Universidade Federal
do Rio de Janeiro
Escola Politécnica

INTERFACE GRÁFICA PARA USUÁRIOS DE LIBRAS

Jônathan Elias Sousa da Costa

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Antônio Cláudio Gómez de Sousa

Rio de Janeiro

Março de 2018

INTERFACE GRÁFICA PARA USUÁRIOS DE LIBRAS

Jônathan Elias Sousa da Costa

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO

Autor:

Jônathan Elias Sousa da Costa

Orientador:

Prof. Antônio Cláudio Gómez de Sousa, D. Sc.

Examinador:

Prof. Henrique Luiz Cukierman, D.Sc.

Examinador:

Fernando Gonçalves Severo, M. Sc.

RIO DE JANEIRO – RJ, BRASIL

MARÇO DE 2018

Costa, Jônathan Elias Sousa da

Interface Gráfica para Usuários de Libras/ Jônathan Elias Sousa da Costa – Rio de Janeiro: UFRJ/ Escola Politécnica, 2018.

XIV, 48 p.: il.; 29,7 cm.

Orientador: Antônio Cláudio Gómez de Sousa.

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2018.

Referências Bibliográficas: p. 43-47.

1. Software de Acessibilidade. 2. Autonomia de surdos. 3. Tecnologias de Aprendizado e Convivência. I. Sousa, Antônio Cláudio Gómez. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Computação e Informação. III. Interface Gráfica para Usuários de Libras.

Declaração de Autoria e de Direitos

Eu, *Jônathan Elias Sousa da Costa* CPF *152.816.737-69*, autor da monografia *Interface Gráfica para Usuários de Libras*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.

Jônathan Elias Sousa da Costa

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica – Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro – RJ CEP 21949-900

DEDICATÓRIA

A meu pai.

Por sê-lo.

AGRADECIMENTO

Ao Eterno Deus, por seu amor e fidelidade, sem a qual não concluiria esta caminhada.

A meu pai, João Elias Souza da Costa, por seu carinho, sustento e orientação durante toda minha vida.

A todos os meus familiares pelo apoio e colaboração.

Aos professores deste curso, por sua excelência e dedicação.

Ao meu orientador, por sua paciência e compreensão.

Aos coordenadores e colegas do Laboratório de Informática para Educação e do Laboratório de Informática e Sociedade, da UFRJ, que apoiaram este trabalho do início ao fim.

Aos membros da comunidade surda do Rio de Janeiro que participaram neste trabalho, pelo incentivo e troca de experiências. Em especial, ao Leandro Narciso dos Santos, *In memoriam*, por sua alegria e colaboração marcantes neste trabalho.

RESUMO

Este trabalho aborda a trajetória de desenvolvimento de uma ferramenta computacional voltada para pessoas surdas, o LIBRASOffice. O aplicativo se propõe a auxiliar um usuário que não domina o português (língua escrita oficial no Brasil), mas sim a língua brasileira de sinais (ou ainda, a Libras), a utilizar o LibreOffice - uma suíte de escritório livre e de código aberto - de forma autônoma. Para isso, as opções disponíveis na interface gráfica do LibreOffice são apresentadas para os usuários também na língua de sinais Libras, via mecanismo de adaptação desenvolvido em Java e C++.

Palavras-Chave: surdo, acessibilidade, software, autonomia, libreoffice, libras.

ABSTRACT

This work discusses the development's path of a computational tool for deaf people, LIBRASOffice. The application proposes to help a user who does not speak portuguese (official written language in Brazil), but rather the brazilian sign language (or Libras), to use the LibreOffice - a free and open source office suite - autonomously. For this, the options available in the graphical interface of LibreOffice are presented to users also in the Libras sign language, through an adaptation mechanism developed in Java and C ++.

Keywords: deaf, accessibility, software, autonomy, libreoffice, pounds.

SIGLAS

UFRJ – Universidade Federal do Rio de Janeiro

COPPE - Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

LIPE – Laboratório De Informática Para Educação

LABIS – Laboratório De Informática e Sociedade

COPPETEC - Fundação Coordenação de Projetos, Pesquisas e Estudos Tecnológicos

LAViD - Laboratório de Aplicações de Vídeo Digital

UFPB - Universidade Federal da Paraíba

PIBIC-EM - Programa Institucional de Bolsas de Iniciação Científica para o Ensino Médio

LO - LibreOffice

LASO - LIBRASOffice

GUI - *Graphical User Interface*

UML - *Unified Modelling Language*

VCL - *Visual Components Library*

UNO - *Universal Network Objects*

COM - *Component Object Model*

DCOM - *Distributed Component Object Model*

IDL - *Interface Definition File*

OLE - *Object Linking and Embedding*

CLI - *Common Language Infrastructure*

VB.NET - *Visual Basic .NET*

API - *Application Programming Interface*

IDE - *Integrated Development Environment*

JDK - *Java Development Kit*

GIF - *Graphics Interchange Format*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 – Contextualização

O Brasil possui, atualmente, 2,1 milhões de deficientes auditivos severos [2]. Destes, 80% possuem dificuldades para ler e escrever [3]. Vivemos, desde 2002, em um país bilíngue [5], que dispõe de duas línguas oficiais: o português e a Libras. A Libras apoia-se em símbolos gestuais-visuais, os sinais, para significar os conceitos que em português são oralizados ou escritos. É a primeira opção de língua para os surdos, que, ao não disporem de processos cognitivos oriundos da audição, identificam com mais naturalidade conceitos comunicados visualmente, como os gestos [40]. Entretanto, as atuais produções de software seguem massivamente padrões de GUI pensadas e projetadas para serem confortavelmente utilizados apenas pelos usuários fluentes na linguagem escrita. Tais softwares geralmente se apresentam em janelas onde as interações com o usuário somente se dão através da decodificação de itens lexicais pertencentes ao português escrito, por exemplo. Desta forma, se um usuário apresenta-se com dificuldades na leitura e escrita do português, dificilmente terá uma experiência autônoma e proveitosa de tais softwares, mesmo que domine a segunda língua oficial de nosso país, a Libras. É nesta situação que geralmente se encontram os surdos. Seu aprendizado no uso de softwares demanda um grande esforço de memorização de palavras em português a fim de associá-las com os sinais da Libras (quando estes existirem) que correspondem à ação a ser executada pelo software no computador.

Neste contexto, o desenvolvimento de softwares que se apresentem aos surdos com interações gestuais-visuais, ao invés de apenas textuais, revela-se como demanda útil à sociedade brasileira.

1.2 – Motivações

A ideia por trás do projeto deu-se no Laboratório de Informática para Educação da UFRJ (LIpE/UFRJ), no decorrer de um curso de extensão sobre conhecimentos informáticos promovido pelo LIpE para 18 funcionários da Fundação COPPETEC, no segundo semestre de 2015. Dentre os funcionários encontravam-se 7 deficientes auditivos, que apresentaram maior dificuldade na apropriação do conteúdo do que os demais. Para os alunos surdos, a presença de intérprete especializado em sinais informáticos era estritamente necessária. Sendo assim, e de acordo com a metodologia participativa [6] empregada no LIpE (que aproxima instrutor e aluno), seriam necessários tantos intérpretes quanto alunos, situação impraticável que tornou-se o motivador principal para este trabalho.

Desde então, a fim de concretizar o objetivo deste trabalho, reuniram-se estudantes da disciplina de graduação "Computadores e Sociedade" do curso de Engenharia de Computação e Informação (ECI) da Escola Politécnica e da linha de pesquisa em Informática e Sociedade (IS) do Programa de Engenharias de Sistemas e Computação (PESC) da COPPE, que, em conjunto com membros do LIpE (Laboratórios de Informática e Sociedade), estabeleceram uma parceria que logrou o comprometimento de membros da comunidade surda do Rio de Janeiro e de Niterói. Ao longo do biênio 2016-2017, estudantes de graduação em ECI e da pós-graduação do PESC, membros do LIpE e do LabIS, bolsistas PIBIC-EM (estudantes surdos do Ensino Médio do projeto PIBIC-EM da IS/PESC), intérpretes e instrutores surdos voluntários das secretarias municipais de educação dos municípios do Rio de Janeiro e Niterói engajaram-se no desenvolvimento do protótipo de software LIBRASOffice. É importante ressaltar que o desenvolvimento do LIBRASOffice beneficiará potencialmente 10 milhões de pessoas (segundo censo de 2010 realizado pelo IBGE, 9,7 milhões de pessoas possuem algum grau de deficiência auditiva).

1.3 – Objetivos

O objetivo deste trabalho é desenvolver um software de acessibilidade que auxilie usuários de computador deficientes auditivos a utilizarem a suíte de escritório de

código-aberto LibreOffice de forma mais autônoma. O software possibilitará a usuários que compreendem bem a Libras, e pouco o português, uma melhor compreensão das opções disponíveis na interface gráfica do LibreOffice, através da introdução de elementos de ajuda na língua Libras, associados a tais opções. Ao utilizarem o LIBRASOffice, os surdos seriam duplamente beneficiados, pois encontrariam uma curva de aprendizagem do LibreOffice mais íngreme (aprendizado acelerado por uma GUI mais acessível) além de disporem de maior domínio e autonomia para utilizarem mais intensamente o computador como ferramenta produtiva, nas esferas pessoal, acadêmica e profissional. Um instrutor que conhece pouco a Libras pode ensinar um aluno que conhece pouco o português, com ajuda da interface adaptada bilíngue.

Para tanto, será estabelecido um processo de adaptação de GUIs que apresentam conteúdos predominantemente em português escrito para GUIs que apresentem o mesmo conteúdo em português e na língua brasileira de sinais (Libras). A adaptação proposta será aplicada nos softwares da suíte de escritório LibreOffice [4] e consiste na inserção de elementos gráficos de acessibilidade além daqueles apresentados por este software. Não serão realizadas modificações estruturais na GUI do LibreOffice.

1.4 – Metodologia

Este trabalho adotou o modelo de desenvolvimento em fases (ver Figura 1.1), mediante processo iterativo e incremental de desenvolvimento de software, apoiado em um planejamento adaptativo, devidamente documentado em Linguagem Unificada de Modelagem (UML, do inglês *Unified Modelling Language*).

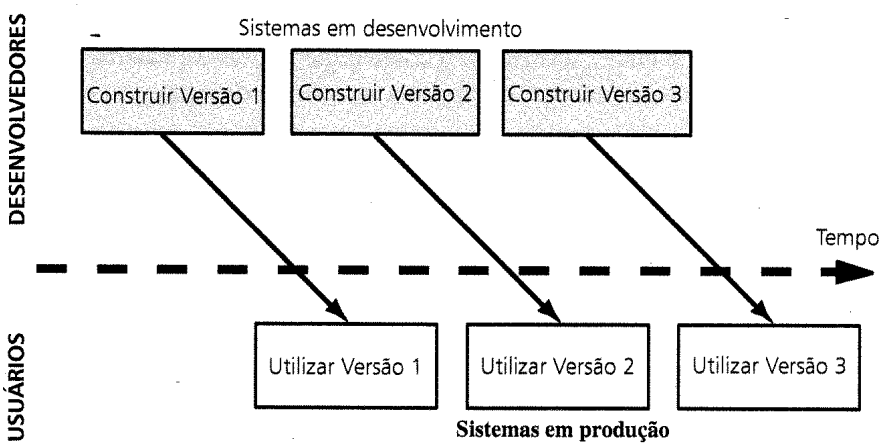


Figura 1.1 – O modelo de desenvolvimento em fases. [20]

No processo iterativo e incremental de desenvolvimento, encontramos uma abordagem mais sofisticada para a produção de *softwares*, apropriada a projetos suscetíveis a significativas alterações de requisitos após um levantamento inicial destes (revolução de requisitos): a cada iteração verifica-se a existência, ou não, de novo requisito a ser implementado, ou seja, um novo incremento. Quando utilizado, o processo iterativo e incremental geralmente demanda um planejamento adaptativo. [1] A Figura 1.2 ilustra a dinâmica do processo iterativo e incremental de desenvolvimento: a cada iteração, repetimos as etapas do famoso modelo em cascata, a fim de melhor acomodar novos requisitos, uma vez que o software inicialmente planejado poderá ser substancialmente alterado:

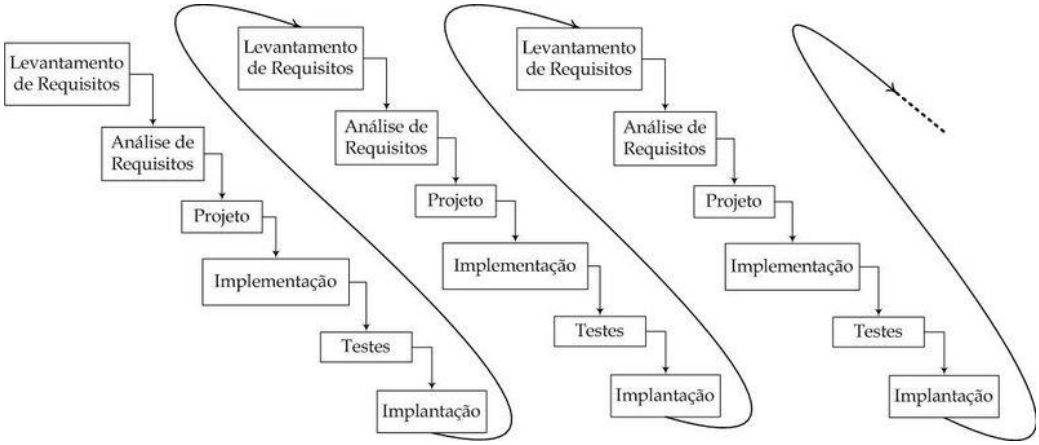


Figura 1.2 – Fluxograma de um processo iterativo de desenvolvimento. [21]

Por aplicar um processo iterativo de desenvolvimento de software, percorremos uma fase exploratória antes do início do desenvolvimento, visando a obtenção em alto-nível dos requisitos, a fim de dividi-los entre as iterações que se seguiram. Ao final de cada iteração, o software adquiriu maturidade suficiente para as avaliações periódicas realizadas pelos usuários finais durante o desenvolvimento, cujos feedbacks garantiram um produto final de melhor qualidade. [1]

O planejamento adaptativo foi aplicado no projeto a fim de acomodar as revoluções de requisitos (alterações nos requisitos iniciais estabelecidos). Desta forma, o projeto não utilizou um planejamento preditivo (cuja qualidade estaria atrelada a um levantamento de requisitos suficientemente preciso), mas sim estratégias de

planejamento suscetíveis a alterações constantes nos requisitos, resultando em um desenvolvimento imprevisível, mas controlado, a fim de gerar o melhor software possível dentro do cronograma. [1]

Os conteúdos em Libras utilizados neste trabalho foram inicialmente extraídos do software aberto VLibras[1]. Posteriormente, estes conteúdos foram produzidos por membros da comunidade surda do Rio de Janeiro interessados no projeto, mediante parcerias intermediadas pelo Laboratório de Informática e Sociedade da Coppe/UFRJ.

1.5 – Descrição

Este trabalho foi organizado em quatro capítulos da seguinte forma:

- No Capítulo 1, é exposta uma breve introdução sobre o projeto, o objetivo e as motivações;
- O Capítulo 2 aborda em detalhes os métodos utilizados para o desenvolvimento e a documentação do software, bem como as tecnologias utilizadas, os ambientes e suas ferramentas.
- No Capítulo 3, descrevemos como deu-se o processo iterativo e incremental de desenvolvimento, abordando as principais diferenças entre cada versão lançada.
- Por fim, no Capítulo 4, apresentamos os resultados conceituais e práticos do projeto, avaliações, aprendizados e propostas futuras.

[1] A Suíte VLibras consiste em um conjunto de ferramentas computacionais de código aberto, responsável por traduzir conteúdos digitais (texto, áudio e vídeo) para a Língua Brasileira de Sinais - LIBRAS, tornando computadores, dispositivos móveis e plataformas Web acessíveis para pessoas surdas.[24]

Capítulo 2

O projeto

2.1 – Proposta de solução

A fim de implementar a acessibilidade para surdos em aplicativos de escritório, utilizando como base o software aberto LibreOffice, este projeto propõe-se a:

1. Exibir descrições em Libras para as diversas opções disponíveis na interface gráfica do LibreOffice;
2. Disponibilizar assistentes em Libras para atividades de maior grau de dificuldade, como a inserção de fórmulas em planilhas.

Para a primeira tarefa, pretendemos adaptar a GUI do LibreOffice, pois ela não lida nativamente com conteúdos em linguagens de sinais. Precisaremos, portanto, alterar parcialmente o funcionamento de alguns dos módulos internos do LibreOffice, relacionados à construção de sua GUI, a fim de implementarmos pontos de adaptação, como a detecção de foque e desfoque de componentes gráficos pelo movimento do mouse do usuário. Uma vez instalados tais pontos, poderemos exibir o conteúdo adaptado em uma GUI própria e externa, independente da estrutura gráfica do LibreOffice (inadequada para os nossos propósitos).

Para a segunda tarefa pretendemos trocar mensagens com o LibreOffice através de sua API, a fim de que possamos externamente, e de acordo com nossas necessidades, instruí-lo a realizar procedimentos geralmente disparados de forma direta na sua GUI, como a inserção de fórmulas e o enfoque ou desfoque de componentes gráficos.

2.2 – Estrutura do LibreOffice

O LibreOffice é um projeto de software de código aberto que implementa uma suíte de aplicativos de escritório, dentre os quais destacam-se: Writer (processador de texto), Calc (processador de planilhas eletrônicas) e Impress (editor de slides) [4].

O LibreOffice é, portanto, um software consideravelmente grande, possuindo mais de 9 milhões de linhas de código [7] que implementam diversas funcionalidades, decompostas em dezenas de módulos (ou ainda, componentes) coesos [39].

Para fins deste trabalho, destacam-se alguns módulos. O primeiro deles é denominado *Visual Components Library* (vcl) [10], módulo responsável pela abstração de *widgets*[2] nativos do sistema operacional como, por exemplo, janelas, botões, controles, seletores de arquivos, etc. Tais abstrações constituem uma biblioteca de componentes gráficos própria ao LibreOffice, possibilitando modificações conceituais na GUI que reflitam em todas as plataformas suportadas. O vcl também implementa rotinas de renderização de baixo-nível, chamadas quando são necessárias interações diretas com o dispositivo de saída gráfica (por exemplo, ao inserirmos um complexo gráfico 3D em uma planilha).

Outro módulo que se destaca neste trabalho é denominado *desktop* [11], que implementa o objeto controlador *XDesktop* [12]. Tal objeto é compartilhado entre todas as instâncias em execução do LibreOffice, sendo central para o despacho de eventos relacionados à construção e destruição de *widgets*. O módulo *desktop* é, portanto, responsável pela implementação de rotinas de inicialização referentes a diversos outros módulos que afetam a GUI, como o *vcl*.

As implementações dos principais aplicativos de escritório integrantes da suíte também são separadas em módulos: o módulo *sc* guarda o código específico a uma instância da planilha eletrônica Calc; o módulo *sw* implementa detalhes do processador de texto Writer; já no módulo *sd* encontram-se implementações específicas às aplicações Impress e Draw [13].

2.3 – API do LibreOffice

O LibreOffice possui uma *Application Programming Interface* (ou API) bem definida, oferecida sob um modelo de componentes baseado em interface desenvolvido especialmente para o LibreOffice - e sob o qual ele é construído - denominado *Universal Network Objects* (UNO) [15]. O modelo UNO, semelhantemente a outros modelos de componentes (destacando-se Microsoft COM/DCOM [14]), permite a um

[2] *Widget* é um componente gráfico disponível em uma biblioteca para construção de GUIs [9]

componente ser instanciado em diversas linguagens diferentes, uma vez que especifica, em linguagem de máquina, as interfaces disponíveis para determinado componente. Sendo assim, quando desejamos obter determinado comportamento dentre os vários possíveis de um objeto componente, basta solicitar ao objeto a interface apropriada. Como tais interfaces são descritas em linguagem de máquina, armazenadas em um *Interface Definition File* (IDL), torna-se possível que objetos componentes instanciados em processos distintos, e possivelmente implementados em linguagens diferentes, comuniquem-se em tempo de execução, mediante algum mecanismo de intercomunicação de processos, como *pipe* ou *sockets*, por exemplo (ver Figura 2.2). Isto é factível, pois as mensagens trocadas entre os processos, através da interface UNO apropriada, serão agnósticas à sintaxe de linguagens de alto nível, sendo codificadas e decodificadas diretamente em representação binária. Esta comunicação poderá se dar, portanto, entre quaisquer pares de processos implementados por linguagens em que seja possível "linkar", diretamente (*bindings*) ou indiretamente (adaptadores), a biblioteca UNO de interfaces de baixo-nível (ver Figura 2.1). Tais processos irão constituir um ambiente de execução UNO [14], que pode ser acessado nas linguagens C++, Java, Python, BASIC além de ser compatível com os padrões Microsoft OLE e Microsoft CLI, o que permite seu uso nas linguagens C# e VB.net [16].

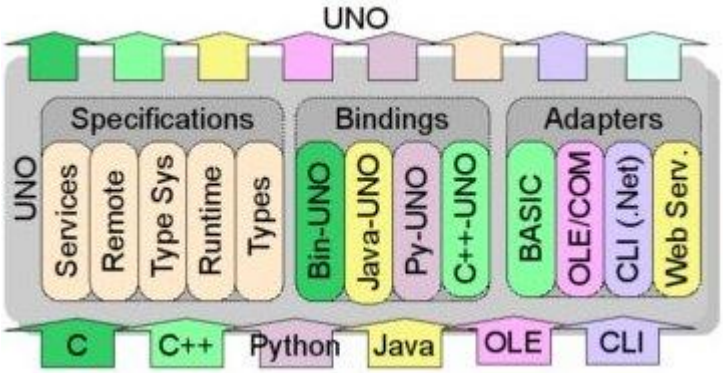


Figura 2.1 – Arquitetura do modelo de componentes UNO. [17]

Para fins de elucidação do funcionamento da API UNO, suponhamos que um programa A, de autoria própria, implementado em Java, deseje inserir uma fórmula em uma célula específica da planilha atualmente ativa na instância em execução do LibreOffice Calc, majoritariamente implementada em C++. Os seguintes passos elucidam a codificação que implementará tal funcionalidade no programa A [18]:

1. Importamos, em A, a biblioteca de interfaces UNO para Java.
2. Instanciamos um socket UNO que permitirá a comunicação inter-processos entre o programa A, quando em execução, e o LibreOffice Calc [19]. Admitimos que o Calc foi instanciado com os argumentos necessários para a criação de um *socket* listener que aguarda conexões entrantes de processo remotos.
3. Instanciamos um objeto "desk" de classe UNO Desktop, que referencia o contexto de processo do Calc.
4. Solicitamos que o objeto "desk", mediante chamada de método apropriado, que retorne uma referência ao componente gráfico atualmente em foco na GUI do Calc. Armazenamos esta referência no objeto "comp" de classe UNO Component.
5. Solicitamos ao objeto "comp" uma interface que nos permita acessar seu componente controlador.
6. Através da nova interface, solicitamos ao objeto "comp" uma referência ao seu controlador. Armazenamos esta referência no objeto "control" de classe UNO Controller.
7. Solicitamos ao objeto "control" uma interface que nos permita acessar o componente que referencia o workbook atual (conjunto de planilhas).
8. Através da nova interface, solicitamos ao objeto "control" uma referência ao workbook. Armazenamos esta referência no objeto "spreadview" de classe UNO SpreadsheetView.
9. Solicitamos ao objeto "spreadview" uma referência à planilha atualmente em foco na GUI. Armazenamos esta referência no objeto "sheet" de classe UNO Spreadsheet.
10. Solicitamos ao objeto "sheet" uma referência à célula desejada. Armazenamos esta referência no objeto "cell" de classe UNO Cell.
11. Por fim, inserimos a fórmula na célula desejada mediante chamada de método apropriado disponível no objeto "cell".

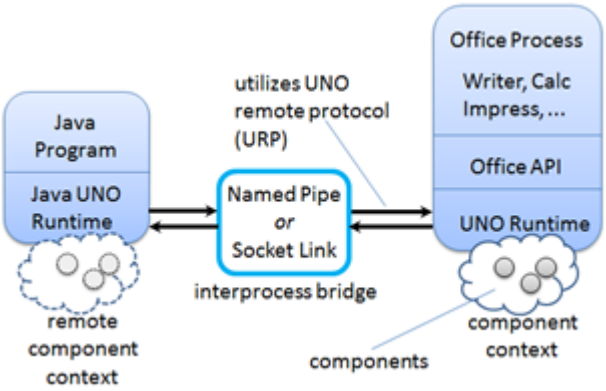


Figura 2.2 – Fluxograma de comunicação inter-processos UNO. [19]

2.4 – Ambientes e ferramentas

Até sua terceira versão, o LIBRASOffice foi codificado e compilado no ambiente de desenvolvimento integrado NetBeans IDE 8.1 *[22]*. A plataforma Java utilizada foi a OpenJDK *(*1.7) [23]. O desenvolvimento deu-se majoritariamente em ambiente Linux e baseou-se no LibreOffice 4.2. Os sinais Libras eram obtidos do VLibras, mediante procedimento de gravação em vídeo do conteúdo reproduzido em sua janela, que recebendo como entrada uma frase em português a traduzia animadamente para Libras. Inicialmente não foram feitas modificações no código-fonte do LibreOffice.

A partir de sua terceira versão, o LIBRASOffice passou a ser majoritariamente desenvolvido sob plataforma Windows, na versão 10, uma vez que inexistia uma versão do software para este sistema operacional, à época requisitada pelos usuários finais. Desde então, temos utilizado, tanto em sistemas Windows quanto Linux, o ambiente de desenvolvimento integrado Eclipse [25], na versão Neon, sob a plataforma Java Oracle JDK (1.8.141) [26]. Para eventuais modificações do código-fonte C++ do LibreOffice, utilizamos o editor avançado de texto Notepad++ [27]. A GUI do LIBRASOffice é totalmente implementada sob Java Swing [28] desde a primeira versão. A documentação do software tem sido realizada com o apoio da ferramenta JavaDoc [29], disponibilizada pela plataforma Java para programas codificados em linguagem de mesmo nome, para geração automática de documentação externa a partir da interna.

O LIBRASOffice possui controle de versionamento desde sua segunda versão, sendo este realizado pela ferramenta Git [30], mediante conhecido ecossistema de versionamento de código “na nuvem”, denominado GitHub [31].

Atualmente o código do LIBRASOffice encontra-se decomposto em três repositórios registrados no GitHub, organizados da seguinte forma:

- Repositório "LASOBack" [32]: guarda um clone do repositório “core” do LibreOffice (por vezes abreviado LO) que está sendo atualmente utilizado pelo LIBRASOffice (acrônimo LASO). Em cima do clone são realizadas modificações em alguns módulos do LO, a nível de código-fonte C++, para o correto funcionamento do LIBRASOffice. Procuramos manter este repositório sempre atualizado, ou seja, correspondendo sempre a um clone da última versão estável do LibreOffice.
- Repositório "LASOFront" [33]: guarda todo o código-fonte Java do LIBRASOffice que efetivemente implementa a interação visual em Libras com o LibreOffice, utilizando modificações implementadas no LASOBack para algumas funcionalidades e a API UNO do LibreOffice para outras.
- Repositório “LIBRASOffice” [34]: repositório principal que armazena página pública de boas-vindas ao projeto, bem como a documentação externa do LIBRASOffice. Este repositório também guarda scripts de linha de comando que automatizam tarefas relacionadas ao processo de construção dos executáveis do LIBRASOffice.

2.5 - Documentação

A documentação interna do LIBRASOffice (aquela redigida diretamente no código-fonte) foi incrementalmente construída desde a primeira versão, sendo sempre formatada de acordo com o padrão JavaDoc, em arquivos de códigos-fonte Java, e de maneira semelhante em arquivos de código-fonte em outras linguagens. Foram devidamente comentadas as classes, métodos não triviais, atributos não triviais e instruções de bibliotecas externas que apresentaram certo grau de ilegibilidade que dificultava sua direta interpretação. Como uma convenção geral, modificações em códigos-fonte já existentes foram imediatamente precedidas pela linha “ADD LIBRAS”

e imediatamente sucedidas pela linha “END LIBRAS”, com ambas as linhas devidamente comentadas de acordo com a linguagem descritora do código-fonte modificado. Esta pequena convenção nos permitiu manter grande rastreabilidade das mudanças feitas no código-fonte altamente modularizado do LibreOffice.

A documentação externa foi parcialmente gerada a partir da documentação interna, uma vez que o JavaDoc é capaz de gerar páginas HTML devidamente organizadas de acordo com a hierarquia de pacotes, classes e métodos codificados, que apresentam de maneira mais direta e elucidativa, à parte do código-fonte, grande parte das especificações internas do software implementado em Java. Os arquivos gerados pelo JavaDoc encontram-se em diretório apropriado no repositório LASOFront, armazenado no GitHub. Ainda no GitHub encontram-se páginas públicas em estilo Wiki que elucidam diversos procedimentos relacionados à compilação, instalação e modificação do LIBRASOffice em diversas plataformas.

Complementam ainda a documentação externa, uma série de casos de uso e diagrama de classes, redigidos em *Unified Modelling Language* (UML) [1]. Tais artefatos de documentação serão apresentados no decorrer do capítulo 3.

Capítulo 3

Desenvolvimento

3.1 – Versões sucessivas

O LIBRASOffice é desenvolvido sob uma livre implementação do processo de desenvolvimento iterativo e incremental, que aqui denominaremos versões sucessivas. Conceitualmente, classificamos o modelo de versões sucessivas como um modelo de desenvolvimento em fases iterativo e incremental apoiado em planejamento adaptativo.

Inicialmente o projeto concentrou-se em grande fase exploratória, uma vez que iniciativas semelhantes não foram encontradas. Mais especificamente, embora tivéssemos encontrado diversos softwares tradutores de texto em português para Libras (dentre os quais notam-se VLibras [24], ProDeaf [41] e Rybená [42]), não foram encontrados softwares que adaptassem uma GUI textual já existente para uma GUI "textual-gestual", capaz de interagir com o usuário através da Libras.

Após a determinação da possibilidade tecnológica de implementarmos a primeira iteração, deu-se o efetivo início do processo de desenvolvimento, que seguiu-se com o lançamento de sucessivas versões, as quais ora aprimoraram as funcionalidades já existentes, ora adicionaram novas funcionalidades, ora apresentaram ambos os avanços.

Os requisitos iniciais solicitavam o desenvolvimento de um software que permitisse renderização de *tooltips*[3] em Libras que adaptasse as *tooltips* já presentes na GUI do LibreOffice. Tal software precisaria ser compatível com o LibreOffice 4.2, executado em ambiente Linux. Os pesquisadores do LIpE também solicitaram um protótipo para "uma prova de conceito", a fim de melhor determinar o nível de factibilidade do projeto.

[3] *tooltips* são pequenas janelas de ajuda que apresentam um curto texto descritor de um determinado componente gráfico disponível na GUI, com o objetivo de guiar o usuário na utilização das opções disponíveis em um software [35].

Posteriormente, ainda durante a primeira fase (ou ainda, versão) observamos a introdução de novo requisito: o software deveria ser capaz de apoiar o usuário na inserção de fórmulas em planilhas eletrônicas mediante janela assistente em Libras.

Nas sub-seções seguintes são apresentados os destaques de cada versão, junto à explicações de decisões de projeto tomadas ao longo do desenvolvimento. Especificações mais completas das atuais funcionalidades do projeto, e de suas estruturas de dados, podem ser encontradas na sub-seção relacionada a última versão.

3.2 – Primeira versão

Como sugerido pelos pesquisadores do LIpE, optamos por iniciar o desenvolvimento do LIBRASOffice construindo um protótipo que trabalhasse especificamente com a planilha de cálculo da suíte LibreOffice, o Calc. Dessa forma, poderíamos testar a operacionalidade de um primeiro protótipo de forma geral, pois, segundo os pesquisadores, os usuários surdos dominam com mais facilidade softwares que lidam diretamente com números. Neste caso, o pré-requisito primordial seria o domínio de operações matemáticas básicas, sendo prescindível o domínio da língua escrita.

Inicialmente procuramos pontuar, junto aos pesquisadores, quais seriam os comandos e fórmulas mais utilizados em processadores de planilhas, a fim de selecioná-los para implementação básica já na primeira iteração.

Selecionamos os comandos mais básicos pertencentes à barra padrão do LibreOffice: Novo, Abrir, Salvar, Imprimir, Cortar, Copiar e Colar. Seguimos selecionando os comandos Desfazer e Refazer, devido à importância de tais comandos na produção de documentos complexos.

Selecionamos as fórmulas matemáticas mais básicas: Soma ou Adição (operador +), Subtração (operador -), Multiplicação (fórmula MUL ou operador *) e Divisão (fórmula DIV ou operador /). Dentre estas definimos que, para um primeiro teste com usuários finais, seria imprescindível implementarmos ao menos a operação de adição em nossa interface, permitindo ao usuário somar o conteúdo de duas células.

Nesta fase também definimos que utilizaríamos gráficos animados no formato *Graphics Interchange Format* (GIF) para exibir conteúdos em linguagem de sinais, devido ao baixo custo computacional envolvido em sua renderização, além da

facilidade de suporte deste formato por bibliotecas gráficas para diversas linguagens de programação com suporte a construção de GUIs, como o Java.

Ao pesquisarmos possíveis softwares tradutores de textos em português para Libras, encontramos o projeto VLibras (ver Figura 3.1), iniciativa do LAViD/UFPB (Laboratório de Aplicações de Vídeo Digital – Universidade Federal da Paraíba) [24]. Decidimos por utilizar o dicionário do VLibras e suas animações como a referência de sinais em Libras a serem utilizados no LIBRASOffice em seus primeiros protótipos.

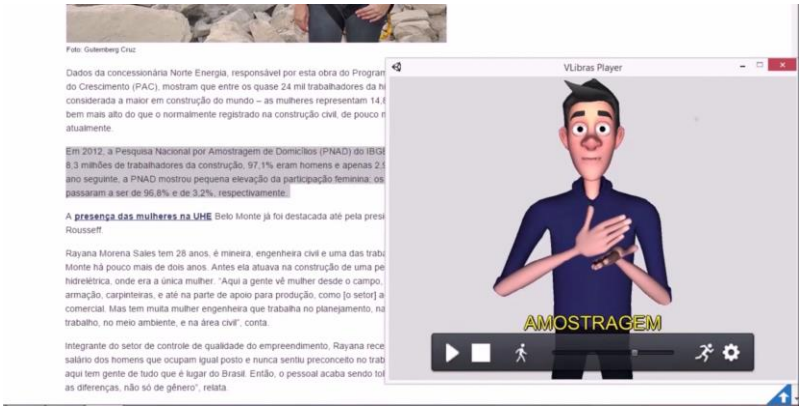


Figura 3.1 – Software “VLibras”, tradutor de português para Libras. [24]

Com os requisitos para a primeira versão já bem definidos, seguimos com o processo de desenvolvimento, definindo um cronograma de 10 semanas para projeto e implementação, de forma que ao fim deste período um protótipo deveria estar pronto para testes no LIpE. Este prazo viria a nortear a escolha do método de implementação. De forma quase que imediata pensamos em basear a implementação do protótipo na modificação do código-fonte do LibreOffice, alterando a forma de exibição das descrições dos comandos: em vez de o texto descritor do comando (exibido ao passar o mouse em cima do botão correspondente) ser exibido em linguagem escrita no interior de um retângulo de fundo amarelo, tal texto seria substituído por um gráfico animado a ser exibido da mesma forma, que descreveria o comando em Libras. Porém, este método de implementação demonstrou-se impraticável à época, uma vez que a estimativa de tempo a ser despendido - no estudo e exploração do código de um grande projeto de software como o LibreOffice - era incompatível com o prazo estipulado pelo cronograma.

Voltamos ao LibreOffice com uma abordagem diferente: decidimos por desenvolver uma extensão externa - interface gráfica totalmente voltada para nossos requisitos - que não estaria limitada à estrutura gráfica implementada na suíte de escritório. Notamos que este método seria o mais viável, devido à robustez da API do LibreOffice e sua estrutura de modelos de componentes baseada em interfaces (UNO). A UNO oferece alta flexibilidade no desenvolvimento de "add-ons", ao permitir o acesso a variadas funções e comandos internos da suíte a partir de diversas linguagens de programação. Para seguirmos com este método foi necessária extensa consulta à documentação da API do OpenOffice/LibreOffice [16].

Escolhemos a linguagem Java para implementarmos nossa solução pois, além do suporte presente na API do LibreOffice, poderíamos contar com sua rica biblioteca multimídia (que manipula gráficos de forma simples e direta) e com a portabilidade de código entre diferentes plataformas de software/hardware.

Seguimos com a definição da estrutura gráfica da interface do primeiro protótipo do LIBRASOffice, que consistia em uma janela auxiliar ao LibreOffice dividida em duas áreas principais (na Figura 3.2, a janela à esquerda): a área inferior, que exibia os comandos implementados na estrutura aba/botão - as abas representavam as categorias de comandos e os botões, os comandos em si - e a área superior, que exibia os gráficos animados capturados do VLibras (descrições em Libras dos comandos). Inicialmente, o LIBRASOffice operou da seguinte forma: bastava o usuário passar o mouse em cima de um botão referente a um comando para que a área superior exibisse a descrição de tal comando. Um clique no botão disparava o comando na instância do LibreOffice Calc inicializada pelo LIBRASOffice.

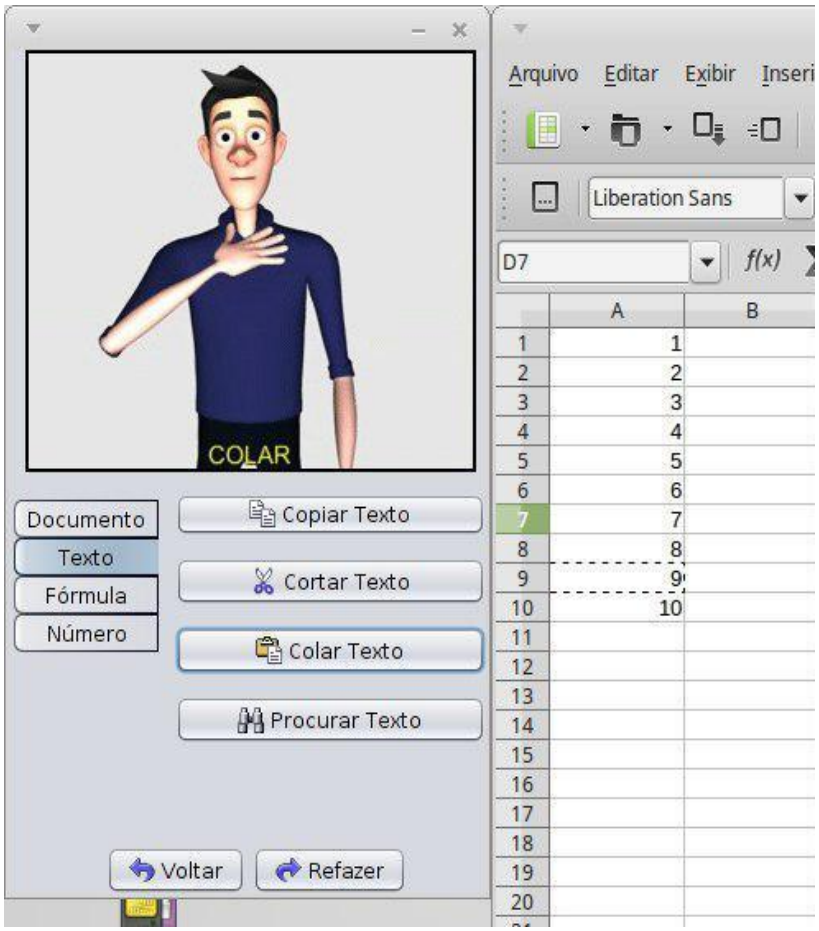


Figura 3.2 – Protótipo completo do LIBRASOffice, 1ª versão.

3.3 – Segunda versão

Após uma avaliação inicial do protótipo pelos pesquisadores do LIpE, percebemos que a forma de navegação inicialmente implementada implicava em problemas de usabilidade, visto que um simples “esbarrão” no mouse poderia alterar a posição do cursor na janela, interrompendo a execução do gráfico animado desejado pelo usuário. Optamos pela seguinte alteração no manuseio da interface: um clique no botão de um comando exibiria a animação correspondente na área superior e dois cliques dispararia tal comando no LibreOffice. Também adicionamos ícones aos botões dos comandos a fim de melhorar o aspecto geral da interface. Lançamos, assim, a segunda versão, que foi utilizado para os primeiros testes com usuários finais.

Após os vários testes realizados, percebemos que os usuários finais (pessoas surdas) destacaram mais a iniciativa que tivemos ao desenvolver a ferramenta junto ao laboratório do que a usabilidade do protótipo desenvolvido, apesar de também

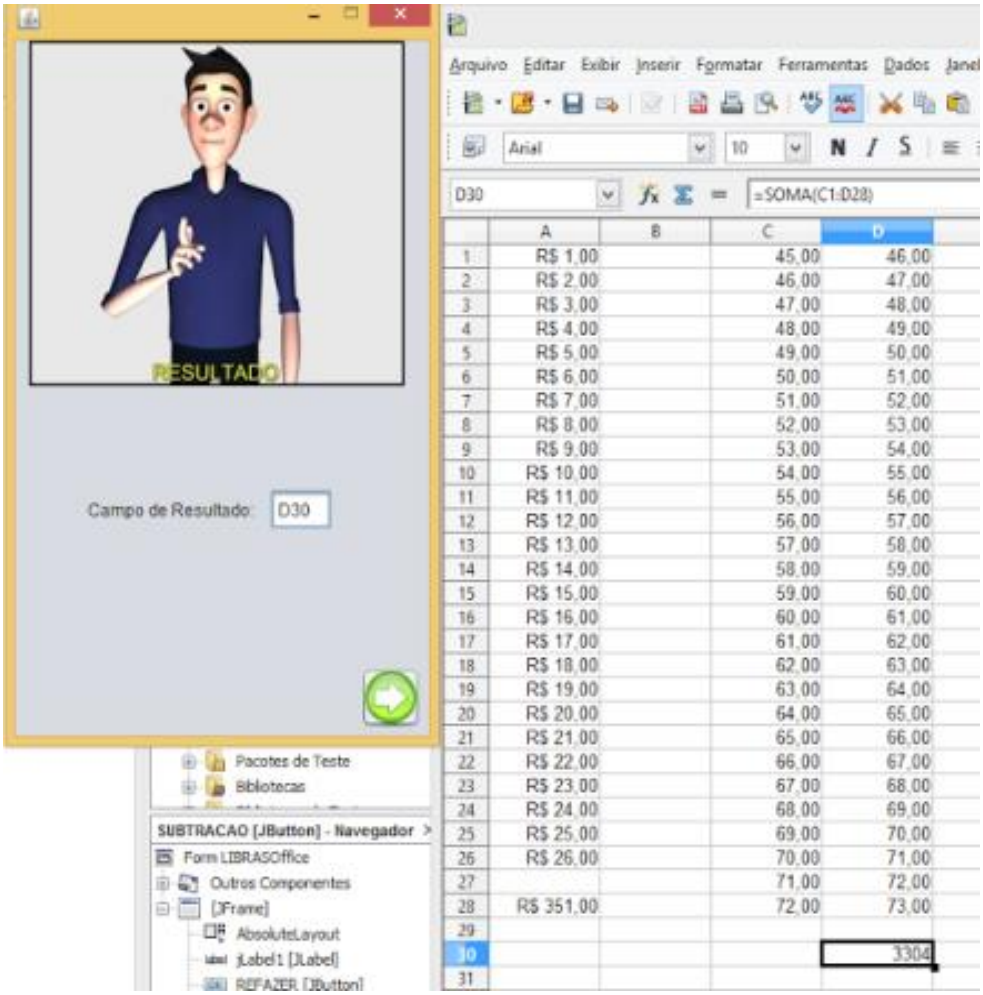
apreciarem, em algum nível, este quesito do projeto. O fato de termos trabalhado em projetar e desenvolver algo feito especificamente para eles e que, além disso, trouxesse independência no uso produtivo do computador a esta categoria de usuários, foi a característica do projeto mais notada por quem o avaliou. Estas avaliações pelos usuários finais estão documentadas em mídia publicamente disponível [43].

O projeto necessitava de muitos aprimoramentos a serem implementados para que alcançássemos uma versão madura e amplamente utilizável. Muitas sugestões foram propostas pelos usuários finais, intérpretes e pesquisadores da área, dentre as quais: correção de alguns sinais utilizados; melhorias no design da interface, a fim de torná-la mais amigável; adição de mais comandos e fórmulas suportados, no caso do LIBRASOffice Calc; suporte ao demais aplicativos da suíte LibreOffice.

3.4 – Terceira versão

A partir da terceira versão, o LIBRASOffice passou a, de fato, disponibilizar um assistente com interface adaptadas para Libras que auxiliava o usuário na inserção de fórmulas em planilhas (ver Figura 3.3). Foi também a partir do terceiro protótipo que iniciamos o suporte aos sistemas operacionais Microsoft Windows (versão 7 ou maior), requisitado pelos usuários finais devido a forte presença desse sistema em diversas escolas públicas atendidas pelo LIpE.

Esta primeira implementação do assistente de fórmulas guiava o usuário através de janelas encadeadas que solicitavam as informações necessárias para a inserção de uma fórmula disponível no LIBRASOffice: a célula “inicial”, a célula “final”, e a célula de “resultado”. Embora tivéssemos utilizado as nomenclaturas “inicial” e “final”, esta primeira versão do assistente de fórmulas não suportava a inserção de fórmulas envolvendo faixa de células.



	A	B	C	D
1	R$ 1,00		45,00	46,00
2	R$ 2,00		46,00	47,00
3	R$ 3,00		47,00	48,00
4	R$ 4,00		48,00	49,00
5	R$ 5,00		49,00	50,00
6	R$ 6,00		50,00	51,00
7	R$ 7,00		51,00	52,00
8	R$ 8,00		52,00	53,00
9	R$ 9,00		53,00	54,00
10	R$ 10,00		54,00	55,00
11	R$ 11,00		55,00	56,00
12	R$ 12,00		56,00	57,00
13	R$ 13,00		57,00	58,00
14	R$ 14,00		58,00	59,00
15	R$ 15,00		59,00	60,00
16	R$ 16,00		60,00	61,00
17	R$ 17,00		61,00	62,00
18	R$ 18,00		62,00	63,00
19	R$ 19,00		63,00	64,00
20	R$ 20,00		64,00	65,00
21	R$ 21,00		65,00	66,00
22	R$ 22,00		66,00	67,00
23	R$ 23,00		67,00	68,00
24	R$ 24,00		68,00	69,00
25	R$ 25,00		69,00	70,00
26	R$ 26,00		70,00	71,00
27			71,00	72,00
28	R$ 351,00		72,00	73,00
29				
30				3304
31				

Figura 3.3 – Versão inicial do assistente de fórmulas do LIBRASOffice.

A distribuição do LIBRASOffice, durante a terceira versão, consistia em um diretório contendo os seguintes itens:

- *LIBRASOffice.sh*: script shell “lançador” do software. Sua função é automatizar a inicialização do LIBRASOffice em tantas distribuições Linux quanto possível.
- *LIBRASOffice.jar*: arquivo gerado pela IDE através da OpenJDK que contém as principais classes Java do projeto, a serem interpretadas por uma Java Virtual Machine (JVM).
- LEIAME.txt: arquivo de texto contendo informações extras sobre como executar manualmente o LIBRASOffice, em caso de falha de outros métodos.
- *lib*: diretório contendo demais classes/recursos necessárias para a execução do LIBRASOffice, dentre as quais: classes Java da API LibreOffice, classes referentes ao layout utilizado (AbsoluteLayout) e animações empacotadas em formato JAR.

Esta versão também foi bem avaliada e a proposta do projeto foi, de fato, validada. A principal recomendação feita pelos usuários finais foi que passássemos a contar com um colega surdo trabalhando ativamente conosco durante todo o desenvolvimento. Desde então, o LIBRASOffice passou a compor projetos de extensão universitária que demandaram o esforço conjunto entre os desenvolvedores e a comunidade surda.

3.5 – Quarta versão

A fim de aprofundarmos o nível de adaptação da GUI do LibreOffice, de forma que os componentes gráficos em Libras desenvolvidos se apresentassem visualmente integrados à GUI já exibida ao usuário, decidimos investir considerável quantidade de tempo no estudo do código interno do LibreOffice e de sua documentação associada. O objetivo deste estudo era encontrar o trecho do código interno responsável pela renderização dos componentes gráficos da GUI do LibreOffice como barras de ferramentas, botões e menus, a fim de que pudéssemos modificar o mecanismo de renderização de *tooltips*.

3.5.1 – Mecanismo adaptador de GUIs para Libras

Após certo tempo de exploração do código e estudo da documentação, conseguimos descobrir a rotina responsável por disparar a exibição de *tooltips* na tela do usuário, implementada dentro do módulo interno “vcl” do LibreOffice. Nesse momento, para avançarmos rumo ao objetivo de integração visual, ponderamos entre duas possibilidades: a primeira consistia na criação ou modificação de componentes gráficos capazes de exibir animações Libras, baseados na estrutura gráfica (*C++*) do próprio LibreOffice, o que nos permitiria alcançar o melhor desempenho possível (desempenho nativo); a segunda possibilidade consistia no desenvolvimento de um “mecanismo” capaz de extrair do LibreOffice informações sobre a opção atualmente ativada na GUI, a fim de delegar a um programa externo a tarefa de renderização da descrição em Libras associada a tal opção. Esta segunda possibilidade nos permitiria desenvolver uma interface gráfica customizada para os nossos objetivos, utilizando a biblioteca gráfica que julgássemos mais apropriada, totalmente à parte da GUI a ser adaptada, uma vez

que o mecanismo citado atuaria para garantir a desejada integração visual. Caso conseguíssemos “engenheirar” tal mecanismo, poderíamos aplicar o processo de adaptação para outros programas de código aberto sem maiores dificuldades. Já dispondo de uma interface gráfica, a nível de protótipo, bem-sucedida e desenvolvida na linguagem Java, a segunda possibilidade nos pareceu consideravelmente mais interessante, pois dentre outras vantagens, iríamos dispor da característica multiplataforma nativa da linguagem Java.

A lógica pensada para tal mecanismo baseou-se no seguinte: constatamos que a rotina interna ao LibreOffice responsável por disparar a exibição das *tooltips*, recebia como argumento o texto a ser exibido no interior da *tooltip*. Sendo assim, poderíamos extrair esta informação e gravá-la em uma nova linha de um arquivo de registro (log), situado em localização pré-determinada no sistema de arquivos. Um programa externo, responsável por efetivamente exibir ajuda em Libras, permaneceria em constante monitoramento de novas linhas em tal log. A cada nova linha detectada, uma análise de seu conteúdo seria realizada, a fim de identificar a tradução em Libras correspondente ao texto de ajuda que seria exibido em português. Em caso positivo, o conteúdo em Libras seria exibido em uma janela personalizada, próxima à posição atual do mouse na tela do usuário.

Tal mecanismo foi efetivamente implementado, constituindo a 4º versão, que foi de fato, uma iteração (ver Figura 3.4).

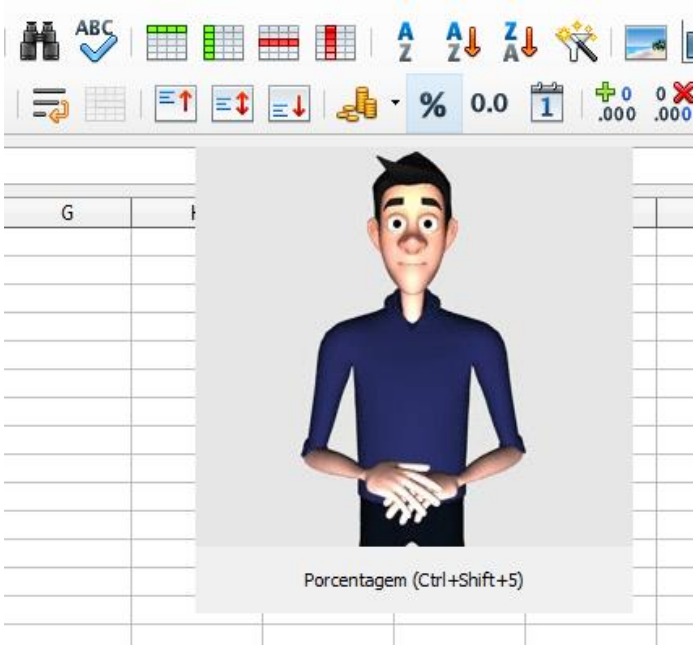


Figura 3.4 – Mecanismo de integração visual em funcionamento.

À época, o código do LIBRASOffice estava pronto para renderizar a ajuda em Libras de todos os botões presentes na barra de ferramentas principal do LibreOffice, embora não tivesse ainda a capacidade de identificar a ativação e desativação de menus da GUI.

3.5.2 – Diagrama de classes

Em projetos de software orientado a objetos, utilizamos o diagrama de classes para documentar visualmente, de forma estruturada, uma visão geral das classes, relacionando seus atributos (estado), métodos (comportamento) e relacionamentos [1].

A Figura 3.5 apresenta o diagrama de classes do LIBRASOffice, em sua quarta versão:

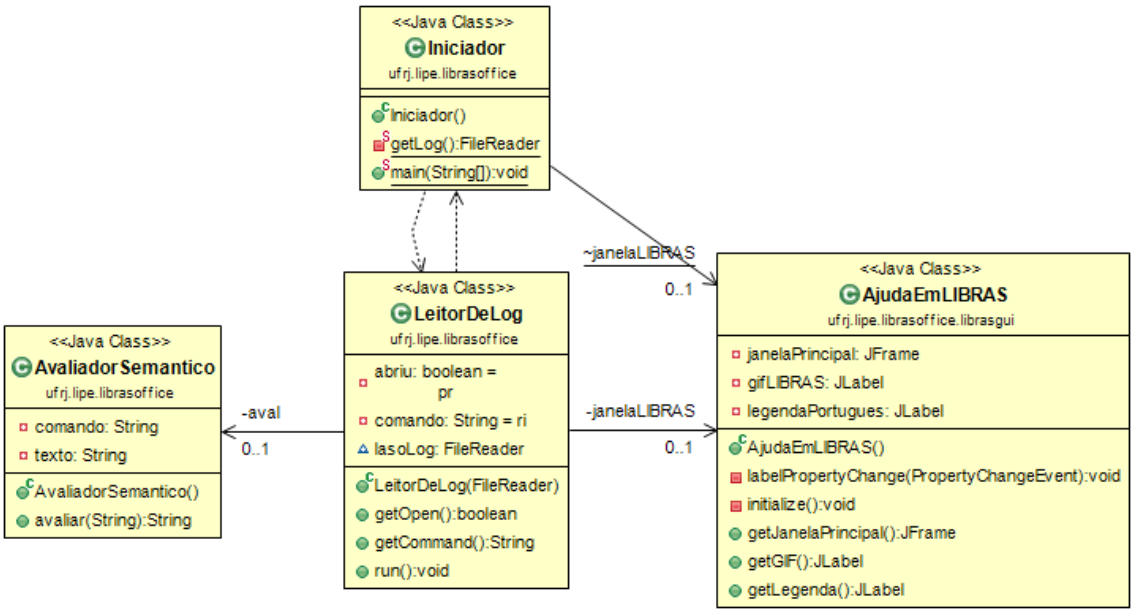


Figura 3.5 – Diagrama de classes do LIBRASOffice.

Estas classes, assim como demais classes que vieram a incrementar o software em versões posteriores, serão descritas em mais detalhes na seção referente à sétima versão (seção 3.8).

3.5.3 – Sinais informáticos em Libras

Ao tentarmos preencher nossa biblioteca de animações Libras com todos os sinais necessários para traduzir a barra principal, percebemos que muitos sinais apresentados pelo tradutor VLibras eram gesticulados de maneira diferente da conhecida por colegas fluminenses fluentes em Libras, como também percebemos que alguns sinais informáticos inexistiam no software tradutor. Iniciamos, então, um levantamento dos sinais necessários que estavam disponíveis no VLibras, a fim de classificá-los em relação ao regionalismo da Libras gesticulada no estado do Rio de Janeiro. Um resultado parcial deste levantamento pode ser observado na Figura 3.6.

Palavras que existem, mas que Leandro discordou da tradução ou não lembrava bem, verificar:	Palavras certas, confirmadas pelo Leandro:	Palavras que não existem no Vlibras, verificar existẽncia de sinônimos:
MAIS	NOVO	**ANEXO**
CHEGAR	ABRIR	**ORDEM**
SALVAR	GUARDAR	**EXPORTAR**
PÁGINA	E-MAIL	**CRESCENTE**
COPIAR	DOCUMENTO	**FUNDO**
ESTILO	EDITAR	**CÉLULA (DE TABELA)**
FORMATAR (ESPERAR	ARQUIVO	
PELO SINAL)	IMPRIMIR	
SUBLINHAR	VISUALIZAR	
ESQUERDA	CORTAR	
DIREITA	COLA(R)	
JUSTIFICAR	DESFAZER (VOLTAR)	
NUMERAR	REFAZER	
MARCADOR	TABELA	

Figura 3.6 – Levantamento de condição regional dos sinais do VLibras.

Outro desafio origina-se do fato de alguns sinais informáticos, já bem comuns na língua escrita, encontrarem-se ainda indefinidos na comunidade surda. A Figura 3.7 apresenta indicações de expressões escritas em português cujos respectivos sinais em Libras encontram-se indefinidos pela comunidade surda fluminense.

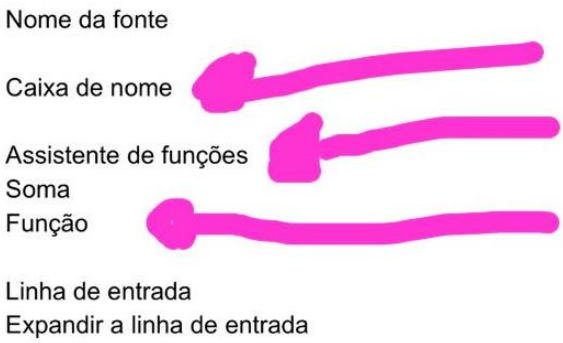


Figura 3.7 – Expressões em português com tradução indefinida em Libras.

Optamos por progressivamente produzir uma biblioteca própria com os sinais informáticos já existentes na Libras, regionalizados para o Rio de Janeiro. Para isso contamos com a ajuda de jovens estudantes surdos, selecionados para o projeto mediante bolsa PIBIC-EM pelo LabIS, que gravaram clipes de vídeo com os sinais necessários (ver Figura 3.8). A partir de então, sempre que possível, as animações em Libras utilizadas pelo projeto apresentavam uma pessoal real da comunidade surda, e não mais o boneco virtual do VLibras.

Figura 3.8 – Mosaico com clipes de vídeo de sinais informáticos do RJ.

3.6 – Quinta Versão

A introdução do uso de sinais Libras regionalizados, de produção própria, e a implementação da tradução de menus da GUI do LibreOffice (ver Figura 3.9), constituíram os grandes incrementos da quinta versão. Como pequeno aprimoramento, a janela de ajuda em Libras passou a apresentar uma área inferior que apresentava a legenda do sinal sendo exibido no momento (ver inferior da Figura 3.9).

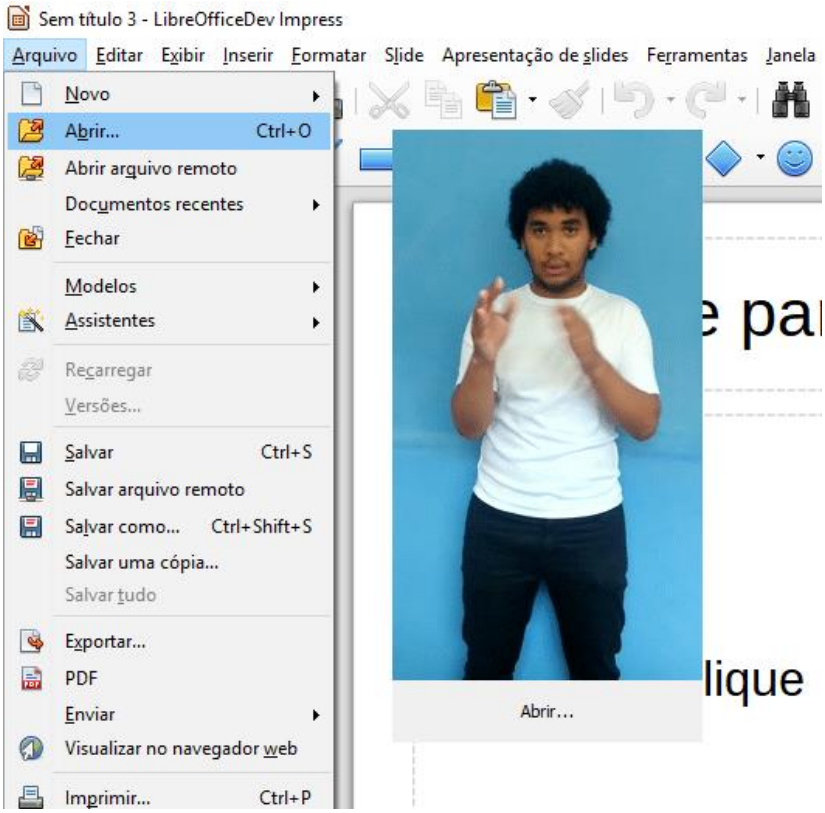


Figura 3.9 – Tradução para Libras de item de menu em português.

Outro pequeno incremento desta versão foi a introdução da janela “Sinal Indisponível”, que é exibida ao usuário sempre que o LIBRASOffice não dispõe da tradução em Libras para a descrição de algum elemento da interface gráfica do LibreOffice. Esta janela simplesmente exibe ao usuário a descrição padrão em português do item que não dispõe de tradução.

À época desta versão, solicitou-se um novo requisito: um assistente para envio remoto de sinais indisponíveis na interface do LIBRASOffice, que será melhor abordado na sétima versão.

3.7 – Sexta Versão

Na sexta versão foi dada especial atenção ao aprimoramento do recurso de inserção de fórmulas mediante assistente em Libras. Sendo assim, foi definido que:

- O assistente deveria ser totalmente funcional para fórmulas básicas, que envolvessem até dois argumentos, como por exemplo: AGORA(), HOJE(), SOMA(A1;A2), MÉDIA(B1:B2).
- O assistente deveria suportar a inserção de fórmulas considerando faixa de células.
- Após o término do assistente, o Calc deveria focar a célula de resultado.

Sendo assim, o assistente de inserção de fórmulas do LIBRASOffice foi totalmente reescrito, sendo implementado da seguinte maneira: uma pequena janela apresenta ao usuário as fórmulas disponíveis (ver Figura 3.10). Após a seleção da fórmula desejada, uma janela com opções comuns às fórmulas é apresentada ao usuário (ver Figura 3.11). Após o preenchimento dos campos necessários, a fórmula é inserida na planilha atual (via método UNO). O assistente é então finalizado e, logo após, a célula de resultado é focada na GUI do Calc (via método UNO). A inserção de fórmulas envolvendo faixa de células passou a ser totalmente suportada.

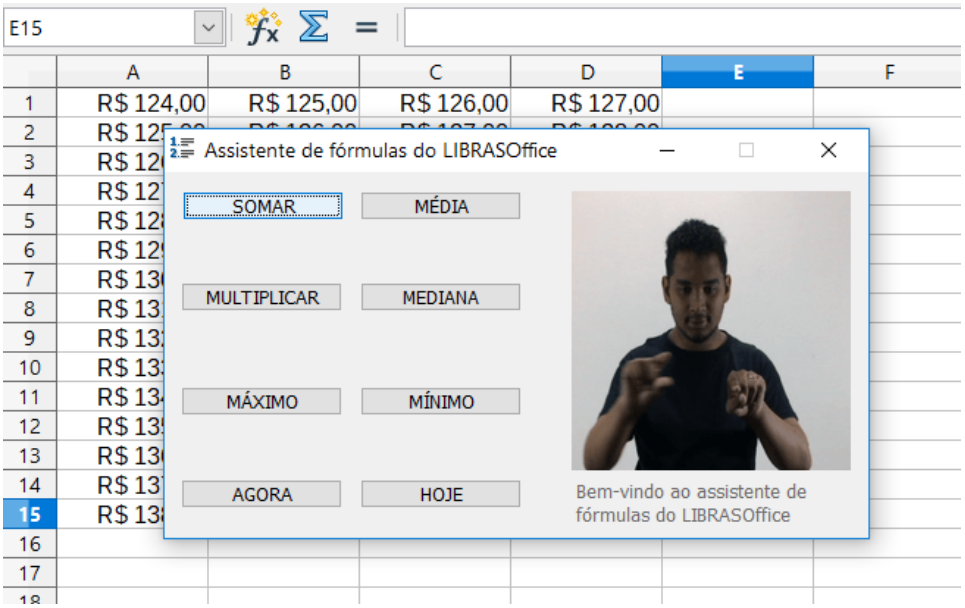


Figura 3.10 – Janela inicial do Assistente de Fórmulas do LIBRASOffice.

Figura 3.11 – Janela de opções para inserção de fórmula via assistente.

Em ambas as janelas, a região que exibe o sinal Libras é dinamicamente alterada à medida que o usuário move o mouse por cima das opções, sempre exibindo a explicação em Libras da opção em foco.

3.8 – Sétima Versão

A sétima versão do LIBRASOffice é a versão final do projeto, para fins deste trabalho. O principal destaque da sétima versão é a introdução de um assistente de envio remoto de sinais, de forma que usuários distantes podem agora colaborar ativamente com o projeto, enviando vídeos de sinais recém-definidos ou regionalizados diretamente para um diretório armazenado "na nuvem", mais especificamente, no Google Drive do LIpE.

Esta funcionalidade foi implementada aproveitando-se a existência da janela "Sinal Indisponível". Tal janela passou a apresentar um botão que leva o usuário ao assistente de envio de sinais, onde são solicitadas informações complementares ao vídeo que será enviado (ou ainda, metadados da mídia), como informações de contato e regionalismo. Também é solicitado ao usuário consentimento em eventual utilização pública da mídia que será enviada. O usuário pode acompanhar o progresso do upload da mídia por uma barra de progresso animada presente na região inferior da janela assistente. Semelhantemente ao assistente de fórmulas, existe uma região da janela dedicada à exibição de conteúdo em Libras, dinamicamente alterada à medida que o usuário move o mouse por cima das opções, sempre exibindo a explicação em Libras da opção em foco (ver Figura 3.12).

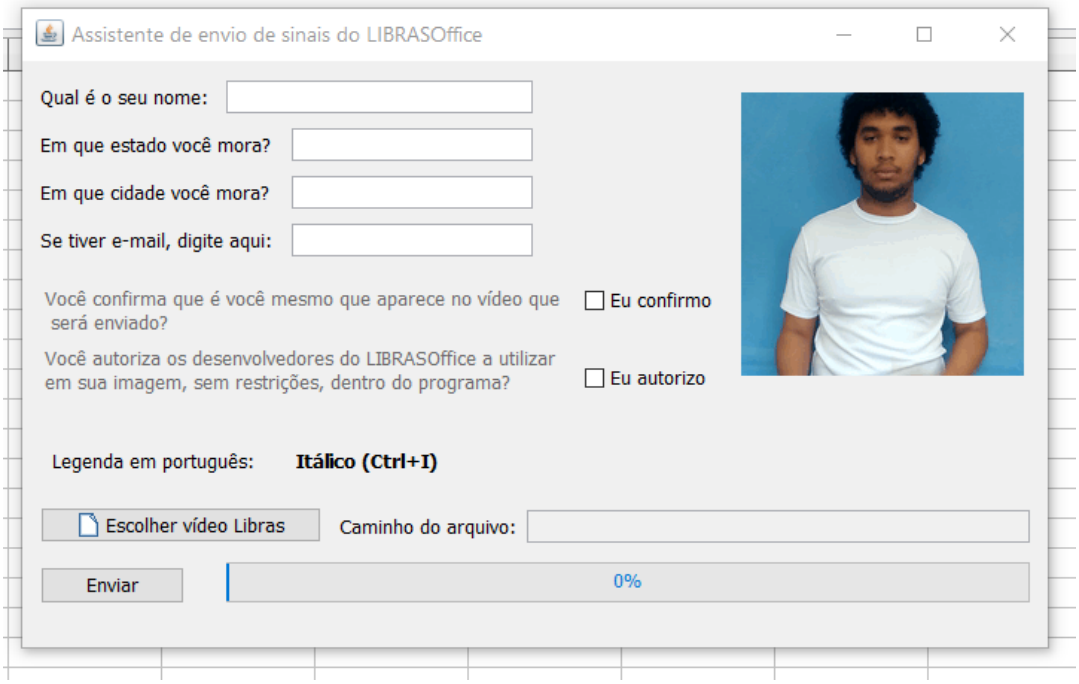


Figura 3.12 – Assistente de envio remoto de sinais.

Ao enviar a mídia o LIBRASOffice, internamente: guarda as informações complementares em arquivo temporário de texto de plano; gera um arquivo temporário compactado, em formato ZIP, contendo ambos os metadados e a mídia (geralmente um vídeo) a ser enviada. Este arquivo ZIP é o arquivo efetivamente enviado ao diretório remoto no Google Drive.

Esta integração do LIBRASOffice com o serviço de armazenamento em nuvem Google Drive é possível através da *Drive API Client Library for Java* [36].

Esta versão também marcou o aprimoramento do sistema de distribuição do LIBRASOffice, que passou a disponibilizar aos usuários, instaladores para Windows e Linux, além de versões portáteis executáveis diretamente de *pen-drives*.

3.8.1 – Diagrama de classes

O diagrama de classes do LIBRASOffice, em sua sétima versão, encontra-se no Apêndice A.

3.8.2 – Diagrama de casos de uso

A Figura 3.13 apresenta o diagrama de casos de uso da sétima versão do LIBRASOffice:

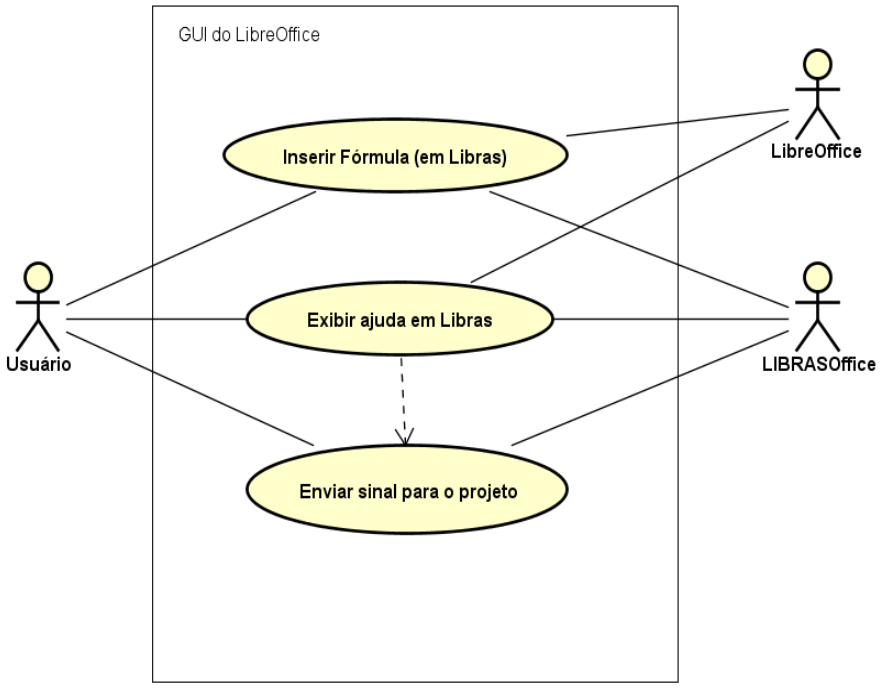


Figura 3.13 – Diagrama de casos de uso do LIBRASOffice.

O ator “LibreOffice” está relacionado à instância em execução da versão modificada do LibreOffice, como relatado em seções anteriores.

O fluxo de ações e a participação de cada ator, em cada cada de uso, são elucidados nas subseções seguintes:

3.8.3 – Descrição dos casos de uso

Seguem descrições do fluxo principal de cada caso de uso, formatado de acordo com a sequência de ações tomadas por cada ator.

3.8.3.1 – Exibir ajuda em Libras

Usuário	**LibreOffice**	**LIBRASOffice**
1. Passa o mouse em cima de um botão ou item de menu, trazendo o foco na GUI do LibreOffice para este *widget*.		
	2. Detecta a ação do usuário e registra em arquivo de log o texto descritor do *widget* em foco na GUI**.**	
		3. Detecta nova entrada no log, analisa-a e exibe na tela do usuário o sinal Libras associado ao *widget* em foco. Se sinal não for encontrado, exibe janela com legenda indicando esta indisponibilidade.

3.8.3.2 - Inserir Fórmula (em Libras)

Usuário	LibreOffice	LIBRASOffice
1. Pressiona o botão “Assistente de fórmulas” na GUI do LIBRASOffice.		
		2. Apresenta janela assistente em Libras que exibe as fórmulas disponíveis.
3. Seleciona a fórmula desejada.		
		4. Apresenta janela assistente em Libras que exibe os campos necessários para a inserção da fórmula selecionada.
5. Preenche os campos solicitados pelo assistente: célula inicial, final e de resultado; ativação, ou não, de faixa de células; Pressiona o botão “INSERIR FÓRMULA”		
		6. Realiza chamada de método UNO para inserção de fórmula na instância em execução do LibreOffice.

	7. Recebe fórmula, via UNO, e insere na planilha atual.	
		8. Realiza chamada de método UNO para levar foco da GUI para a célula de resultado.
	9. Recebe comando de foco, via UNO, e leva foco para célula de resultado.	

3.8.3.3 - Enviar sinal para o projeto

Usuário	**LibreOffice**	**LIBRASOffice**
1. Pressiona o botão “ENVIE O SINAL PARA NÓS” na GUI do LIBRASOffice.		
		2. Apresenta janela assistente em Libras que exibe os campos necessários (metadados) para o envio remoto de mídia contendo sinal Libras não presente no software.

3. Preenche os campos solicitados pelo assistente: nome; endereço; email; consentimento de uso; arquivo local de mídia. Pressiona o botão “ENVIAR”.

4. Envia mídia junto aos metadados, mediante arquivo compactado, para diretório remoto no Google Drive do LIpE.

3.8.4 – Dicionários de dados

Os atributos presentes nos diagramas de classe em UML podem ser melhor elucidados se acompanhados de um dicionário de dados[4]. As tabelas abaixo constituem o dicionário de dados, separado por classes, para o diagrama apresentado na seção 3.8.1.

3.8.4.1 - AvaliadorSemantico

Tabela 3.1 – tabela descritora da classe AvaliadorSemantico

Atributo	**Descrição**
linhaLog	Armazena linha de texto que descreve *widget* em foco na GUI do LibreOffice, extraída de log gerado por versão modificada do mesmo.
comando	Armazena substring de "linhaLog" que descreve a funcionalidade do *widget*.
tipoWidget	Armazena substring de "linhaLog" que descreve o tipo de *widget* (tooltip, menu, …).
comandoLibras	Armazena código textual para identificação de sinal Libras associado a "comando"

3.8.4.2 - InterpretadorDeLog

Tabela 3.2 – dicionário de dados da classe InterpretadorDeLog

Atributo	**Descrição**
aberto	Armazena o estado atual de "abertura" do arquivo de log gerado pelo backend.

[4] Um dicionário de dados é um documento estruturado que descreve os atributos de uma entidade (ou classe), elucidando sua semântica. São geralmente utilizados na área de Banco de Dados, acompanhando diagramas Entidade-Relacionamento (que representam bases de dados) para melhor explicar os atributos de cada entidade. [37]

comandoLibras	Armazena referência para “comandoLibras” de AvaliadorSemantico.
lasoLog	Armazena referência para manipulador de arquivo de log carregado na memória.

.

3.8.4.3 - JanelaFlutuanteLIBRAS

Tabela 3.3 – dicionário de dados da classe JanelaFlutuanteLIBRAS

Atributo	**Descrição**
gifLIBRAS	Armazena referência para *widget* (rótulo) desta janela que exibe sinal Libras (arquivo GIF animado).
legendaPortugues	Armazena referência para *widget* (rótulo) que apresenta legenda em português para o sinal sendo exibido.
btnAssistente	Armazena referência para *widget* (botão) de rótulo “ASSISTENTE DE FÓRMULAS”.
assitenciado	Armazena o estado atual de visibilidade de “btnAssistente”.

3.8.4.4 - Iniciador

Tabela 3.4 – dicionário de dados da classe Iniciador

Atributo	**Descrição**
LASO_TMP_PATH	Armazena texto que referencia caminho absoluto para diretório de arquivos temporários do sistema operacional em execução, no qual o log gerado pelo LibreOffice (modificado por este trabalho) será encontrado.

3.8.4.5 - AssistenteDeEnvio

Tabela 3.5 – dicionário de dados da classe AssistenteDeEnvio

Atributo	**Descrição**
gifLIBRAS	Armazena referência para *widget* (rótulo) desta janela que exibe sinal Libras (arquivo GIF animado).
txtMediaPath	Armazena referência para *widget* (campo de texto) que exibe caminho absoluto no sistema de arquivos para o arquivo de mídia selecionado pelo usuário.
arquivoMidia	Armazena referência para manipulador de arquivo de mídia carregado na memória.
arquivoZIP	Armazena referência para manipulador de arquivo temporário ZIP carregado na memória.
caminhoArquivoZIP	Armazena texto que referencia caminho absoluto para temporário arquivo ZIP.
txtSeuNome	Armazena referência para *widget* (campo de texto) que solicita nome do usuário.
txtEmQueEstado	Armazena referência para *widget* (campo de texto) que solicita a Unidade Federativa do usuário.
txtEmQueCidade	Armazena referência para *widget* (campo de texto) que solicita a Cidade do usuário.
txtSeTiverEmail	Armazena referência para *widget* (campo de texto) que solicita o e-mail do usuário.

3.8.4.6 - TransmissorGDrive

Tabela 3.6 – dicionário de dados da classe TransmissorGDrive

Atributo	**Descrição**
service	Armazena referência para manipulador de acesso a diretório remoto no Google Drive.
ioFile	Armazena referência para manipulador de arquivo carregado na memória, a ser enviado para diretório remoto.
progressBar	Armazena referência para *widget* (rótulo) que exibirá barra de progresso de upload do arquivo para diretório remoto.
lblProgress	Armazena referência para widge (rótulo)t que informará, textualmente, progresso de upload do arquivo para diretório remoto.

3.8.4.7 - ControlProgressoUpload

Tabela 3.7 – dicionário de dados da classe ControlProgressoUpload

Atributo	**Descrição**
pbar	Armazena referência para "progressBar" de TransmissorGDrive.
plbl	Armazena referência para "lblProgress" de TransmissorGDrive.

3.8.4.8 - AssistenteFormulas

Tabela 3.8 – dicionário de dados da classe AssistenteFormulas

Atributo	**Descrição**
gifLIBRAS	Armazena referência para *widget* (rótulo) desta janela que exibe sinal Libras (arquivo GIF animado).
aberto	Armazena o estado atual de exibição da janela do assistente de fórmulas, a fim de evitar duplicação desta janela.
formula	Armazena texto identificador UNO para a fórmula selecionada.
noArgs	Armazena o estado de necessidade de argumentos para a fórmula selecionada.
initialCell	Armazena referência para *widget* (campo de texto) que solicita ao usuário a célula inicial para a fórmula.
finalCell	Armazena referência para *widget* (campo de texto) que solicita ao usuário a célula final para a fórmula.
resultCell	Armazena referência para *widget* (campo de texto) que solicita ao usuário a célula de resultado para a fórmula.
opcoesDeFormula	Armazena referência para grupo (painel) de *widgets* que apresentam ao usuário as fórmulas disponíveis.
camposDaFormula	Armazena referência para grupo (painel) de *widgets* de campos necessários para inserção de determinada fórmula na planilha atual.
isCellRange	Armazena referência para *widget* (*checkbox*) que determina faixa de células.

txtStartFormula txtEndFormula txtRange	Armazena referência para *widget* (campo de texto) que exibe ao usuário como a fórmula deveria ser montada para ser inserida diretamente na planilha, sem o uso de assistente.

3.8.4.9 - ControladorUNO

Tabela 3.9 – dicionário de dados da classe ControladorUNO

Atributo	**Descrição**
LODesktop	Armazena referência para objeto UNO "Desktop" da instância do LibreOffice em execução. Objeto-base para uso da API UNO.
xDHelper	Armazena referência para objeto UNO "DispatchHelper". Objeto necessário para solicitação de alteração de foco na GUI do LibreOffice.
xDesk	Armazena referência para objeto UNO "XDesktop". Objeto necessário para acessar *widgets* da GUI do LibreOffice.

Capítulo 4

Resultado

4.1 – Processo genérico de adaptação de GUIs para Libras

Em termos técnicos, o processo que permite adaptação de GUI para Libras presente na última versão do LIBRASOffice pode ser sintetizado nas seguintes atividades:

- identificam-se rotina de renderização de elementos gráficos a serem traduzidos na interface gráfica do LibreOffice, como *tooltips* (“janelinhas de ajuda”) e itens de menus;
- modificam-se as rotinas acima identificadas a fim de se extrair, para um arquivo de registro (*log*), o texto a ser exibido no elemento gráfico;
- captura-se periodicamente (tempo suficientemente curto) a última linha do log, que é analisada. Esta análise retorna o nome do sinal correspondente ao texto do elemento gráfico;
- exibe-se, em interface gráfica apropriada, externa ao programa adaptado, o sinal Libras correspondente ao texto do elemento gráfico ativo no momento.

Este processo de adaptação de interfaces gráficas para Libras foi estabelecido de forma a ser possível sua replicação em outros softwares, de código-aberto, que possuam GUIs bem estruturadas, tratadas por rotinas bem definidas.

4.2 – Avaliação Final do Software

Constantes conversas com funcionários e alunos surdos, realizadas desde o primeiro protótipo, revelaram que a comunidade surda aprova com louvor a produção de softwares que os auxiliem na realização mais autônoma de tarefas diárias. Testes de validação, documentados, com 2 intérpretes, 2 professoras de escolas públicas com alunos surdos e 6 alunos e funcionários surdos de diversas faixas etárias, realizados ao longo de todo o processo de desenvolvimento, retornaram ótimos feedbacks.

Todos afirmaram que o software adaptado é mais fácil de ser utilizado, mas destacaram que ainda faltam recursos para uma utilização plena, como descrições em Libras mais extensas de cada opção que o programa oferece.

4.3 – Aprendizado

Durante o desenvolvimento do LIBRASOffice contamos com a participação de diversos colaboradores, dentre alunos surdos e ouvintes, professores universitários e de escolas públicas, intérpretes públicos e privados, coordenadores e coordenados, todos ouvidos em diferentes fases do processo de desenvolvimento e diretamente responsáveis por decisões que guiaram o andamento do projeto. Tal projeto não envolve apenas codificação, uma vez que demanda a produção de conteúdos multimídia em Libras (atividade já dominada pela comunidade surda) e até mesmo a definição de sinais informáticos ainda regionalmente indefinidos pela comunidade surda.

Em meio a este dinâmico contexto, os conceitos formais da engenharia de software foram fundamentais para o sucesso da produção de um artefato de software efetivamente utilizável. Fundamentais, mas não suficientes (para mais informações sobre a construção de softwares a partir da cultura popular, ver Severo [44]). O desenvolvimento de softwares voltados a pessoas surdas envolve uma série de desafios que transcendem a técnica: desde a comunicação dificultada por línguas de naturezas diferentes até a codificação de GUIs “orientadas a gestos”, projetos como o LIBRASOffice demandam considerável grau de comprometimento social dos desenvolvedores para com os usuários de seu produto, sendo imprescindível portanto, que o olhar sociotécnico permeie o processo de desenvolvimento do início ao fim [38].

Os conhecimentos adquiridos ao longo do curso, junto às ricas experiências obtidas na extensão universitária, garantiram a concretização do projeto, que cumpriu com seus objetivos, superando suas expectativas iniciais.

4.4 – Propostas Futuras

Para o futuro, pretende-se continuar ativamente o desenvolvimento do projeto. Há a possibilidade de contínua expansão do acervo de sinais presente no software, que passaria a melhor suportar os regionalismos da Libras. Também há a possibilidade de aprimoramento do assistente de fórmulas, de maneira que ele possa auxiliar a inserção de fórmulas complexas, envolvendo combinação de fórmulas simples. Há ainda a possibilidade de inclusão de explicações em Libras mais extensas para cada opção, de maneira que o usuário encontre ainda mais autonomia no aprendizado e no uso do software adaptado.

Há ainda a possibilidade de estender o processo de adaptação a outros softwares de produtividade, como navegadores web e editores de vídeo.

Bibliografia

[1] FOWLER, M.; *UML essencial*. Trad. 3 ed, Porto Alegre, Bookman, p. 40-52, 2005.

[2] IBGE, *Censo Demográfico 2010: Características gerais da população, religião e pessoas com deficiência*. Rio de Janeiro: IBGE, p. 114, 2012.

[3] HANDTALK, "Sobre", https://www.handtalk.me/sobre, 2017, (Acesso em 26 de novembro de 2017).

[4] LIBREOFFICE, "O que é o LibreOffice", https://pt-br.libreoffice.org/descubra/libreoffice/, 2017, (Acesso em 08 de outubro de 2017).

[5] BRASIL, Lei n. 10.436, de 24 de abr. de 2002. Dispõe sobre a Língua Brasileira de Sinais - Libras e dá outras providências, Brasília, DF, abr 2002.

[6] THIOLLENT, M. Extensão Universitária: Conceitos, Métodos e Práticas. Tradução. Rio de Janeiro: UFRJ, 2003. p. 57-68

[7]OPENHUB, "LibreOffice", https://www.openhub.net/p/libreoffice, 2018, (Acesso em 08 de março de 2018).

[8] ORACLE, " OpenJDK ", http://openjdk.java.net/, 2018, (Acesso em 08 de março de 2018).

[9] LINFO, "Widget Definition", http://www.linfo.org/widget.html, 2018, (Acesso em 08 de março de 2018).

[10] LIBREOFFICE, "VCL", https://docs.libreoffice.org/vcl.html, 2018, (Acesso em 08 de março de 2018).

[11] LIBREOFFICE, “Desktop”, https://docs.libreoffice.org/desktop.html, 2018, (Acesso em 08 de março de 2018).

[12] LIBREOFFICE, “XDesktop Interface Reference”, https://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1frame_1_1XDesktop.html, 2018, (Acesso em 08 de março de 2018).

[13] LIBREOFFICE, “LibreOffice Modules”, https://docs.libreoffice.org/index.html, 2018, (Acesso em 08 de março de 2018).

[14] MICROSOFT, “COM, DCOM, and Type Libraries”, https://msdn.microsoft.com/pt-br/library/windows/desktop/aa366757(v=vs.85).aspx, 2018, (Acesso em 08 de março de 2018).

[15] OPENOFFICE, “Overview: Universal Network Objects (UNO), https://www.openoffice.org/udk/common/man/uno.html, 2018, (Acesso em 08 de março de 2018).

[16] LIBREOFFICE, “LibreOffice 6.0 API Documentation”, https://api.libreoffice.org/, 2018, (Acesso em 08 de março de 2018).

[17] RAMME, K., “UNO ”, https://wiki.openoffice.org/wiki/Uno, 2018, (Acesso em 08 de março de 2018).

[18] OPENOFFICE, “Example: Working with a Spreadsheet Document”, https://wiki.openoffice.org/wiki/Documentation/DevGuide/FirstSteps/ExampleWorking_with_a_Spreadsheet_Document, 2018, (Acesso em 08 de março de 2018).

[19] DAVISON, A., “Java LibreOffice Programming”, http://fivedots.coe.psu.ac.th/~ad/jlop/, 2018, (Acesso em 08 de março de 2018).

[20] PFLEEGER, S. L. *Engenharia de Software: Teoria e Prática*, cap. Modelagem do Processo e Ciclo de Vida, Prentice Hall do Brasil, 2.ed, pp. 45–46, 2004.

[21] BEZERRA, EDUARDO, *Princípios de Análise e Projeto de Sistema com UML,* cap. O processo de desenvolvimento de software, Elsevier, 3. ed, 2015.

[22] ORACLE, "Welcome to NetBeans", https://netbeans.org/, 2018, (Acesso em 08 de março de 2018).

[24] MPOG, "VLIBRAS - Tradução de Português para Libras", http://www.vlibras.gov.br/, 2018, (Acesso em 08 de março de 2018).

[25] ECLIPSE, "Eclipse desktop & web IDFs", https://www.eclipse.org/ide/, 2018, (Acesso em 08 de outubro de 2018).

[26] ORACLE, "Java SE | Oracle Technology Network | Oracle", http://www.oracle.com/technetwork/pt/java/javase/overview/index.html, 2018, (Acesso em 08 de março de 2018).

[27] NOTEPAD++, "Notepad ++ Home", https://notepad-plus-plus.org/, 2018, (Acesso em 08 de março de 2018).

[28] BERNADINI, "Java - Progamacao GUI.pptx", http://www.lncc.br/~rogerio/poo/04a%20-%20Programacao_GUI.pdf, 2018, (Acesso em 08 de março de 2018).

[29] ORACLE, "How to Write Doc Comments For The Javadoc Tool", http://www.oracle.com/technetwork/java/javase/tech/index-137868.html, 2018, (Acesso em 08 de março de 2018).

[30] GIT, "GIT", https://git-scm.com/, 2018, (Acesso em 08 de março de 2018).

[31] GIT, “The word’s leading software development platform - GitHub”, https://github.com/, 2018, (Acesso em 08 de março de 2018).

[32] GIT, “LASOBack”, https://github.com/jeliascosta/LASOBack, 2018, (Acesso em 08 de março de 2018).

[33] GIT, “LASOFront”, https://github.com/jeliascosta/LASOFront, 2018, (Acesso em 08 de março de 2018).

[34] GIT, “LIBRASOffice”, https://github.com/jeliascosta/LIBRASOffice, 2018, (Acesso em 08 de março de 2018).

[35] MICROSOFT, “What is a ToolTip?”, https://msdn.microsoft.com/en-us/vba/language-reference-vba/articles/what-is-a-tooltip, 2018, (Acesso em 08 de março de 2018).

[36] GOOGLE, “Overview (Drive API v3 (REV, 105) 1.23,0)”, https://developers.google.com/resources/api-libraries/documentation/drive/v3/java/latest/, 2018, (Acesso em 08 de março de 2018).

[37] SPARXSYSTEM, “Data Dictionary| Enterprise Architect User Guide”, http://www.sparxsystems.com.au/enterprise_architect_user_guide/13.0/guidebooks/tech_data_dictionary.html, 2018, (Acesso em 08 de março de 2018).

[38] CUKIERMAN, H. L., TEIXEIRA, C., PRIKLADNICKIOI, R., “*Um Olhar Sociotécnico sobre a Engenharia de Software*”, RITA, v. 14, n° 2, pp. 205, 2007.

[39] LIBREOFFICE, “Development/Code Overview - The Document Foundation Wiki”, https://wiki.documentfoundation.org/Development/Code_Overview, 2018, (Acesso em 08 de março de 2018).

[40] GRANEMANN, JUSSARA L., "*Língua Brasileira De Sinais – Libras Como L1 Para Estudantes Surdos Nos Anos Iniciais Do Ensino Fundamental*", REVELLI, v. 9, n. 2, pp. 270, 2017.

[41] PRODEAF, "ProDeaf", http://prodeaf.net/, 2018, (Acesso em 19 de março de 2018).

[42] RYBENÁ, "Rybená", http://portal.rybena.com.br/site-rybena/, 2018, (Acesso em 19 de março de 2018).

[43] VIEIRA, ROSELI, "LIBRASOffice (Protótipo/Prototype)", https://youtu.be/0TSB6F9v9U0, 2016, (Acesso em 19 de março de 2018).

[44] SEVERO, FERNANDO GONÇALVES. *TICs e TACs: o refazimento de softwares e engenheiros no limiar entre as ciências e os segredos*. 2016. 164 f. Dissertação (Mestrado em Engenharia de Sistemas e Computação) – Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Apêndice A

Diagrama de classes da 7ª versão