'slot
89                  $tipusy = $matching->nounsFitKeys[$word->classes[$i]];
90
91                  if ($matching->nounsFit[$tipusx][$tipusy] < $matchscore) {
92                      $matchscore = $matching->nounsFit[$tipusx][$tipusy];
93                      $matchindexclass = $i;
94                  }
95
96              }
97          }
98
99          // mirar si el nom pot fer de complement de nom d'algun dels noms que estiguin de provisionals a l'slot
100         // Si és un slot de tipus nom i la paraula no és un pronom (que no poden fer de NC)
101         if (!$isPronoun && $matching->isSetKeyNoun($this->type)) { // HO MIREM SEMPRE, JA QUE una paraula POT SER AL LLISTAT PROV DE L'SLOT I
102                                                                    // DE NC D'UN ALTRE NOM
```

9.2. Mostra del *TT*, 100 línies de codi (Jocomunico, 2016)

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Myslot {


 // en-US: FEATURES
 // ca-ES: PROPIETATS
 // es-ES: PROPIEDADES
    var $category; /* en-US: Slot type -> Theme, subject... It doesn't have 1|2 attached at the end of the key of the patterns' slotarray
                      keys if there are any subverbs.*/
                   /* ca-ES: Tipus d'slot -> Theme, Subj... No té els 1|2 del final que tenen les keys de l'slotarray dels patrons si hi
                      ha subverbs.*/
                   /*es-ES: Tipo de slot -> Theme, Subj... No tiene los 1|2 del final que tienen las keys de la slotarray de los patrones si
                   hay subverbos.*/
    var $grade; // en-US: If the slot is mandatory, optional, or if it is not used.
                // ca-ES: Si és obligatori, opt o no pot ser-hi.
                // es-ES: Si es obligatorio, opt, o no puede haber.
    var $type = null; // en-US: If there is any preferred type of name, toys, adverbs, verbs... (if it is mandatory or optional).
                      // ca-ES: Si vol qualsevol tipo de nom, joguines, adverbis, verbs... (si és obligatori o optatiu).
                      // es-ES: Si necesita cualquier tipo de nombre, juguete, adverbio, verbo...(si es obligatorio o optativo).
    var $defvalue = null; // en-US: Default value if the slot is graded as mandatory.
                          // ca-ES: Valor per defecte si és de grade obligatori.
                          // es-ES: Valor por defecte si es de grade obligatorio.
    var $prep = null; // en-US: Preposition that preceeds the slot.
                      // ca-ES: Preposició que precedeix a l'slot.
                      // es-ES: Preposición que precede al slot.
    var $art = null;
    var $defvalueused = false; // en-US: It indicates if the slot has been filled with the default value.
                               // ca-ES: Indica si l'slot s'ha omplert amb el def value.
                               // es-ES: Indica si el slot se ha llenado con el def value.
    var $verbless = false; /*en-US: When no verb was inputted and it is added later using the values in defaultverb columns
                           or a verbless pattern.*/
                           /*ca-ES: Per quan no s'ha introduït cap verb i s'afegeix després amb les columnes
                           defaultverb o els verbless patterns.*/
                           /*es-ES: Para cuando no se ha introducido ningún verbo y se añade después con las columnas defaultverb
                           o los verbless patterns*/
    var $full = false; // en-US: If the slot is already filled or blocked.
                       // ca-ES: Si l'slot ja està ple o bloquejat.
                       // es-ES: Si el slot ya está lleno o bloqueado.
    var $paraulafinal = null; // en-US: Word that will finally fill the slot (class Myword).
                              // ca-ES: Paraula que acaba omplint l'slot (classe Myword).
                              // es-ES: Palabra que acaba llenando el slot (clase Myword).
    var $puntsfinal; // en-US: Score depending on how good is the fit of the word in the slot.
                     // ca-ES: Punts segons com de bo és el fit de la paraula a l'slot.
                     // es-ES: Puntos según cómo de bueno es el fit de la palabra en el slot.
    var $indexclassfinalword = 0; // en-US: Class index in order to know which class has been used if the word has several classes.
                                  // ca-ES: L'index de la classe per saber quina classe s'ha agafat si una paraula en tenia vàries.
                                  // es-ES: El index de la clase para saber qué clase se ha cogido si una palabra tenía varias.
    var $puntsguanyats = -1000;

    var $paraulestemp = array(); /*en-US: Array of the words/myword [0], the score [1]*/
                                 /*ca-ES: Array de les paraules/myword [0], els seus punts [1] que poden omplir l'slot i l'index
                                 de la classe [2].*/
                                 /*es-ES: Array de las palabras/myword [0], sus puntos [1] que pueden llenar el slot y el index
                                 de la clase [2].*/
    var $level = 1; // en-US: Level of the slot: if it is a subverb slot, it will be level 2.
                    // ca-ES: Nivell on es troba l'slot, si és d'un slot de subverb serà 2.
                    // es-ES: Nivel donde se encuentra el slot, si es de un slot de subverb será 2.
    var $parent; /*en-US: If it is a slot of level 2, here we will find the name of the original slot that is being replaced,
                 which is usually the subverb slot.*/
                  /*ca-ES: Si l'slot és de nivell 2, aquí hi ha el nom de l'slot original que substitueixen,
                 en general el que era subverb.*/
                  /*es-ES: Si el slot es de nivel 2, aquí hay el nombre del slot original que sustituyen, en general el que era subverb.*/
    var $complements = array(); // en-US: Slot ARRAY for noun complements (NC) if they are names.
                                // ca-ES: ARRAY de slots pels complements de nom (NC) que siguin noms.
                                // es-ES: ARRAY de slots para los complementos de nombre (NC) que sean nombres.
    var $NCassigned = false;
    var $NCassignedkey = null;
```

```
69
70      var $cmpAdjs = array(); // en-US: Slot ARRAY for noun complements (NC) if they are adjectives.
71                              // ca-ES: ARRAY de slots pels complements de nom (NC) que siguin adjectius.
72                              // es-ES: ARRAY de slots para los complementos de nombre (NC) que sean adjetivos.
73      var $CAdjassigned = false;
74      var $CAdjassignedkey = null;
75
76      var $cmpAdvs = array(); /*en-US: Slot ARRAY for noun complements (NC) if they are adverbs (such as adverbs of place:
77                              "in front of" the table).*/
78                              /*ca-ES: ARRAY de slots pels complements de nom (NC) que siguin adverbis (com ara pels de lloc:
79                              "davant" la taula).*/
80                              /*es-ES: ARRAY de slots por los complementos de nombre (NC) que sean adverbios (como por ejemplo
81                              por los de lugar: "delante de" la mesa)*/
82      var $CAdvassigned = false;
83      var $CAdvassignedkey = null;
84
85      var $cmpMod = array(); // en-US: lot ARRAY for noun complements (NC) if they are modifiers (quantifiers, etc.).
86                             // ca-ES: ARRAY de slots pels complements de nom (NC) que siguin modificadors (quantificadors, etc.).
87                             // es-ES: ARRAY de slots por los complementos de nombre (NC) que sean modificadores (cuantificadores, etc.).
88      var $CModassigned = false;
89      var $CModassignedkey = array();
90
91      /*
92      en-US: VARIABLES OF THE GENERATOR
93      ca-ES: VARIABLES PEL GENERADOR
94      es-ES: VARIABLES PARA EL GENERADOR
95      */
96
97
98      var $slotstring = array(); /*en-US: Array with the written representation of the slot, where every position has a pair with [0]
99                              the written form of the word and [1] a reference to the word (myWord) if the word is not a preposition,
100                             an article, a conjunction... [2] if the nucleus is a name [3] if it is masc [4] if it is plural
101                             [5] if it doesn't need any article -> if there is a number or a quantifier before the name
102                             [6] if the name is a noun complement [7] ***ONLY IN CATALAN*** if the nucleus has a possessive adjective,
103                             the article will be placed before the possessive article.*/
104
105                             /*ca-ES: Array amb la representació escrita de l'slot on a cada posició té una tupla amb [0] la forma
106                             escrita de la paraula i [1] una referència a la paraula (myWord) si no és una preposició, un article,
107                             una conjunció... [2] si el nucli és un nom [3] si és masculí [4] si és plural [5] si no necessita
108                             article -> quan hi ha un número o un quantificador davant del nom o [6] si el nom és un complement
109                             de nom [7] ***NOMÉS EN CATALÀ*** i el nucli té un possessiu, que l'article anirà davant del possessiu.*/
110
111                             /*es-ES: Array con la representación escrita de la slot dónde a cada posición tiene una tupla con [0] la forma
112                              escrita de la palabra y [1] una referencia a la palabra (myWord) si no es una preposición, un artículo,
113                             una conjunción... [2] si el núcleo es un nombre [3] si es masculino [4] si es plural [5] si no necesita
114                             artículo -> cuando hay un número o un cuantificador ante el nombre o [6] si el nombre es un complemento
115                             de nombre [7] *SÓLO EN CATALÁN** y el núcleo tiene un posesivo, que el artículo irá ante el posesivo.*/
116
117     var $isInfinitive = false; // en-US: Only for verbs. If after conjugating them, they are in infinitive form.
118                             // ca-ES: Només pels verbs. Si després de conjugar-los, estan en infinitiu.
119                             // es-ES: Sólo para los verbos. Si después de conjugarlos, están en infinitivo.
120
121     function __construct() {}
122
123     public function isFull()
124     {
125         return $this->full;
126     }
127
128     public function nounFitsSlot($word, $keyslot)
129     {
130         $CI = &get_instance();
131         $CI->load->library('Mymatching');
132
133         $matching = new Mymatching();
134
135         $langnouncorder = $CI->session->userdata('uinterfacelangncorder');
136
137             if ($word->classes[$i] == "pronoun") $isPronoun = true;
138             // en-US: We check if the name class exists and if names are accepted by the slot type.
139             // ca-ES: Comprovem que la classe de nom existeixi i que el type de l'slot sigui de nom.
140             // es-ES: Comprobamos que la clase de nombre exista y que el type del slot sea de nombre.
141             if ($matching->isSetKeyNoun($word->classes[$i]) && $matching->isSetKeyNoun($this->type)) {
142                 $tipusx = $matching->nounsFitKeys[$this->type]; // en-US: We get the index of the name type of the slot.
143                                                                 // ca-ES: Agafem l'index del tipus de nom de l'slot.
144                                                                 // es-ES: Cogemos el index del tipo de nombre del slot.
145                 $tipusy = $matching->nounsFitKeys[$word->classes[$i]];
146
147                 if ($matching->nounsFit[$tipusx][$tipusy] < $matchscore) {
148                     $matchscore = $matching->nounsFit[$tipusx][$tipusy];
149                     $matchindexclass = $i;
150                 }
151
152             }
153         }
154
155         /*en-US: We check if the word could act as a noun complement (NC) of any of the names that are temporarily fitting a slot,
156         if it is a type-name slot and if the word is not a pronoun (which cannot act as NC).*/
157         /*ca-ES: Mirar si el nom pot fer de complement de nom (NC) d'algun dels noms que estiguin de provisionals a l'slot,
158         si és un slot de tipus nom i la paraula no és un pronom (que no poden fer de NC).*/
159         /*es-ES: Mirar si el nombre puede hacer de complemento de nombre (NC) de alguno de los nombres que estén de provisionales
160         en el slot, si es un slot de tipo nombre y la palabra no es un pronombre (que no pueden hacer de NC).*/
161         if (!$isPronoun && $matching->isSetKeyNoun($this->type)) { /*en-US: We always check this, since a word might be in a temporary list
162                                                                     to fit a slot and in the nc list of another noun.*/
163                                                                     /*ca-ES: Ho mirem sempre, ja que una paraula pot ser al llistat provisional
164                                                                     de l'slot i al llistat de nc d'un altre nom.*/
165                                                                     /*es-ES: Lo miramos siempre, puesto que una palabra puede ser al listado
166                                                                     provisional del slot y al listado de nc de otro nombre.*/
```

9.3. ISO 639: Codis d'idioma

ISO-639 Language Codes

Language Name	Language Code	Language Name	Language Code
Abkhazian	ab	Latvian (Lettish)	lv
Afar	aa	Limburgish (Limburger)	li
Afrikaans	af	Lingala	ln
Albanian	sq	Lithuanian	lt
Amharic	am	Macedonian	mk
Arabic	ar	Malagasy	mg
Armenian	hy	Malay	ms
Assamese	as	Malayalam	ml
Aymara	ay	Maltese	mt
Azerbaijani	az	Maori	mi
Bashkir	ba	Marathi	mr
Basque	eu	Moldavian	mo
Bengali (Bangla)	bn	Mongolian	mn
Bhutani	dz	Nauru	na
Bihari	bh	Nepali	ne
Bislama	bi	Norwegian	no
Breton	br	Occitan	oc
Bulgarian	bg	Oriya	or
Burmese	my	Oromo (Afan, Galla)	om
Byelorussian (Belarusian)	be	Pashto (Pushto)	ps
Cambodian	km	Polish	pl
Catalan	ca	Portuguese	pt
Chinese (Simplified)	zh	Punjabi	pa
Chinese (Traditional)	zh	Quechua	qu
Corsican	co	Rhaeto-Romance	rm
Croatian	hr	Romanian	ro
Czech	cs	Russian	ru
Danish	da	Samoan	sm
Dutch	nl	Sangro	sg
English	en	Sanskrit	sa
Esperanto	eo	Serbian	sr
Estonian	et	Serbo-Croatian	sh
Faeroese	fo	Sesotho	st
Farsi	fa	Setswana	tn
Fiji	fj	Shona	sn
Finnish	fi	Sindhi	sd

French	fr	Sinhalese	si
Frisian	fy	Siswati	ss
Galician	gl	Slovak	sk
Gaelic (Scottish)	gd	Slovenian	sl
Gaelic (Manx)	gv	Somali	so
Georgian	ka	Spanish	es
German	de	Sundanese	su
Greek	el	Swahili (Kiswahili)	sw
Greenlandic	kl	Swedish	sv
Guarani	gn	Tagalog	tl
Gujarati	gu	Tajik	tg
Hausa	ha	Tamil	ta
Hebrew	he	Tatar	tt
Hindi	hi	Telugu	te
Hungarian	hu	Thai	th
Icelandic	is	Tibetan	bo
Indonesian	id	Tigrinya	ti
Interlingua	ia	Tonga	to
Interlingue	ie	Tsonga	ts
Inuktitut	iu	Turkish	tr
Inupiak	ik	Turkmen	tk
Irish	ga	Twi	tw
Italian	it	Uighur	ug
Japanese	ja	Ukrainian	uk
Javanese	ja	Urdu	ur
Kannada	kn	Uzbek	uz
Kashmiri	ks	Vietnamese	vi
Kazakh	kk	Volapük	vo
Kinyarwanda (Ruanda)	rw	Welsh	cy
Kirghiz	ky	Wolof	wo
Kirundi (Rundi)	rn	Xhosa	xh
Korean	ko	Yiddish	yi
Kurdish	ku	Yoruba	yo
Laothian	lo	Zulu	zu
Latin	la		

9.4. ISO 3166: Codis de país

ISO-3166 Country Codes			
Country Name	**Country Code**	**Country Name**	**Country Code**
AFGHANISTAN	AF	LIBYAN ARAB JAMAHIRIYA	LY
ALBANIA	AL	LIECHTENSTEIN	LI
ALGERIA	DZ	LITHUANIA	LT
AMERICAN SAMOA	AS	LUXEMBOURG	LU
ANDORRA	AD	MACAO	MO
ANGOLA	AO	MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MK
ANTARCTICA	AQ	MADAGASCAR	MG
ANTIGUA AND BARBUDA	AG	MALAWI	MW
ARGENTINA	AR	MALAYSIA	MY
ARMENIA	AM	MALDIVES	MV
ARUBA	AW	MALI	ML
AUSTRALIA	AU	MALTA	MT
AUSTRIA	AT	MARSHALL ISLANDS	MH
AZERBAIJAN	AZ	MARTINIQUE	MQ
BAHAMAS	BS	MAURITANIA	MR
BAHRAIN	BH	MAURITIUS	MU
BANGLADESH	BD	MAYOTTE	YT
BARBADOS	BB	MEXICO	MX
BELARUS	BY	MICRONESIA, FEDERATED STATES OF	FM
BELGIUM	BE	MOLDOVA, REPUBLIC OF	MD
BELIZE	BZ	MONACO	MD
BENIN	BJ	MONGOLIA	MN
BERMUDA	BM	MONTSERRAT	MS
BHUTAN	BT	MOROCCO	MA
BOLIVIA	BO	MOZAMBIQUE	MZ
BOSNIA AND HERZEGOVINA	BA	MYANMAR	MM
BOTSWANA	BW	NAMIBIA	NA
BOUVET ISLAND	BV	NAURU	NR

BRAZIL	BR	NEPAL	NP
BRITISH INDIAN OCEAN TERRITORY	IO	NETHERLANDS	NL
BRUNEI DARUSSALAM	BN	NETHERLANDS ANTILLES	AN
BULGARIA	BG	NEW CALEDONIA	NC
BURKINA FASO	BF	NEW ZEALAND	NZ
BURUNDI	BI	NICARAGUA	NI
CAMBODIA	KH	NIGER	NE
CAMEROON	CM	NIGERIA	NG
CANADA	CA	NIUE	NU
CAPE VERDE	CV	NORFOLK ISLAND	NF
CAYMAN ISLANDS	KY	NORTHERN MARIANA ISLANDS	MP
CENTRAL AFRICAN REPUBLIC	CF	NORWAY	NO
CHAD	TD	OMAN	OM
CHILE	CL	PAKISTAN	PK
CHINA	CN	PALAU	PW
CHRISTMAS ISLAND	CX	PALESTINIAN TERRITORY, OCCUPIED	PS
COCOS (KEELING) ISLANDS	CC	PANAMA	PA
COLOMBIA	CO	PAPUA NEW GUINEA	PG
COMOROS	KM	PARAGUAY	PY
CONGO	CG	PERU	PE
CONGO, THE DEMOCRATIC REPUBLIC OF THE	CD	PHILIPPINES	PH
COOK ISLANDS	CK	PITCAIRN	PN
COSTA RICA	CR	POLAND	PL
CÔTE D'IVOIRE	CI	PUERTO RICO	PR
CROATIA	HR	QATAR	QA
CUBA	CU	RÉUNION	RE
CYPRUS	CY	ROMANIA	RO
CZECH REPUBLIC	CZ	RUSSIAN FEDERATION	RU
DENMARK	DK	RWANDA	RW
DJIBOUTI	DJ	SAINT HELENA	SH
DOMINICA	DM	SAINT KITTS AND NEVIS	KN
DOMINICAN REPUBLIC	DO	SAINT LUCIA	LC

ECUADOR	EC	SAINT PIERRE AND MIQUELON	PM
EGYPT	EG	SAINT VINCENT AND THE GRENADINES	VC
EL SALVADOR	SV	SAMOA	WS
EQUATORIAL GUINEA	GQ	SAN MARINO	SM
ERITREA	ER	SAO TOME AND PRINCIPE	ST
ESTONIA	EE	SAUDI ARABIA	SA
ETHIOPIA	ET	SENEGAL	SN
FALKLAND ISLANDS (MALVINAS)	FK	SERBIA AND MONTENEGRO	CS
FAROE ISLANDS	FO	SEYCHELLES	SC
FIJI	FJ	SIERRA LEONE	SL
FINLAND	FI	SINGAPORE	SG
FRANCE	FR	SLOVAKIA	SK
FRENCH GUIANA	GF	SLOVENIA	SI
FRENCH POLYNESIA	PF	SOLOMON ISLANDS	SB
FRENCH SOUTHERN TERRITORIES	TF	SOMALIA	SO
GABON	GA	SOUTH AFRICA	ZA
GAMBIA	GM	SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	GS
GEORGIA	GE	SPAIN	ES
GERMANY	DE	SRI LANKA	LK
GHANA	GH	SUDAN	SD
GIBRALTAR	GI	SURINAME	SR
GREECE	GR	SVALBARD AND JAN MAYEN	SJ
GREENLAND	GL	SWAZILAND	SZ
GRENADA	GD	SWEDEN	SE
GUADELOUPE	GP	SWITZERLAND	CH
GUAM	GU	SYRIAN ARAB REPUBLIC	SY
GUATEMALA	GT	TAIWAN, PROVINCE OF CHINA	TW
GUINEA	GN	TAJIKISTAN	TJ
GUINEA-BISSAU	GW	TANZANIA, UNITED REPUBLIC OF	TZ
GUYANA	GY	THAILAND	TH

HAITI	HT	TIMOR-LESTE	TL
HEARD ISLAND AND MCDONALD ISLANDS	HM	TOGO	TG
HONDURAS	HN	TOKELAU	TK
HONG KONG	HK	TONGA	TO
HUNGARY	HU	TRINIDAD AND TOBAGO	TT
ICELAND	IS	TUNISIA	TN
INDIA	IN	TURKEY	TR
INDONESIA	ID	TURKMENISTAN	TM
IRAN, ISLAMIC REPUBLIC OF	IR	TURKS AND CAICOS ISLANDS	TC
IRAQ	IQ	TUVALU	TV
IRELAND	IE	UGANDA	UG
ISRAEL	IL	UKRAINE	UA
ITALY	IT	UNITED ARAB EMIRATES	AE
JAMAICA	JM	UNITED KINGDOM	GB
JAPAN	JP	UNITED STATES	US
JORDAN	JO	UNITED STATES MINOR OUTLYING ISLANDS	UM
KAZAKHSTAN	KZ	URUGUAY	UY
KENYA	KE	UZBEKISTAN	UZ
KIRIBATI	KI	VANUATU	VU
KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP	VENEZUELA	VE
KOREA, REPUBLIC OF	KR	VIET NAM	VN
KUWAIT	KW	VIRGIN ISLANDS, BRITISH	VG
KYRGYZSTAN	KG	VIRGIN ISLANDS, U.S.	VI
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LA	WALLIS AND FUTUNA	WF
LATVIA	LV	WESTERN SAHARA	EH
LEBANON	LB	YEMEN	YE
LESOTHO	LS	ZAMBIA	ZM
LIBERIA	LR	ZIMBABWE	ZW

9.5. Resultat de la TA en català abans i després de la preedició

***ST* en català sense preedició**	**TA en castellà**	**TA en anglès**
amb tota la **info** recollida, movem els pronoms de lloc si cal	con toda la colección de información, se mueven los pronombres de sitio si es necesario	with all the info collection, move the site pronouns if necessary
l'hola sempre va a davant i **les** que tenen **1** a la propietat front	Hola era siempre en frente y tener 1 en el frente de la propiedad	Hello was always in front and they have 1 on the property front
si hi ha un verb de desig, posar**-los** abans**,** amb **els** de permís **van** darrere **del** permís	Si hay un verbo del deseo, los pongo antes de salir de con el permiso	If there is a verb of desire, put them before they leave behind with permission
indiquem**, que si la frase era negativa,** ja no caldrà afegir el **no**	indica, que si la frase es negativa, ya no es necesario añadir la no-	indicates, that if the phrase was negative, no longer need to add the non-
si és una **ordre els** pronoms aniran darrere el verb i tindran una altra forma	Si es un pedido será pronombres detrás del verbo y otra manera	If it is an order will be pronouns behind the verb and have another way

***ST* en català amb preedició**	**TA en castellà**	**TA en anglès**
amb tota la **informació** recollida, movem els pronoms de lloc si cal	con toda la información recogida, se mueven los pronombres de sitio si es necesario	with all information collected, move the site pronouns if necessary
la paraula "hola" i les **paraules** que tenen **el valor** 1 a la propietat front **sempre van davant**	la palabra "Hola" y las palabras que tienen el valor 1 en el frente de la propiedad siempre en frente	the word "Hello" and the words that have the value 1 on the property front always in front
si hi ha un verb de desig, posar **el verb de desig** abans. **En el cas dels verbs** de permís, **els verbs de desig** han d'anar darrere del **verb de** permís	Si hay un verbo del deseo, poner el verbo de deseo antes. En el caso de los verbos de permiso, los verbos de deseo deben ir detrás del verbo de permiso	If there is a verb of desire, put the verb of desire before. In the case of verbs of permission, the verbs of desire must be behind the verb of permission
si la frase era negativa, ja no caldrà afegir la paraula "no"	Si la frase es negativa, ya no es necesario agregar la palabra "no"	If the phrase was negative, no longer need to add the word "not"
si és una ordre, els pronoms aniran darrere el verb i tindran una altra forma	Si es un comando, los pronombres se detrás del verbo y otra manera	If it is a command, the pronouns will be behind the verb and have another way

9.6. Resultat de la TA en castellà abans i després de la preedició

***ST* en castellà sense preedició**	**TA en català**	**TA en anglès**
Imagenes	Imatges	Images
Comprobar que **ha entrado** texto en el campo nombre	Comproveu que hàgiu introduït text en el nom del camp	Check that you have entered text into the field name
Usamos una variable en vez del return **por que** la función promise tarda **mas** en retornar el resultado y **nos dava** error al comprobarlo en el submit	Utilitzem una variable en comptes el retorn per la funció té la promesa més per retornar el resultat i ens e dava no ha pogut comprovar les sotmeten	We use a variable instead of the return by the function takes promise more to return the result and us dava failed to check it on the submit
Ponemos como idioma por defecto **el primero** de la lista que ha seleccionado el usuario	Posem com a llengua per defecte la primera llista que l'usuari ha seleccionat	We put as a language by default the first list that the user has selected
Llamamos **las** funciones para **printar** el error en el formulario si **nunca se han llamado**	Anomenem les funcions error d'impressió en el formulari si mai han anomenat	We call the functions print error on the form if they have never called

***ST* en castellà amb preedició**	**TA en català**	**TA en anglès**
Imágenes	Imatges	Images
Comprobar que **hay** texto en el campo nombre	Comprovar aquest text en el nom del camp	Check that text in the field name
Usamos una variable en vez del return **porque** la función promise tarda **más** en retornar el resultado y **producía un** error al comprobarlo en el submit	Fem servir una variable en comptes el retorn perquè la funció promesa pren més temps per tornar el resultat i produeix una comprovació d'error en les sotmeten	We use a variable instead of the return because the promise function takes longer to return the result and produced an error check on the submit
Ponemos como idioma por defecto el **primer idioma** de la lista que ha seleccionado el usuario	Posem com a llengua per defecte la primera llengua a la llista que l'usuari ha seleccionat	We put as a language by default the first language in the list that the user has selected
Llamamos **a** las funciones para **mostrar por pantalla** el error en el formulario si nunca **antes** se han llamado **las funciones**	Anomenem les funcions per mostrar l'error en el formulari a pantalla si mai han estat anomenats funcions	We call the functions to display the error in the form on screen if functions have never before been called

9.7. Resultat de la TA en anglès abans i després de la preedició

***ST* en anglès sense preedició**	**TA en català**	**TA en castellà**
Choose the buttons to show **on bar**	Triar els botons per mostrar en el bar	Elija los botones Mostrar en la barra
function to change **html** view	funció per canviar la vista d'html	función para cambiar a vista html
Check if **url user** exists	Comprovar si existeix l'usuari url	Compruebe si existe usuario url
Send new password	Enviar contrasenya nova	Enviar nueva contraseña
Check **new password length**	Comprovi la llargada de contrasenya nova	Compruebe la longitud de contraseña nueva

***ST* en anglès amb preedició**	**TA en català**	**TA en castellà**
Choose the buttons to show on **the** bar	Triar per mostrar a la barra de botons	Elija los botones a mostrar en la barra de
A function to change **the HTML** view	Una funció per canviar la vista d'HTML	Una función para cambiar la vista HTML
Check if **the URL of the** user exists	Comprovar si existeix l'adreça URL de l'usuari	Comprobar si existe la URL del usuario
Send **a** new password	Enviar una nova contrasenya	Enviar una nueva contraseña
Check **the length of the new password**	Comprovi la llargada de la contrasenya nova	Comprobar la longitud de la nueva contraseña

9.8. Pseudocodi

6.1.1. Obtenció de l'idioma i la variant geogràfica dels comentaris

```
create array language_codes_ISO639_array
language_codes_ISO639_array =
       column = 'language_name'
       column = 'language_code'
content of language_codes_ISO639_array = input
(C:\User\Documents\project\ISOfiles\ISO639.csv)

create array country_codes_ISO3166_array
country_codes_ISO3166_array =
       column = 'country_name'
       column = 'country_code'
content of country_codes_ISO3166_array = input
(C:\User\Documents\project\ISOfiles\ISO3166.csv)

create source_languages_array
       column = 'language_name'
       column = 'language_code'
       column = 'country_code'

Loop
       print (language_codes_ISO639_array)
       print('Check the list above and type in the language code (two lower
case letters) of the source language you will be translating from: ')
       add user's input to column language_code in source_languages_array
       for row in language_codes_ISO639_array
              if row.language_code == user's input
                     add row.language_name to column language_name in
source_languages_array
       print (country_codes_ISO3166_array)
       print('Check the list above and type in the country code (two upper case
       letters) of the source language you will be translating from: ')
       add user's input to column country_code in source_languages_array
       print ('Any other language(Y/N)?')
       if answer == 'N'
              break and continue

create target_languages_array
       column = 'language_code'
       column = 'country_code'

Loop
       print (language_codes_ISO639_array)
       print('Check the list above and type in the language code (two lower
       case letters) of the target language you will be translating to: ')
       add user's input to column 'language_code' in target_languages_array
       print (country_codes_ISO3166_array)
       print('Check the list above and type in the country code (two upper case
       letters) of the target language you will be translating to: ')
       add user's input to column 'country_code' in target_languages_array
       print ('Any other language(Y/N)?')
       if answer == 'N'
              break and continue
```

6.1.2. Anàlisi de l'extensió i la codificació dels fitxers i extracció dels comentaris del codi

```
create array of lists comments_api_array
comments_api_array =
      column = 'commentID_col'
      column = 'comment_col'
      column = 'language_col'

create list ST_files_url
commentID = 0

Loop
      print('Enter the URL address of the ST file and press RETURN: ')
      ST_file = input (url)
      file_ext = ST_file extension
      add ST_file to ST_files_url

      if file_ext == 'php' or file_ext == 'js'
            comment_opening_typeA = "//"
            comment_ending_typeA = "\n"
            comment_opening_typeB = "/*"
            comment_ending_typeB = "*/"
      if file_ext == html
            comment_opening_typeA = "<!--"
            comment_ending_typeA = "-->"
      if file_ext == css
            comment_opening_typeA = "/*"
            comment_ending_typeA = "*/"


      for line in ST_file
            commentID = commentID + 1
            if line contains comment_opening_typeA
                  comment = line from comment_opening_typeA to
                  comment_ending_typeA
                  add comment to comment_col in comments_api_array
                  write '$$' + commentID + '$$' after
                  comment_opening_typeA in ST_file
                  add '$$' + commentID + '$$' to commentID_col in
                  comments_api_array
            else if line contains comment_opening_typeB
                  comment = line from comment_opening_typeB to
                  comment_ending_typeB
                  add comment to comment_col in comments_api_array
                  write '$$' + commentID + '$$' after
                  comment_opening_typeB in ST_file
                  add '$$' + commentID + '$$' to commentID_col in
                  comments_api_array

      print ('Do you have any more files you would like to translate(Y/N):
      ')
      if answer == 'N'
            break and continue
```

6.1.3. API de reconeixement d'idioma

```
print ('Are the comments in your ST written in more than one
language(Y/N)? ')
if answer == 'Y'
      for row in comments_api_array
            output row.comment_col to API
            API_languageName = input API
            for language in source_languages_array
                  if language.language_name == API_languageName
                        row.language_col in comments_api_array =
language.language_code + '-' + language.country_code

save comments_api_array to ('comments_api_results.csv') to
C:\User\Documents\project\translationFiles\
```

6.1.4. Obtenció dels fitxers de treball en format CSV

```
print('Enter the URL address of the file comments_api_results.csv and
press RETURN: ')

create array of lists comments_api_array
comments_api_array =
      column = 'commentID_col'
      column = 'comment_col'
      column = 'language_col'
      comments_api_array = input (url)

create array comments_languages_array

for row in source_languages_array
      add new column with name row.language_code + '-' +
row.country_code

for row in comments_api_array
      add row.commentID_col and row.comment_col and language_col to
      comments_languages_array in column row.language_col

for column in comments_languages_array
      save column to ('comments_'+ column name + '.csv') to
      C:\User\Documents\project\translationFiles\
```

6.1.7. Substitució del ST pel TT

```
create array TT_comments_array
      column = 'commentID_col'
      column = 'comment_col'
      column = 'language_col'

Loop
      print('Enter the URL address of the TT file (postedited or translated)
      and press RETURN: ')
      TT_file = input (url)
      for row in TT_file
            add row.commentID_col and row.comment_col and language_col to
            TT_comments_array
      print ('Do you have any other TT files (postedited or translated)(Y/N)?
            Note that if you have translated your Source Text (comments) to
            different target languages, when introducing all your Target Text
            files (translated comments), comments in each language will be
            placed one after another in the code.')
      if answer == 'N'
            break and continue

sort TT_comments_array by language_col

for url in ST_files_url
      ST_file = input (url)
      file_ext = ST_file extension

      if file_ext == 'php' or file_ext == 'js'
            comment_opening_typeA = "//"
            comment_ending_typeA = "\n"
            comment_opening_typeB = "/*"
            comment_ending_typeB = "*/"
      if file_ext == html
            comment_opening_typeA = "<!--"
            comment_ending_typeA = "-->"
      if file_ext == css
            comment_opening_typeA = "/*"
            comment_ending_typeA = "*/"

      for line in ST_file
            if line contains comment_opening_typeA
                  commentID = extract commentID from comment
                  delete line in ST_file
                  for row in TT_comments_array
                        if row.commentID_col == commentID
                              add comment_opening_typeA + row.language_col +
                              ': ' + comment + comment_ending_typeA to line
            else if line contains comment_opening_typeB
                  commentID = extract commentID from comment
                  delete line in ST_file
                  for row in TT_comments_array
                        if row.commentID_col == commentID
                              add comment_opening_typeB + row.language_col +
                              ': ' + comment + comment_ending_typeB to line

save ST_file to ('TT_' + url.name + file_ext) to
C:\User\Documents\project\translationFiles\ with character encoding file_cod
```