Desafios sobre Acessibilidade em Ferramentas CASE

Simone de Fátima Skura[1], **Marlon Marcon**[1], **Franciele Beal**[2]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Dois Vizinhos
Dois Vizinhos - PR - Brasil

[2]Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Pato Branco
Pato Branco - PR - Brasil

simoneskura@gmail.com, {marlonmarcon,fbeal}@utfpr.edu.br

Abstract. *The software development cycle involves four stages: specification, design and implementation, validation, and evolution. CASE tools are essential but can pose obstacles for visually impaired developers. This work emphasizes the importance of including these developers by systematically mapping studies from 2012 to 2023 on the accessibility of software development environments for blind people. The aim is to identify challenges, adaptations, and recommended solutions to make these tools more inclusive. The results show few studies in this area, suggesting a relevant research opportunity; moreover, alternatives that can contribute to accessibility were identified.*

Resumo. *O ciclo de desenvolvimento de software tem quatro etapas: especificação, projeto e implementação, validação e evolução. Ferramentas CASE são essenciais, mas podem ser obstáculos para desenvolvedores com deficiência visual. Este trabalho destaca a importância da inclusão desses desenvolvedores, realizando um mapeamento sistemático de estudos de 2012 a 2023 sobre acessibilidade em ambientes de desenvolvimento de software para cegos. O objetivo é identificar desafios, adaptações e soluções recomendadas para tornar essas ferramentas mais inclusivas. Os resultados mostram poucos trabalhos na área, sugerindo uma oportunidade de pesquisa relevante, além disso, identificaram alternativas que podem contribuir para a acessibilidade.*

1. Introdução

O desenvolvimento de software é um processo de engenharia composto por quatro atividades principais: especificação, projeto e implementação, validação e evolução. Na especificação, as funcionalidades e restrições são definidas, seguidas pela transformação em software executável na etapa de projeto e implementação. A validação e teste garantem a conformidade com a especificação, enquanto a atividade de evolução lida com adaptações às mudanças. Para apoiar essas atividades, as ferramentas CASE (sigla em inglês para *Computer-Aided Software Engineering*) desempenham um papel crucial, incluindo ferramentas para criação de diagramas, ambientes de desenvolvimento integrado (IDE, sigla em inglês para *Integrated Development Environment*), controle de versões de arquivos, gestão de projetos e comunicação. Essas ferramentas aumentam a produtividade, facilitam o desenvolvimento e melhoram a qualidade do software [Sommerville 2011, Pressman and Maxim 2021].

DOI: 10.5753/sbie.2024.242571

As ferramentas CASE são complexas e oferecem diversas funções, geralmente requerendo treinamento para sua utilização. No entanto, estudos como o de [Nascimento et al. 2022] destacam que as ferramentas CASE utilizadas no desenvolvimento de software podem apresentar barreiras para profissionais com deficiência visual, prejudicando sua produtividade e integração nos processos de desenvolvimento. Essas limitações também são observadas na academia, conforme evidenciado no trabalho de [Silveira et al. 2019], que apontou problemas de acessibilidade em ferramentas CASE usadas em sala de aula. Para contornar essas barreiras, estudantes e profissionais cegos dependem de tecnologias assistivas, como leitores de tela como o NVDA[1], JAWS[2] e VoiceOver[3], que proporcionam uma certa independência ao ler informações na tela do computador por meio de síntese de voz. Entretanto, as ferramentas CASE utilizam interfaces gráficas de usuário (GUI sigla em inglês para *Graphical User Interface*) e muitos elementos que compõem essas interfaces não são legíveis por leitores de tela. Outra barreira é o mouse, que as pessoas com deficiência visual tem dificuldades para fazer o uso adequado. A entrada de dados por meio do teclado, menus com opções de teclas de atalho e comando de voz, são mais adequadas para essas pessoas [Luque et al. 2014].

Diante deste contexto, o presente trabalho investigou estudos sobre a acessibilidade de ambientes de desenvolvimento de software para desenvolvedores cegos, para responder as seguintes questões: *a) Quais ambientes de desenvolvimento de software (na academia e no mercado de trabalho) apresentam problemas de acessibilidade aos desenvolvedores cegos? b) Quais recomendações, adaptações ou recursos que podem ser usados para melhorar a acessibilidade dessas ferramentas?*. Para a investigação proposta, optou-se por um Mapeamento Sistemático (MS) para categorizar e sintetizar informações existentes sobre o tema de busca. MS incluiu estudos publicados em bases digitais nacionais e internacionais entre os anos 2012 e 2024. A busca foi realizada no Google Scholar, que retornou 68 artigos. Após aplicação de critérios de inclusão e exclusão, foram selecionados 17 artigos, a metodologia de pesquisa, resultados do mapeamento, a discussão e conclusão são apresentados nas seções subsequentes.

Como contribuições deste trabalho, podem-se citar:

- Levantamento da literatura na forma de um MS que identifica problemas e sugestões relacionados ao processo de desenvolvimento de software usando ferramentas CASE;
- Verificação prática das sugestões apontadas nos estudos, por meio de testes e validação de uso das ferramentas e dicas apresentadas;
- Apresentação de um relato de experiências, com objetivo de auxiliar no processo de adaptação de ferramentas CASE no contexto educacionais, para estudantes e professores, ou profissional, na área de Engenharia de Software.

2. Metodologia da Pesquisa

Este trabalho utilizou como base as diretrizes apresentadas por [Petersen et al. 2015]. Inicialmente, foram definidas cinco (5) questões de pesquisa específicas para responder às questões apresentadas na introdução deste trabalho:

[1]NVDA - NonVisual Desktop Access - leitor de tela gratuito para Windows
[2]JAWS - Job Access With Speech - leitor de tela pago para Windows
[3]VoiceOver - leitor de tela da Apple

QP1: Quais tecnologias assistivas os desenvolvedores cegos utilizam?
QP2: Quais são os ambientes de desenvolvimento de software utilizados pelos cegos?
QP3: Quais linguagens de programação os desenvolvedores cegos utilizam?
QP4: Quais são os principais problemas de acessibilidade apontados pelos estudos?
QP5: Os trabalhos apresentam alternativas para resolver os problemas de acessibilidade? Se sim, quais?

2.1. A Busca

Para a definição das palavras-chave e o agrupamento dos termos para formar a *string* de busca foi utilizada a metodologia PICO (*Population* ou População, *Intervention* ou Intervenção, *Comparison* ou Comparação e *Outcome* ou Resultados) de [Petersen et al. 2015]. A seguir é apresentada a string de busca usada neste trabalho:

(*"blind software developers"* OR *"visually impaired software developers"*) AND
(*"case tools"* OR *"software development tools"* OR *IDE* OR
"Integrated Development Environment" OR
"software development environments") AND (*accessibility*)

Onde:

População se refere aos usuários que utilizam as ferramentas CASE *("blind software developers", "visually impaired software developers")*;
Intervenção se refere ao que se quer encontrar *("case tools", "software development tools", IDE, "Integrated Development Environment", "software development environments")*;
Comparação não se aplica, pois, o objetivo é analisar as ferramentas CASE, não compará-las;
Resultados corresponde ao que ser quer atender, ou melhorar *(accessibility)*.

A *string* foi executada no Google Scholar[4]. A escolha pelo Google Scholar é justificada, pois ele indexa as bases de dados de artigos científicos mais utilizadas em pesquisas. Como resultado foram retornados 68 resultados. Foram considerados trabalhos entre os anos 2012 e 2024. Para a seleção dos estudos foram aplicados critérios de inclusão e exclusão.

2.2. Critérios de Inclusão

- Estudos sobre a acessibilidade de ambientes de desenvolvimento de software para desenvolvedores cegos, tanto em meio profissional ou acadêmico;
- Estudos nos idiomas inglês e português;
- Estudos gratuitos.

2.3. Critérios de Exclusão

- Não se tratar de um artigo (sumários, índices, etc.);
- Estudos duplicados;
- Estudos não disponíveis em texto completo;

[4]Google Scholar - http://scholar.google.com.br/

- Estudos na forma de dissertações, teses e revisões de literatura;
- Estudos que não atendem os critérios de inclusão.

Para a seleção dos estudos foram executadas 3 etapas: na primeira, foram lidos os títulos, resumos e palavras-chave e foram selecionados 34 estudos dos 68. Na segunda etapa, os estudos selecionados foram enviados para o Núcleo de Acessibilidade e Inclusão (NAI) da Universidade para serem adaptados, pois a primeira autora é cega e os estudos não estavam acessíveis para uma leitura integral. A terceira etapa consistiu na leitura integral dos estudos, onde, foram selecionados 17 estudos para responder às questões de pesquisa deste trabalho. A Figura 1 representa graficamente o processo descrito.

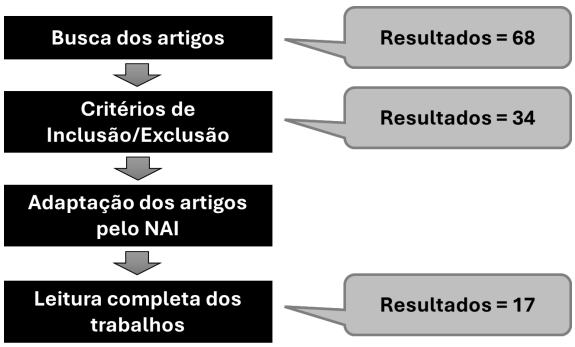

Figura 1. Processo de seleção dos artigos para o presente estudo.

3. Resultados e Discussões

Este MS teve como objetivo responder às questões de pesquisa apresentadas na seção anterior. Como citado anteriormente, foram publicados 17 estudos entre os anos de 2012 e 2024 que se enquadraram no contexto aqui apresentado. Destes, 14 foram publicados após 2017, sendo 10 deles após 2020, o que indica pesquisas recentes, tema atual e relevante (Figura 2).

Figura 2. Estudos por ano

Para responder a QP1, dos 17 estudos selecionados, apenas 13 estudos informaram as tecnologias assistivas usadas pelos usuários. A Figura 3 mostra o gráfico das tecnologias citadas e a quantidade de artigos que citou cada uma delas. As barras representam a totalidade dos artigos que mencionaram as tecnologias (13 trabalhos), sendo que

a porção mais escura está relacionada à proporção de citações das tecnologias. O NVDA e JAWS foram citados em 12 estudos, a linha Braille em 7 estudos, o VoiceOver em 5 estudos, Orca em 3 estudos, e as tecnologias Window Eyes e o Narrador do Windows, foram citadas uma vez cada.

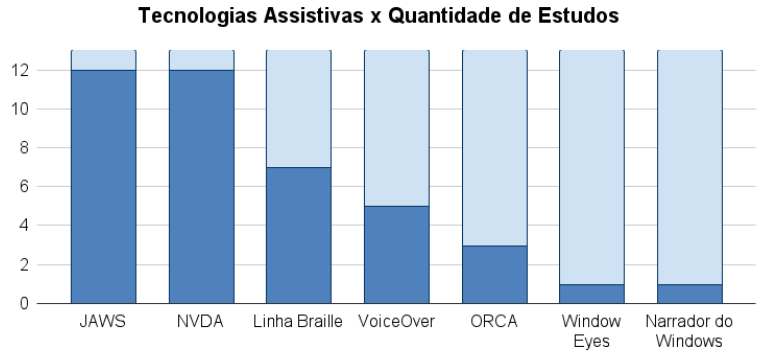

Figura 3. Resposta da QP1.

A Figura 4 apresenta a resposta da QP2 sobre os ambientes de desenvolvimento utilizados pelos cegos. De 17 trabalhos, apenas 8 estudos informaram os ambientes de desenvolvimento usados pelos participantes. O Eclipse foi citado por 6 estudos, o Visual Studio Code (VSCode) também foi citado por 6 estudos. O Netbeans foi citado por 2 estudos, o Xcode também foi citado por 2 estudos. Pycharm, Emacs, IDLE, Code Warrior e Visual ForPro foram citados por 1 estudo cada.

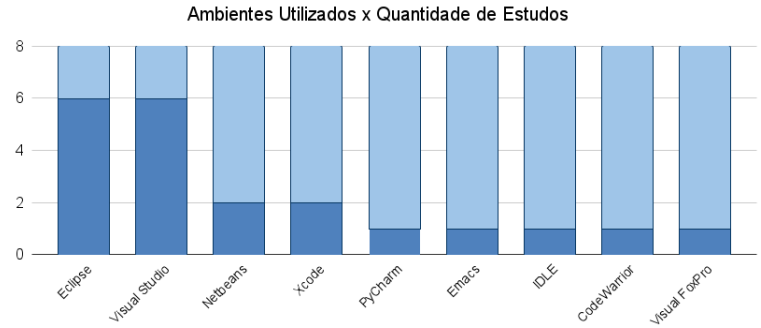

Figura 4. Resposta da QP2.

A Figura 5, apresenta a resposta da QP3. Dos 17 trabalhos, apenas 10 informaram a linguagem. Python, Java, C# e C foram as linguagens mais citadas, sendo 8, 7, 6 e 6 citações respectivamente. As linguagens C++, JavaScript e PHP foram citadas 5 vezes cada. A Perl foi citada em 4 estudos. A CSS, HTML, Ruby e SQL foram citadas em 3 estudos. A .NET, Objective-C, TypeScript e Swift foram citadas 2 vezes cada. A Jaws Scripting, Kotlin, COBOL, Assembly, Grovvy, Fortran, ActionScript, Visual FoxPro e Dart, foram citadas 1 vez cada.

As questões de pesquisa QP4 e QP5 estão relacionadas diretamente aos problemas de acessibilidade e a apresentação de alternativas a esses problemas. Ao analisar os trabalhos do presente MS, os pontos explorados por eles foram divididos em seis grupos relacionados às fases do desenvolvimento de sistemas, e também ao processo de aprendizagem destas ferramentas por estudantes ou profissionais cegos. Os grupos foram identificados como: 1) Escrita de Código e Navegação; 2) Depuração de Código; 3) Ambientes

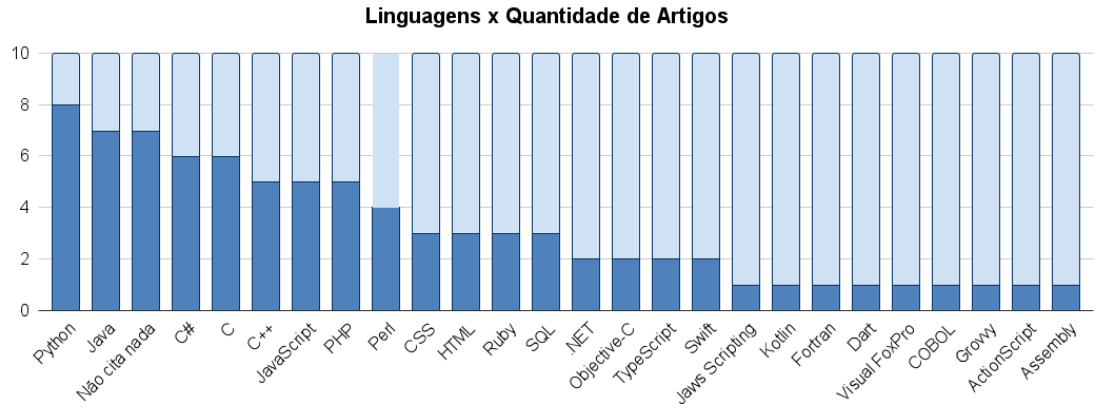

Figura 5. Resposta da QP3.

de Desenvolvimento (IDE) e Diagramas; 4) Ferramentas Diversas; 5) Ensino e materiais de aprendizagem; 6) Controle de Versões.

As seções a seguir apresentam os problemas apontados pelos trabalhos (respondendo à QP4) e as soluções apresentadas por estes para mitigar tais problemas (QP5). Além disso, quando as dicas e sugestões envolviam o uso de alguma ferramenta de apoio específica ou de algum comando, ou tecla de atalho, estes foram testadas pela primeira autora do trabalho, visando avaliar se, de fato, se aplicam no contexto de pessoas cegas e se ainda estão disponíveis para uso.

3.1. Escrita de Código e Navegação

Dentre os artigos utilizados no presente estudo, 12 deles abordaram problemas relacionados à Escrita de Código e Navegação, os pontos citados por estes trabalhos são apresentados no Quadro 1, na qual a coluna *Problemas* se refere às respostas a QP4 e a coluna *Sugestões* à QP5.

A partir das sugestões apresentadas, foram realizados testes nas ferramentas apresentadas. Com relação à Utilização de teclas de atalho no IDE Eclipse (Quadro 1, Sugestão 9), foi avaliado se as teclas de atalho citadas por [Albusays et al. 2017] e [Petrausch and Loitsch 2017] estavam disponíveis na versão 2019/03 do Eclipse IDE, e as seguintes teclas são funcionais nesta versão:

- **F12**: para definir foco no editor;
- **F3**: para ir para a declaração do elemento selecionado;
- **CTRL+1**: para mostrar correção rápida, opções de navegação com teclas de seta;
- **ALT+SHIFT+R**: para renomear o elemento selecionado e todas as referências;
- **CTRL+SHIFT+Seta cima/baixo**: para ir para o próximo método.

Em relação a utilização do *plugin* Structjumper para o Eclipse (Quadro 1, Sugestão 17), em busca nos complementos não foi encontrado o referido *plugin*, e como esta sugestão é de um trabalho de 2015 [Baker et al. 2015], os autores acreditam que foi descontinuado o suporte a ele, não sendo possível testá-lo.

3.2. Depuração de Código

Esta seção aborda os problemas e sugestões relacionadas à Depuração de Código. Foram identificados 10 trabalhos que apontaram problemas e identificaram possíveis soluções para eles, as quais são listadas no Quadro 2.

Quadro 1. Problemas e sugestões relacionadas a Escrita de Código e Navegação

Problemas	Sugestões
1. Identificar corretamente os espaços em branco para indentação de código; 2. Identificar nível de aninhamento em estruturas de código; 3. Compreender o aninhamento de blocos de código; 4. Compreender o escopo ao navegar por estruturas aninhadas; 5. Retornar para uma linha específica em um código extenso ao revisar outros arquivos; 6. Identificar a falta ou excesso de parênteses; 7. Usar eficientemente o recurso de autocompletar; 8. Identificar números de linha no código; 9. Identificar relacionamentos entre classes e subclasses; 10. Localizar erros e avisos em tempo real durante a codificação; 11. Navegar pelo código usando o teclado; 12. Localizar informações específicas em uma base de código; 13. Recursos de autocompletar(preenchimento automático); 14. Skimming de Código (os leitores de tela forçam os usuários a lerem todo o documento); 15. Sobrecarga auditiva; 16. Dificuldades em Colaborar na programação em pares devido à falta de acesso à tela do colega; 17. Identificar o cursor do colega que enxerga.	1. Utilizar a linha braille [Mealin and Murphy-Hill 2012]; 2. Utilizar sistemas de recomendação de comandos [Mealin and Murphy-Hill 2012]; 3. Utilizar ferramentas de navegação de código [Mealin and Murphy-Hill 2012]; 4. Pesquisar em código-fonte utilizando palavras-chave [Albusays et al. 2017]; 5. Utilizar a navegação por bloco em linguagens que utilizam a indentação, como Python [Albusays et al. 2017]; 6. Escrever script para forçar o leitor de tela a ler a quantidade de espaços e não a palavra ”espaço”[Albusays et al. 2017], [Zen et al. 2023b]; 7. Utilizar editores de texto para fazer anotações enquanto utiliza as IDEs [Albusays and Ludi 2016], [Albusays et al. 2017], [Zen et al. 2023a]; 8. Escrever script que informe o número de uma linha ou execute outras funções necessárias [Albusays et al. 2017]; 9. Utilizar teclas de atalho no IDE Eclipse, tais como a seguir [Albusays et al. 2017], [Petrausch and Loitsch 2017]; 10. Adicionar comentários descritivos [Albusays et al. 2017], [Pandey et al. 2021]; 11. Criar recurso de navegação hierárquica para que o usuário não precise fazer a leitura do código de forma linear [Albusays et al. 2017], [Zen et al. 2023b]; 12. Criar *feedbacks* sonoros [Albusays et al. 2017], [Zen et al. 2023b]; 13. Adicionar marcadores ou tags nos códigos [Albusays et al. 2017], [Zen et al. 2023b]; 14. Criar indicadores sonoros de alinhamento e escopo [Albusays et al. 2017]; 15. Implementar ferramenta sonora para auxiliar no acesso às relações entre classes e subclasses [Albusays et al. 2017]; 16. Criar documentação contendo as teclas de atalho das ferramentas [Huff et al. 2020]; 17. Utilizar o plugin Structjumper para o Eclipse para facilitar a navegação e compreensão do código Java [Baker et al. 2015]; 18. Adicionar comentários descritivos, usar maiúsculas e minúsculas e nomes longos de variáveis [Albusays et al. 2017], [Pandey et al. 2021]; 19. Utilizar editores como Grid-Coding de [Ehtesham-Ul-Haque et al. 2022] que representa o código-fonte em uma estrutura em grade 2D. Cada linha representa uma linha de código e cada coluna, um escopo no código. Esse tipo de representação pode facilitar a identificação de contextos, erros de sintaxe e organização do código de maneira mais clara e acessível.

Em relação às sugestões para melhoria da acessibilidade na tarefa de depuração de código, mais especificamente ao uso de teclas de atalho do Eclipse (Quadro 2, Su-

Quadro 2. Problemas e sugestões relacionadas a Depuração de Código

Problemas	Sugestões
1. Navegar e utilizar as ferramentas de depuração, exemplo de ferramentas citadas: FindBugs e Firebug; Compreender o fluxo de execução; 2. Encontrar erros no código; Identificar o progresso durante a depuração; 3. Acessar o status dos pontos de interrupção; 4. Ler o código linha a linha para identificar erros; 5. Mudar da tela de erro para a edição do código; 6. Usar a função *printf()* como solução alternativa, porém, esse método é demorado e ineficaz; 7. Informações sobre erros por meio de dicas visuais e os valores das variáveis e pontos de interrupção, estão indisponíveis nas IDE´s.	1. Utilizar os seguintes atalhos do Eclipse [Petrausch and Loitsch 2017]; 2. Solicitar o envio prévio de código antes de reuniões para revisão [Pandey et al. 2021].

gestão 1), a sugestões de [Petrausch and Loitsch 2017] foram testadas e validadas na versão 2019/3 do Elipse IDE.

- **CTRL+F7** para pular para a lista de erros;
- **CTRL+./CTRL+,** para percorrer uma lista de erros ou avisos.

3.3. Ambientes de Desenvolvimento (IDE) e Diagramas

Quando se trata de Ambientes de Desenvolvimento Integrados e de ferramentas para compreensão de Diagramas, 10 estudos exploraram problemas e sugestões relacionados ao uso de IDEs e três trabalhos relacionaram-se aos Diagramas. O Quadro 3 apresenta de forma integradas ambas categorias de ferramentas.

O grupo relacionado às IDEs e Diagramas é o que mais apresentou sugestões para enfrentar os problemas elencados. Tais itens são listados no Quadro 3, e a seguir os testes realizados para tais soluções são descritos.

Em relação ao item 1 da coluna de Sugestões do Quadro 3, como indicado por [Petrausch and Loitsch 2017], foram testados e validados os seguintes atalhos na versão 2019/3 do Eclipse:

- **CTRL+SHIFT+L**: Para listar todos os atalhos no Eclipse, porém, ao ser testado, não funcionou, ao digitar este atalho, alterna entre os modos de teclado ABNT 1 ou ABNT2;
- **ALT+SHIFT+Q**: Para visualização aberta (mesmo as não exibidas);
- **F10**: Para definir foco no menu;
- **CTRL+O**: Mostrar Contorno.

O item 2 do mesmo Quadro orienta a instalação de *plugins* ou extensões de acessibilidade, como o ACTF[5] (*Accessibility Tools Framework*) do Eclipse [Petrausch and Loitsch 2017]. O trabalho que sugere tal ferramenta foi publicado em 2017, e após consulta este não é mais encontrado na biblioteca de extensões do Eclipse.

[5]ACTF - https://eclipse.dev/actf/

Além disso, em consulta ao repositório oficial, verificou-se que a última versão do plugin foi lançada em 2018, estando, portanto este indisponível.

Quadro 3. Problemas e sugestões relacionadas a IDEs e Diagramas

Problemas	Sugestões
1. Instalação dos ambientes; 2. Acessibilidade no Eclipse e Visual Studio Code (VS Code); 3. Tarefas relacionadas a componentes visuais, como arquitetura do programa e layouts de interface de usuário; 4. Falta de acessibilidade em componentes de interface do usuário, componentes inacessíveis; 5. Falta de confiança para executar tarefas relacionadas ao design de interface de usuário; 6. Interpretar layouts de interface; Inspecionar o próprio trabalho de design; 7. Escrever e determinar medidas dos componentes em arquivos CSS; 8. Editar layouts visuais de páginas web; Implementar o design de páginas web devido à falta de conhecimento sobre a localização dos elementos visuais na página; 9. Dificuldade para encontrar as teclas de atalhos; 10. As IDE'S e os leitores de tela podem utilizar um mesmo atalho para executar diferentes funções; 11. Compreender e interpretar o layout de interfaces criadas; 12. Além disso, verificar o posicionamento, alinhamento e formatação dos elementos; 13. Falta de acessibilidade em diagramas, como UML (*Unified Modeling Language*) e arquitetura de software.	1. Utilizar atalhos acessíveis do Eclipse [Petrausch and Loitsch 2017]; 2. Instalar plugins ou extensões de acessibilidade, como o ACTF (*Accessibility Tools Framework*) do Eclipse [Petrausch and Loitsch 2017]; 3. Criar avisos sonoros para dar *feedbacks* no Eclipse [Petrausch and Loitsch 2017]; 4. Utilizar um editor em conjunto com IDEs [Albusays and Ludi 2016], [Albusays et al. 2017], [Zen et al. 2023a]; 5. Utilizar editor de código acessível para editar o design de páginas web [Potluri et al. 2019]; 6. Instalar complementos de acessibilidade, como o IndentNav para o NVDA [Kearney-Volpe and Hurst 2021]; 7. Utilizar a inteligência artificial para descrever imagens e auxiliar na criação de CSS [Kearney-Volpe and Hurst 2021]; 8. Usar extensões de acessibilidade como CodeTalk para o VS Code [Potluri et al. 2018]; 9. Utilizar a extensão Live Share do VS Code para revisão e refatoração de código e colaboração em equipe [Potluri et al. 2022]; 10. Utilizar o Addon Developer Toolkit do NVDA para desenvolvimento de Interface do Usuário [Pandey et al. 2021]; 11. Utilizar o Web Accessibilizer que auxilia as pessoas com deficiência visual a tornar as páginas HTML mais acessíveis bem como estudar seu layout [Pandey et al. 2022]; 12. Criar diagramas UML acessíveis com ferramentas como o PlantUML; 13. Explorar ferramentas que analisam diagramas UML, JSON e XML em estruturas de árvore [Kearney-Volpe and Hurst 2021]; 14. Utilizar o TangibleGrid para entender e projetar layouts de páginas web [Li et al. 2022].

Na sequência, o item 6 sugere instalar complementos de acessibilidade, como o IndentNav[6] para o NVDA [Kearney-Volpe and Hurst 2021]. Tal complemento foi testado na versão NVDA:2023.3.4, e a seguir, são apresentados os passos para instalação do mesmo: a) **Insert+n** para acessar o menu de configurações do NVDA; b) Clicar em ferramentas; c) Clicar em loja de complementos; d) Quando um aviso aparecer, clicar em "OK"; e) Ir com tab até guia e com setas direcionais esquerda/direita até encontrar "com-

[6]IndentNav - https://addons.nvda-project.org/addons/indentNav.en.html

plementos disponíveis"; f) ir com tab até encontrar a lista de complementos; g) Percorrer a lista com setas direcionais até encontrar o complemento IndentNav; h) Teclar tab até encontrar "Ações"; i) Clicar em instalar; j) Reiniciar o NVDA.

O item 8 orienta a usar extensões de acessibilidade como CodeTalk[7] para o VS Code [Potluri et al. 2018]. A seguir são apresentados exemplos de teclas de atalho do CodeTalk que foram testadas na versão 1.89.1 do VS Code:

- **Ctrl+ , ctrl+M**: Resumo do código;
- **Ctrl+ , ctrl+F**: Lista de funções;
- **Ctrl+ , ctrl+G**: Obter contexto;
- **Ctrl+ , ctrl+J**: Mover para o contexto;
- **Ctrl+ , ctrl+E**: Informação de erro;
- **Ctrl+ , ctrl+B**: TalkPoints.

Foi testada a extensão Live Share[8] do VS Code para revisão e refatoração de código e colaboração em equipe como sugere o item 9 [Potluri et al. 2022]. Essa extensão foi testada e validada na versão 1.89.1 do VS Code.

Os items 10, 11 e 12 envolvem a instalação e utilização de ferramentas específicas para prover acessibilidade, como é o caso do *Addon Developer Toolkit*[9] do NVDA para desenvolvimento de Interface do Usuário [Pandey et al. 2021]; do *Web Accessibilizer*[10] que é uma página feita em JavaScript que auxilia as pessoas com deficiência visual a tornar as páginas HTML mais acessíveis bem como estudar seu layout; e o PlantUML[11] para modelagem UML. Todas as tecnologias citadas estão disponíveis e funcionais, por outro lado, o TangibleGrid (item 14), voltado para entender e projetar layouts de páginas web [Li et al. 2022], não funcionou nos testes realizados.

3.4. Ferramentas Diversas

Além das ferramentas citadas anteriormente, pertencentes a grupos específicos de atividades, em quatro estudos foram identificados problemas relacionados a ferramentas específicas, apresentadas nesta seção. Tais trabalho apontam problemas na utilização das mesmas, porém não sugerem ações para mitigar tais situações (Quadro 4).

3.5. Ensino e materiais de aprendizagem

Outra categoria de extrema relevância apontada no MS é a que explora ferramentas e materiais para o ensino e aprendizagem. No Quadro 5 os problemas e as sugestões identificadas em três trabalhos são apresentados, não havendo dentre as sugestões, testes sugeridos.

[7]CodeTalk - https://www.microsoft.com/en-us/research/project/codetalk/
[8]Live Share - https://code.visualstudio.com/learn/collaboration/live-share
[9]Developer Toolkit NVDA - https://addons.nvda-project.org/addons/developerToolkit.en.html
[10]Disponível em: https://www.stsolution.org/WebAccessibilizer
[11]Disponível em: https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000

Quadro 4. Problemas e sugestões relacionadas a Ferramentas diversas

Problemas	Sugestões
1. Acessibilidade em Ferramentas de Gestão de Projetos como Trello e Jira; 2. Acessibilidade em Ferramentas de Comunicação como Slack, Skype e Microsoft Teams; 3. Problemas em construtores de GUI; 4. Problemas em Emuladores, como Android e iOS; 5. Identificar novos recursos após atualizações de ferramentas.	Não foram identificadas sugestões de melhoria para esta categoria.

Quadro 5. Problemas e sugestões relacionadas a Ensino e Aprendizagem

Problemas	Sugestões
1. Ensinar estrutura de dados (fluxogramas, matrizes, listas, árvores, pilhas, filas, tabelas hash e gráficos); 2. Materiais de aprendizagem não acessíveis; 3. Tutoriais em vídeo sem descrição; 4. Imagens de código sem textos alternativos; 5. Em aulas ou palestras, nem sempre o professor ou palestrante verbaliza o que escreve ou digita.	1. Criar site com materiais acessíveis [Kearney-Volpe and Hurst 2021]; 2. Solicitar documentações acessíveis e descritivas [Pandey et al. 2021]; 3. Utilizar livros didáticos digitais online [Crockett and Gannod 2020], [Zen et al. 2023a]; 4. Tornar as equações acessíveis utilizando ferramentas como: LATEX, MathType[12], ou EquatIO[13] [Crockett and Gannod 2020]; 5. Criar diagramas táteis usando o Microsoft Visio, converter a fonte para *Braille* e imprimir os diagramas utilizando impressora braile [Crockett and Gannod 2020]; 6. Ler a documentação oficial em busca de informações sobre acessibilidade [Pandey et al. 2022]; 7. Criar uma comunidade colaborativa de aprendizagem e prática [Kearney-Volpe and Hurst 2021].

3.6. Controle de Versões

No caso do grupo de controle de versões, somente um estudo abordou problemas, sem fornecer sugestões para tratá-los. A seguir, o Quadro 6 expõe os problemas elencados.

Quadro 6. Problemas e sugestões relacionadas a Controle de versões

Problemas	Sugestões
1. Visualizar e compreender operações complexas realizadas no controle de versões; 2. Identificar o autor de uma alteração específica no código-fonte; 3. Determinar quais componentes específicos foram modificados em uma revisão do código-fonte.	Não foram identificadas sugestões de melhoria para esta categoria.

4. Relato de experiência

Levando em consideração que a primeira autora deste trabalho é cega e cursa o 8º período do curso de Engenharia de Software, o presente trabalho apresenta, além do MS e das validações das sugestões realizadas pelos trabalhos mapeados, um relato com experiências.

A autora possui grande experiência na avaliação e uso de ferramentas computacionais, aplicadas no contexto de sua formação. Durante esta jornada, muitas dificuldades foram encontradas, e a falta de um trabalho como o apresentado aqui dificultou bastante o processo de adaptação dos conteúdos práticos do curso.

A seguir são compartilhadas algumas sugestões que a autora obteve durante sua trajetória acadêmica que podem ser úteis para outros estudantes, professores ou profissionais da área.

- Sobre a adaptação de materiais, sugere-se que os arquivos devem ser convertidos para um formato textual, como, por exemplo, txt, docx ou pdf. Além disso, as tabelas, gráficos e imagens devem ser descritos;
- No contexto do ensino de matemática, sugere-se o uso da ferramenta Multiplano;
- A modelagem UML, por ser uma tarefa majoritariamente visual, sugere-se a descrição detalhada em formato textual;
- Para disciplinas e conceitos relacionados a Bancos de Dados, a sugestão é criar desenhos em alto relevo, tais como: Diagrama Entidade Relacionamento (DER) e Diagrama de Tabelas do Modelo Relacional;
- Por fim, outra indicação é o uso de diferentes papéis táteis durante a criação de figuras para auxiliar no aprendizado do conceito de Gestalt em Interação Humano Computador (IHC).

5. Conclusão

Este trabalho apresentou os resultados de um MS que investigou problemas de acessibilidade que desenvolvedores cegos enfrentam ao utilizar ferramentas CASE. Foram investigados trabalhos publicados nos últimos 12 anos. Apenas 17 trabalhos foram selecionados, sugerindo que pouco se pesquisa sobre o tema e mostra a relevância da continuidade da pesquisa. Além disso, os resultados deste MS, identificaram alternativas que podem resolver, ou amenizar, esses problemas de acessibilidade. Para trabalhos futuros, pretende-se dar continuidade na pesquisa, organizar e compartilhar as descobertas em um website.

Foram identificados problemas e sugestões relacionados ao processo de desenvolvimento e as ferramentas CASE. Além disso, verificou-se se as sugestões realizadas nos trabalhos selecionados estão disponíveis e ainda não válidas para as ferramentas na atualidade. Por fim, um relato de experiências é apresentado, visando auxiliar estudantes e profissionais cegos, bem como professores que ministrem aulas para estudantes com essa deficiência.

Acredita-se que o presente trabalho apresente uma importante contribuição para a acessibilidade de ferramentas CASE, visto que coleta, organiza e analisa de forma prática a aplicabilidade de ferramentas alternativas para atividades profissionais na área de Ciência da Computação e Engenharia de Software.

Referências

Albusays, K. and Ludi, S. (2016). Eliciting programming challenges faced by developers with visual impairments: Exploratory study. In *2016 IEEE/ACM Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 82–85.

Albusays, K., Ludi, S., and Huenerfauth, M. (2017). Interviews and observation of blind software developers at work to understand code navigation challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '17, page 91–100, New York, NY, USA. Association for Computing Machinery.

Baker, C. M., Milne, L. R., and Ladner, R. E. (2015). Structjumper: A tool to help blind programmers navigate and understand the structure of code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 3043–3052, New York, NY, USA. Association for Computing Machinery.

Crockett, A. R. and Gannod, G. C. (2020). Improving understanding of data structures for the blind with tactile media and a user-centered iterative approach. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–8.

Ehtesham-Ul-Haque, M., Monsur, S. M., and Billah, S. M. (2022). Grid-coding: An accessible, efficient, and structured coding paradigm for blind and low-vision programmers. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, New York, NY, USA. Association for Computing Machinery.

Huff, E. W., Boateng, K., Moster, M., Rodeghero, P., and Brinkley, J. (2020). Examining the work experience of programmers with visual impairments. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 707–711.

Kearney-Volpe, C. and Hurst, A. (2021). Accessible web development: Opportunities to improve the education and practice of web development with a screen reader. *ACM Trans. Access. Comput.*, 14(2).

Li, J., Yan, Z., Jarjue, E. H., Shetty, A., and Peng, H. (2022). Tangiblegrid: Tangible web layout design for blind users. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, New York, NY, USA. Association for Computing Machinery.

Luque, L., Veriscimo, E. d. S., Pereira, G. d. C., and Filgueiras, L. (2014). Can we work together? on the inclusion of blind people in uml model-based tasks. In *Inclusive Designing*, pages 223–233. Springer.

Mealin, S. and Murphy-Hill, E. (2012). An exploratory study of blind software developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 71–74.

Nascimento, F. L. d., Barbosa, P. L. S., and Viana, W. (2022). Programando às cegas: investigando a acessibilidade de ambientes de desenvolvimento de software. In *Anais do VII Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software*, pages 1–10. SBC.

Pandey, M., Bondre, S., O'Modhrain, S., and Oney, S. (2022). Accessibility of ui frameworks and libraries for programmers with visual impairments. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–10.

Pandey, M., Kameswaran, V., Rao, H. V., O'Modhrain, S., and Oney, S. (2021). Understanding accessibility and collaboration in programming for people with visual impairments. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1).

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18.

Petrausch, V. and Loitsch, C. (2017). Accessibility analysis of the eclipse ide for users with visual impairment. *Studies in Health Technology and Informatics*, 242:922–929.

Potluri, V., He, L., Chen, C., Froehlich, J. E., and Mankoff, J. (2019). A multi-modal approach for blind and visually impaired developers to edit webpage designs. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '19, page 612–614, New York, NY, USA. Association for Computing Machinery.

Potluri, V., Pandey, M., Begel, A., Barnett, M., and Reitherman, S. (2022). Codewalk: Facilitating shared awareness in mixed-ability collaborative software development. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '22, New York, NY, USA. Association for Computing Machinery.

Potluri, V., Vaithilingam, P., Iyengar, S., Vidya, Y., Swaminathan, M., and Srinivasa, G. (2018). Codetalk: Improving programming environment accessibility for visually impaired developers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–11, New York, NY, USA. Association for Computing Machinery.

Pressman, R. S. and Maxim, B. R. (2021). *Engenharia de software*. Grupo A.

Silveira, S. R., Bertolini, C., da Cunha, G. B., Bigolin, N. M., and Steffens, R. (2019). Estratégias para apoiar os processos de ensino e de aprendizagem de alunos com deficiência visual: relato de experiências em um curso de bacharelado em sistemas de informação. *Redin-Revista Educacional Interdisciplinar*, 8(1).

Sommerville, I. (2011). *Engenharia de software*. Pearson, São Paulo.

Zen, E., Costa, V., and Tavares, T. (2023a). Experiências educacionais em disciplinas de programação de computadores: uma análise qualitativa na perspectiva dos estudantes com deficiência visual. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 960–971, Porto Alegre, RS, Brasil. SBC.

Zen, E., Tavares, T. A., and Costa, V. K. d. (2023b). Desafios e percepções sobre acessibilidade em ambientes de desenvolvimento integrado. *Revista Novas Tecnologias na Educação*, 21(2):244–253.