Ferramentas CASE: desafios sobre acessibilidade

CASE Tools: challenges on accessibility

Simone de Fátima Skura[1], Marlon Marcon[2], Franciele Beal[3]

RESUMO

O ciclo de desenvolvimento de software envolve quatro etapas fundamentais: especificação, projeto e implementação, validação e evolução. As ferramentas CASE são essenciais nesse processo, mas podem apresentar obstáculos significativos para desenvolvedores com deficiência visual devido a interfaces gráficas inacessíveis e falta de suporte a leitores de tela. Este trabalho enfatiza a necessidade de tornar essas ferramentas acessíveis e promover a inclusão de desenvolvedores com cegueira em todo o processo de desenvolvimento. Desse modo, foi realizado um mapeamento sistemático de estudos publicados entre os anos de 2012 e 2023, em bases nacionais e internacionais, sobre a acessibilidade de ambientes de desenvolvimento de software para cegos com o objetivo de identificar os desafios enfrentados por eles, as adaptações e as soluções recomendadas para melhorar, ou tornar, essas ferramentas mais inclusivas. Os resultados mostram uma pequena quantidade de trabalhos na área, sugerindo uma oportunidade de pesquisa a ser explorada, destacando a relevância deste trabalho.
PALAVRAS-CHAVE: Acessibilidade; Desenvolvedores de software cegos; Ferramentas CASE.

ABSTRACT

The software development cycle comprises four fundamental stages: specification, design and implementation, validation, and evolution. CASE tools are crucial in this process but can pose significant obstacles for visually impaired developers due to inaccessible graphical interfaces and a lack of screen reader support. This work emphasizes the need to make these tools accessible and promote the inclusion of visually impaired developers throughout the development process. Thus, a systematic mapping of studies published between 2012 and 2023, in both national and international databases, was conducted to explore the accessibility of software development environments for the blind. The aim was to identify the challenges they face, adaptations and recommended solutions to enhance or make these tools more inclusive. The results indicate a limited number of studies in this field, suggesting a research opportunity to be explored and underscoring the relevance of this work.
KEYWORDS: Accessibility; Blind software developers; CASE tools.

INTRODUÇÃO

O desenvolvimento de software envolve quatro etapas: especificação, projeto e implementação, validação e evolução. A especificação define funcionalidades e restrições, transformadas em software no projeto e implementação. Validação e teste garantem conformidade, enquanto a evolução lida com mudanças. Ferramentas CASE (sigla em inglês para *Computer-Aided Software Engineering*) são cruciais, incluindo criação de diagramas, IDEs (IDE, sigla em inglês para *Integrated Development Environment*), controle de versões, gestão de projetos e comunicação. Essas ferramentas aumentam

[1] Bolsista da Universidade Tecnológica Federal do Paraná, Dois Vizinhos, Paraná, Brasil E-mail: simoneskura@gmail.com. ID Lattes: http://lattes.cnpq.br/3005116662369099.
[2] Docente no Curso de Engenharia de Software/COENS-DV. Universidade Tecnológica Federal do Paraná, Dois Vizinhos, Paraná, Brasil. E-mail: marlonmarcon@utfpr.edu.br. ID Lattes: http://lattes.cnpq.br/3467949718089933.
[3] Docente no Curso de Engenharia de Software/COENS-DV. Universidade Tecnológica Federal do Paraná, Dois Vizinhos, Paraná, Brasil. E-mail: fbeal@utfpr.edu.br. ID Lattes: http://lattes.cnpq.br/0143431637515430.

a produtividade, facilitam o desenvolvimento e melhoram a qualidade do software (SOMMERVILLE, 2011);(PRESSMAN; MAXIM, 2021). Entretanto, estudos como os de (NASCIMENTO; BARBOSA; VIANA, 2022) e de (SILVEIRA et al., 2019) destacam que elas podem apresentar barreiras para desenvolvedores com deficiência visual, prejudicando sua produtividade e integração nos processos de desenvolvimento. As ferramentas CASE que utilizam interfaces gráficas de usuário (GUI sigla em inglês para *Graphical User Interface*) possuem elementos que não são legíveis pelos leitores de tela, uma das principais tecnologias assistivas utilizadas pelos usuários cegos (NVDA[1], JAWS[2] ou VoiceOver[3], p.ex.). Outra barreira é o mouse, que esses profissionais 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 eles (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.

METODOLOGIA DA PESQUISA

Este trabalho utilizou como base as diretrizes apresentadas por (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Inicialmente, foram definidas cinco (5) questões de pesquisa específicas para responder as questões apresentadas na introdução deste trabalho: 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?

Para responder as questões de pesquisa acima, foi definida a seguinte *string* de busca: *("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)*. Essa *string* foi executada no Google Scholar e retornou 57 resultados. Foram considerados trabalhos entre os anos 2012 e 2023. Para a seleção dos estudos foram aplicados critérios de inclusão e exclusão. Critérios de Inclusão: a) Estudos sobre a acessibilidade de ambientes de desenvolvimento de software para desenvolvedores cegos, tanto em meio profissional ou acadêmico; b) Estudos nos idiomas inglês e português; c) Estudos gratuitos. Critérios de exclusão: a) Não se tratar de um artigo (sumários, índices, etc.); b) Estudos duplicados; c) Estudos não disponíveis em texto completo; d) Estudos na forma de dissertações, teses e revisões de literatura; e) Estudos que não atendem os critérios de inclusão.

[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

Para a seleção dos estudos foram executadas 3 etapas: na primeira, foram lidos os títulos, resumos e palavras-chave e foram selecionados 29 estudos dos 57. Na segunda etapa, os estudos selecionados foram enviados para o Núcleo de Acessibilidade e Inclusão (NAI) da Universidade para serem adaptados, pois a 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 15 estudos para responder as questões de pesquisa deste trabalho.

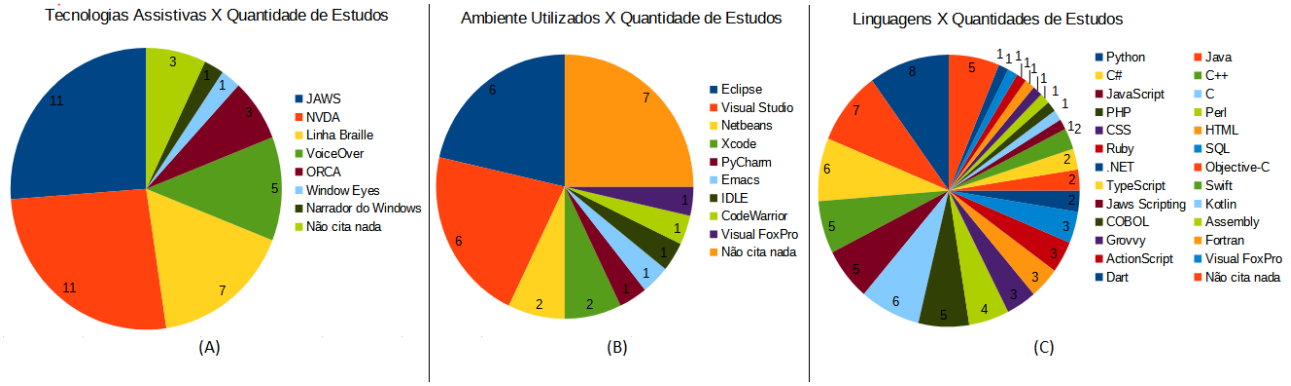
RESULTADOS E DISCUSSÕES

Segundo os dados extraídos, foram publicados 15 estudos entre os anos de 2012 e 2022. Destes, 13 foram publicados após 2017, sendo 4 deles em 2022, o que indica pesquisas recentes, tema atual e relevante.

Este MS teve como objetivo responder as questões de pesquisa apresentadas na seção anterior. Desta forma, as respostas das questões QP1, QP2 e QP3 são apresentadas na Figura 1.

Para responder a QP1, dos 15 estudos selecionados, apenas 12 estudos informaram as tecnologias assistivas usadas pelos usuários. A Figura 1(A) mostra o gráfico das tecnologias citadas e a quantidade de artigos que citou cada uma delas. As tecnologias NVDA e JAWS foram citadas em 11 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 1 – Respostas das Questões de Pesquisas 1, 2 e 3.



Fonte: próprio autor.

A Figura 1(B) apresenta a resposta da QP2 sobre os ambientes de desenvolvimento utilizados pelos cegos. De 15 trabalhos, apenas 8 estudos informaram os ambientes de desenvolvimento usados pelos participantes. O Eclipse foi citado por 6 estudos, o Visual Code 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. Já a Figura 1(C), apresenta a resposta da QP3. Dos 15 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.

A QP4 abordou sobre os principais problemas de acessibilidade apontados pelos estudos tanto em meio profissional quanto acadêmico apresentados a seguir:

- **Escrita de Código e Navegação -** dificuldades identificadas em 11 estudos: Identificar corretamente os espaços em branco usados na indentação do código; Identificar o nível de aninhamento ao navegar por estruturas de código aninhadas; Retornar para uma linha específica em um código extenso ao revisar instruções de código em outros arquivos; Identificar a falta ou excesso de parênteses, o que pode levar a erros de sintaxe; Acessar e usar eficientemente o recurso de autocompletar; Identificar números de linha no código. Identificar relacionamentos entre classes e subclasses; Localizar erros e avisos em tempo real durante a codificação; Navegar pelo código usando o teclado; Dificuldades em Colaborar na programação em pares devido à falta de acesso à tela do colega; Além disso, identificar o cursor do colega que enxerga.
- **Depuração de Código -** dificuldades identificadas em 9 estudos: Navegar e utilizar as ferramentas de depuração, exemplo de ferramentas citadas: FindBugs e Firebug; Compreender o fluxo de execução; Encontrar erros no código; Identificar o progresso durante a depuração; Acessar o status dos pontos de interrupção; Ler o código linha a linha para identificar erros; Mudar da tela de erro para a edição do código; Usar a função *printf()* como solução alternativa, porém, esse método é demorado e ineficaz.
- **Ambientes de Desenvolvimento (IDE) -** dificuldades identificadas em 8 estudos: Acessibilidade no Eclipse e Visual Studio Code (VS Code); Tarefas relacionadas a componentes visuais, como arquitetura do programa e layouts de interface de usuário; Falta de acessibilidade em componentes de interface do usuário, componentes inacessíveis; Falta de confiança para executar tarefas relacionadas ao design de interface de usuário; Interpretar layouts de interface; Inspecionar o próprio trabalho de design; Escrever e determinar medidas dos componentes em arquivos CSS; 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.
- **Ferramentas Diversas -** dificuldades identificadas em 4 estudos: Acessibilidade em Ferramentas de Gestão de Projetos como Trello e Jira; Acessibilidade em Ferramentas de Comunicação como Slack, Skype e Microsoft Teams. Problemas em construtores de GUI. Problemas em Emuladores, como Android e iOS. Identificar novos recursos após atualizações de ferramentas.
- **Diagramas -** dificuldades identificadas em 3 estudos: Falta de acessibilidade em diagramas, como UML (*Unified Modeling Language*) e arquitetura de software.
- **Ensino e materiais de aprendizagem -** dificuldades identificadas em 2 estudos: Ensinar estrutura de dados (fluxogramas, matrizes, listas, árvores, pilhas, filas, tabelas hash e gráficos); Materiais de aprendizagem não acessíveis; tutoriais em vídeo sem descrição; Imagens de código sem textos alternativos.
- **Controle de Versões -** dificuldades identificadas em 1 estudo: Visualizar e compreender operações complexas realizadas no controle de versões; Identificar o autor de uma alteração

específica no código-fonte; Determinar quais componentes específicos foram modificados em uma revisão do código-fonte.

Por fim, é apresentada a resposta da QP5, sobre as alternativas sugeridas para resolver, ou amenizar, os problemas de acessibilidade.

- **Escrita de Código, Navegação e Depuração:** Utilizar a linha braille. Utilizar sistemas de recomendação de comandos. Utilizar ferramentas de navegação de código. Pesquisar em código-fonte utilizando palavras-chave. Utilizar a navegação por bloco em linguagens que utilizam a indentação, como Python. Escrever script para forçar o leitor de tela a ler a quantidade de espaços e não a palavra "espaço". Utilizar editores de texto para fazer anotações enquanto utiliza as IDEs. Escrever script que informe o número de uma linha ou execute outras funções necessárias. Utilizar teclas de atalho no IDE Eclipse, tais como: 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; e, CTRL+SHIFT+Seta para cima/para baixo para ir para o próximo método. Criar um recurso de navegação hierárquica para que o usuário não precise fazer a leitura do código de forma linear (Tree View). Criar *feedbacks* sonoros. Adicionar marcadores ou tags nos códigos. Criar indicadores sonoros de alinhamento e escopo. Criar documentação contendo as teclas de atalho das ferramentas. Utilizar o plugin Structjumper para o Eclipse para facilitar a navegação e compreensão do código Java. Adicionar comentários descritivos, usar maiúsculas e minúsculas e nomes longos de variáveis. Utilizar na depuração os seguintes atalhos do Eclipse: CTRL+F7 para pular para a lista de erros, ou, CTRL+./CTRL+, para percorrer uma lista de erros ou avisos. Solicitar o envio prévio de código antes de reuniões para revisão.

- **Ambientes de Desenvolvimento (IDE) e Diagramas:** Utilizar atalhos acessíveis do Eclipse, tais como: CTRL+SHIFT+L para listar todos os atalhos no Eclipse e F10 para definir foco no menu. Instalar plugins ou extensões de acessibilidade, como o ACTF (*Accessibility Tools Framework*) do Eclipse. Criar avisos sonoros para dar *feedbacks* no Eclipse. Instalar complementos de acessibilidade, como o IndentNav para o NVDA. Utilizar a inteligência artificial para descrever imagens e auxiliar na criação de CSS. Utilizar frameworks como DOJO e Bootstrap para facilitar o desenvolvimento de GUI. Usar extensões de acessibilidade como CodeTalk para o VS Code. Utilizar a extensão Live Share do VS Code para revisão e refatoração de código e colaboração em equipe. Utilizar o *Addon Developer Toolkit* do NVDA para desenvolvimento de Interface do Usuário. Utilizar o TangibleGrid para entender e projetar layouts de páginas web. Criar diagramas UML acessíveis com ferramentas como o PlantUML.

- **Ensino e Materiais de Aprendizagem:** Criar site com materiais acessíveis. Solicitar documentações acessíveis e descritivas. Utilizar livros didáticos digitais online. Tornar as equações acessíveis utilizando ferramentas como: LATEX, MathType , ou EquatIO. Criar diagramas táteis usando o Microsoft Visio com a fonte *New Courier* de 24 pontos, em seguida, converter a fonte para *BrailleKiama True Type Font* sem pontos de sombra e imprimir os diagramas utilizando a

impressora TactPlus (impressora braille) e papel Swell (papel para a produção de imagens em relevo). Ler a documentação oficial em busca de informações sobre acessibilidade. Criar uma comunidade colaborativa de aprendizagem e prática.

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 11 anos. Apenas 15 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.

Agradecimentos

A estudante bolsista voluntária agradece a UTFPR-DV pela oportunidade e aprendizado.

Conflito de interesse

Não há conflito de interesse.

REFERÊNCIAS

LUQUE, L et al. Can we work together? on the inclusion of blind people in uml model-based tasks. In: INCLUSIVE Designing. [S.l.]: Springer, 2014. P. 223–233.

NASCIMENTO, Felipe Lúcio do; BARBOSA, Pedro Luis Saraiva; VIANA, Windson. Programando às cegas: investigando a acessibilidade de ambientes de desenvolvimento de software. In: SBC. ANAIS do VII Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software. [S.l.: s.n.], 2022. P. 1–10.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. **Information and Software Technology**, v. 64, p. 1–18, 2015.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software**. [S.l.]: Grupo A, 2021.

SILVEIRA, Sidnei Renato et al. 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**, v. 8, n. 1, 2019.

SOMMERVILLE, Ian. **Engenharia de software**. São Paulo: Pearson, 2011.