CÓDIGO 31

EXPERTISE DO DESENVOLVEDOR E A INTERDISCIPLINARIDADE

DEVELOPER EXPERTISE AND INTERDISCIPLINARITY

Guilherme Henrique de Assis[1]
guilhenrique.assis@gmail.com

Amanda Damasceno de Souza[2]
amanda.dsouza@fumec.br

Data de Submissão: 20/09/2022.
Data de Aprovação: 11/11/2022.

RESUMO

Determinar a expertise de um desenvolvedor é uma tarefa importante em diversos cenários, por exemplo durante a contratação de um novo desenvolvedor ou durante a alocação de tarefas. Apesar da importância desta tarefa, ela não é trivial. Este trabalho tem por objetivo discutir as relações entre uma pesquisa no âmbito da Pós-graduação em Sistemas de Informação e Gestão do Conhecimento e a Interdisciplinaridade entre a Ciência da Computação e Ciência da Informação, focando em algumas abordagens teóricas. A pesquisa tem por objetivo propor uma ferramenta para medição da expertise de um desenvolvedor em frameworks específicos. Este trabalho inicialmente contextualiza os conceitos de software, framework e expertise. Depois são discutidos aspectos relativos à tarefa de determinar a expertise de um desenvolvedor e a integração entre áreas, incluindo as relações da Ciência da Computação com a Interdisciplinaridade. Por fim são discutidas as relações da ferramenta a ser proposta com teorias da Interdisciplinaridade, mais especificamente a Cibernética de Primeira e Segunda Ordem e a Teoria Geral dos Sistemas.

Palavras-chave: expertise; interdisciplinaridade; desenvolvedor; ciência da computação.

ABSTRACT

Determining a developer's expertise is an important task in many scenarios, such as hiring a new developer or allocating tasks. However, despite the importance of this task, it is not trivial. This work aims to discuss the relationship between a research within the Postgraduate Program in Information Systems and Knowledge Management and the Interdisciplinarity between Computer Science and Information Science, focusing on some theoretical approaches. The research proposes a tool for measuring a developer's expertise in specific frameworks. This work initially contextualizes the concepts of software, framework, and expertise. Then, aspects related to determining a developer's expertise and the integration between areas are discussed, including the relationship between Computer Science and Interdisciplinarity. Finally, the relations of the tool to be proposed with theories of Interdisciplinary are discussed, more specifically First and Second-Order Cybernetics and General Systems Theory.

Keywords: *expertise; interdisciplinarity; developer; computer science.*

1 Programa de Pós-Graduação em Sistemas de Informação e Gestão do Conhecimento da Universidade FUMEC – PPGSIGC
https://orcid.org/0000-0001-8904-3944
guilhenrique.assis@gmail.com

2 Programa de Pós-Graduação em Sistemas de Informação e Gestão do Conhecimento da Universidade FUMEC – PPGSIGC
https://orcid.org/0000-0001-6859-4333
amanda.dsouza@fumec.br

1 INTRODUÇÃO

A presente pesquisa tem por objetivo discutir as relações entre uma pesquisa no âmbito da Pós-graduação em Sistemas de Informação e Gestão do Conhecimento e a Interdisciplinaridade entre a Ciência da Computação e Ciência da Informação, focando em algumas abordagens teóricas.

Saracevic (1996, p. 50) aborda que "a relação entre a Ciência da Informação e a Ciência da Computação reside na aplicação dos computadores e da computação na recuperação da informação, assim como nos produtos, serviços e redes associados". Enquanto a Ciência da Informação trata da informação e sua comunicação para uso pelos humanos a Ciência da Computação aborda os algoritmos que transformam informações. A Ciência da Informação é um campo de pesquisa e de prática profissional de natureza interdisciplinar e ligada à tecnologia da informação (SARACEVIC, 1996).

Neste contexto, a pesquisa consiste em estudar e propor técnicas que auxiliem na medição da expertise de um desenvolvedor de software, seja para servir de apoio durante a contratação de um desenvolvedor ou para apoiar o processo de Engenharia de Software dentro de um projeto, realizando uma melhor alocação de tarefas. A ideia é que, a partir da produção realizada pelo próprio desenvolvedor ao longo da sua carreira ou em um projeto específico, seja possível indicar qual o nível de conhecimento do desenvolvedor em determinada tecnologia. Antes de iniciar a discussão das relações do trabalho com a Interdisciplinaridade, faz-se uma breve introdução sobre conceitos importantes deste estudo.

Este estudo trata-se de pesquisa com abordagem metodológica exploratória, uma vez que objetiva proporcionar maior familiaridade com o problema. Do ponto de vista dos procedimentos técnicos, envolve levantamento bibliográfico elaborado a partir de material já publicado (GIL, 1994).

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Software

Segundo Sommerville (2011), o termo software não se refere somente a um programa de computador, mas também a toda documentação e configuração associados, que são necessários para o seu correto funcionamento. Ainda segundo Sommerville (2011), o conceito de Engenharia de Software foi inicialmente proposto em 1968 durante uma conferência para discutir a chamada crise do software, pois o desenvolvimento informal não estava sendo suficiente, já que projetos estavam atrasando o planejamento inicial, com custos extrapolando o previsto e resultados insatisfatórios. Neste cenário surgiu a Engenharia de Software, que segundo Sommerville (2011) é um ramo da engenharia cujo foco é o desenvolvimento dentro de custos adequados de sistemas de software de alta qualidade.

Apesar dos constantes avanços no estado da arte do desenvolvimento de software, esta continua sendo uma tarefa desafiadora e que exige um grande nível de conhecimento. Os desenvolvedores são continuamente expostos a novas tecnologias, componentes e ideias (ROBILLARD; WALKER; ZIMMERMANN *et al.*, 2009). A área de desenvolvimento de software está crescendo rápido e sendo utilizada em diferentes cenários, aumentando consequentemente a necessidade de desenvolvedores de software (JAVEED *et al.*, 2020).

Conhecer as habilidades do desenvolvedor é essencial para determinar a sua eficiência e se ele consegue trabalhar bem em determinada equipe. As habilidades de um desenvolvedor podem ser divididas em habilidades técnicas, como conhecimento de programação e qualidade do trabalho, e habilidades sociais, como capacidade de colaboração, habilidades de gerenciamento de projetos e motivação (SARMA *et al.*, 2016). A respeito dos conhecimentos técnicos, um dos pontos

que impactam a produtividade e a qualidade do software desenvolvido é a experiência técnica do desenvolvedor e seu conhecimento (JAVEED *et al.*, 2020).

2.2 Framework

No contexto de desenvolvimento de software, um framework é um grupo de código compartilhado que possui funcionalidade genérica e que os desenvolvedores podem reutilizar em seu código para acelerar o processo de desenvolvimento (EDWIN, 2014). Os frameworks prometem maior produtividade e menor tempo para entrega dos projetos através da reutilização de arquitetura e de código (RIEHLE, 2000).

Edwin (2014) descreve quatro características dos frameworks: comportamento padrão, inversão de controle, extensibilidade e código do framework não modificável. Comportamento padrão significa que o framework funcionará de maneira padrão caso não ele seja customizado. Inversão de controle significa que o fluxo de controle é definido pelo framework e não pelo código do desenvolvedor que o executa. Extensibilidade significa que um usuário pode customizar partes do framework, substituindo o código padrão pelo seu próprio em locais pré-determinados. Por fim, código do framework não modificável indica que um usuário pode customizar partes do framework, mas não pode modificá-lo, pois, o objetivo do framework é facilitar o processo de desenvolvimento, evitando lidar com as responsabilidades do framework.

2.3 Expertise

O termo expert é definido por Merriam-Webster (2022) como "aquele com habilidade ou conhecimento especial, possuindo o domínio de um assunto específico", e também como "possuir ou mostrar habilidade especial ou conhecimento derivado de treinamento ou experiência" (EXPERT, 2022). Em várias áreas como esportes individuais, a expertise pode ser medida mais objetivamente usando métricas de performance, mas em outros domínios esta medida é mais difícil (ERICSSON; TOWNE, 2010).

Expertise não é somente possuir um talento, mas sim o resultado obtido da dedicação a uma área específica (CROSS *et al.*, 2004). Para ser um expert em alguma área, é necessário um tempo mínimo de prática e envolvimento para alcançar altos níveis de desempenho (ERICSSON; TOWNE, 2010).

Por definição, experts possuem mais conhecimento relevante de um domínio específico do que novatos, porém este não é o único fator importante, mas também a forma como seu conhecimento é organizado, se tornando mais acessível, funcional e eficiente (BEDARD; CHI, 1992). Experts tendem a basear a organização do seu conhecimento no significado, possuindo maiores e mais fortes relações entre conceitos, enquanto os novatos fazem isso de forma superficial de acordo com as informações apresentadas (BEDARD; CHI, 1992).

Expertise não pode ser confundida com experiência, pois um indivíduo pode ter experiência considerável, mas mesmo assim não ser um expert, da mesma forma que pessoas com níveis similares de expertise podem ter diferentes níveis de experiência, e vice-versa (JACOBY *et al.*, 1986).

3 DETERMINANDO A EXPERTISE DE UM DESENVOLVEDOR

Determinar a expertise de um desenvolvedor é essencial em várias situações, como por exemplo durante um processo de contratação ou para realizar alocação de tarefas dentro da equipe (MORADI DAKHEL; DESMARAIS; KHOMH *et al.*, 2021). A expertise em desenvolvimento de

software pode ser medida considerando diferentes aspectos. Baltes e Diehl (2018) definiu os seguintes conceitos que podem ser usados para determinar a expertise de um desenvolvedor: experiência, conhecimento, qualidade do código desenvolvido, habilidades gerais e contexto do trabalho.

Experiência é relacionada a quantidade e qualidade. Quantidade refere-se ao tempo de experiência, como por exemplo quantos anos de experiência como desenvolvedor. Qualidade refere-se à profundidade do conhecimento durante o tempo de experiência, por exemplo se o desenvolvedor trabalhou somente em projetos pequenos ou também em projetos mais complexos.

Conhecimento refere-se a um conhecimento específico em uma tecnologia de acordo com a sua profundidade. Podendo variar entre profundo ou superficial, além de também dizer respeito a amplitude, se ele é mais específico ou é mais amplo, por exemplo, se possui conhecimentos em algoritmos, estruturas de dados e paradigmas de programação.

Sobre a qualidade do código fonte, Baltes e Diehl (2018) aponta algumas características que podem ajudar a determinar um expert, como por exemplo se o código é bem estruturado, legível, se tem bom desempenho e se é fácil de manter. Sobre as habilidades gerais, Baltes e Diehl (2018) cita habilidades de comunicação como principal exemplo, permitindo que os desenvolvedores compartilhem seus conhecimentos com a equipe e peçam ajuda quando necessário.

Por fim, contexto do trabalho significa o desenvolvedor conseguir lidar com diferentes situações, como por exemplo relações entre pessoas, pressão por prazos apertados e falta de requisitos bem definidos. A expertise do desenvolvedor pode ser medida qualitativamente ou quantitativamente, indicando as habilidades e os níveis de profundidade delas, permitindo que ela seja aplicada em vários campos, como por exemplo para realizar a alocação eficaz de tarefas (YAN *et al.*, 2018).

3.1 Integração entre áreas de estudo

Segundo Prikladnicki *et al.* (2008), tem-se percebido cada vez mais que, para enfrentar determinados desafios encontrados pela Engenharia de Software. É necessário o estudo e a integração entre diversas disciplinas e áreas do conhecimento, por exemplo a Ciência da Computação, Administração, Educação, Psicologia, Sociologia, Linguística, Engenharia de Produção, Ciência da Informação, entre outras, pois o software é complexo, modificável e abstrato, desafiando constantemente a capacidade humana.

A Ciência da Informação e a Ciência da Computação demandam operação conjunta pois são campos de saber que abordam o mesmo objeto, a informação (CAFEZEIRO; COSTA; KUBRUSLY, 2016). Davenport e Prusak (1998) explica as diferenças entre dado, informação e conhecimento. Dado é um conjunto de fatos distintos e objetivos relacionados a eventos. Em um contexto organizacional, os dados são descritos como registros estruturados de transações. Informação é uma mensagem, geralmente em um documento ou uma comunicação audível ou visível, e esta mensagem tem um emissor e um receptor. O objetivo da informação é mudar a forma como o destinatário vê algo, e é ele quem decide se essa mensagem recebida é informação ou não. Dado se transforma em informação quando seu criador acrescenta significado a ele.

Avaliar a experiência do desenvolvedor pode ser analisado como transformar dados em informações. Diversos estudos analisam dados brutos produzidos pelo desenvolvedor, por exemplo, código-fonte, e geram informações a partir dele, classificando o desenvolvedor de acordo com seu nível de especialização. Por exemplo, Moradi Dakhel, Desmarais e Khomh (2021) analisa o código fonte

dos desenvolvedores visando identificar características que auxiliem na avaliação da expertise.

Podemos dizer que a Ciência da Computação, suas ferramentas e subáreas, apesar de ser considerada uma ciência exata, é naturalmente interdisciplinar, pois grande parte dos problemas que ela busca resolver através de suas soluções e tecnologias são problemas, na maioria das vezes, oriundos de outras áreas. Dentre as abordagens teóricas da Interdisciplinaridade, foram selecionadas três para que pudéssemos traçar algumas relações entre o tema estudado e as teorias. As selecionadas foram a Cibernética de Primeira, a Cibernética de Segunda Ordem e a Teoria Geral dos Sistemas. Antes de traçarmos as relações, na próxima seção será realizada uma rápida discussão sobre o que são esses conceitos.

3.2 A Ciência da Computação e a Interdisciplinaridade

Antes de citar a Interdisciplinaridade, vamos definir o que seria a Multidisciplinaridade e a Pluridisciplinaridade. Segundo Sommerman (2006), o multidisciplinar se preocupa com um aspecto mais numérico e quantitativo, sem necessariamente existir relações entre as disciplinas, enquanto o pluridisciplinar se refere a existência de relações complementares entre disciplinas mais ou menos afins. Ainda segundo Sommerman (2006), a Interdisciplinaridade é um passo adiante nessa relação, se referindo a interação entre duas ou mais disciplinas, podendo ir da simples troca de ideias até a integração de conceitos, podendo até mesmo gerar um novo corpo disciplinar.

A área da Ciência da Computação, por ser uma área interdisciplinar, possui várias técnicas que foram inspiradas em conceitos vindos de outras áreas. Por exemplo, na área da Inteligência Artificial temos o conceito de rede neural, que foi inspirado no funcionamento de um neurônio humano e na relação de conexão entre eles. Cada neurônio realiza uma pequena parte de um processamento maior, no qual cada um fornece entradas para os próximos e assim por diante, sendo capazes processar um grande volume de informações e de aprender determinados comportamentos de acordo com os dados encontrados, conseguindo agir no futuro baseado no que lhe foi apresentado previamente. Também temos os algoritmos genéticos, que foram baseados em conceitos de evolução da biologia, a partir dos quais os algoritmos conseguem gerar novas instâncias de elementos, inspirados em conceitos como mutação, seleção natural e recombinação.

A Interdisciplinaridade possui várias teorias, dentre elas três foram escolhidas para terem suas relações com o tema do trabalho apresentado no início discutidas, a Cibernética de Primeira Ordem, a Cibernética de Segunda Ordem e a Teoria Geral dos Sistemas. Segundo Heylighen e Joslyn (2001), a Cibernética é a ciência que estuda os princípios abstratos da organização de sistemas complexos, não se preocupando muito em como são compostos, mas sim como eles funcionam, como utilizam informações e modelos mantendo seu objetivo principal.

A Cibernética de Primeira Ordem foi originada em uma série de reuniões interdisciplinares ocorridas entre 1944 e 1953, conhecidas como *Macy Conferences on Cybernetics*. Durante os anos de 1950, os pensadores da Cibernética atuaram em conjunto com os da Teoria Geral dos Sistemas, que focavam o estudo em sistemas de todos os níveis de generalidade, enquanto os da Cibernética tinham um foco maior nos sistemas direcionados a objetivos e sistemas que tinham uma relação de controle (HEYLIGHEN; JOSLYN, 2001).

Segundo Glanville (2002), a Cibernética de Segunda Ordem, que também é conhecida como Cibernética das Cibernéticas ou como a Nova Cibernética, foi desenvolvida entre 1968 e 1975 por estudiosos, entre eles Heinz von Foerster,

em reconhecimento ao poder e consequências dos estudos da cibernética sobre a circularidade. Ainda segundo Glanville, (2002), na Cibernética de Segunda Ordem o observador é incluído no estudo, sendo circularmente conectado com o observado. Ele ainda cita um exemplo do termostato para explicar o conceito da circularidade, mostrando que os dois componentes que compõe um termostato, o aquecedor e o sensor, possuem uma relação circular na qual o sensor faz o aquecedor ligar ou desligar de acordo com a temperatura, mas por outro lado o aquecedor também faz o sensor ligar ou desligar, não existindo assim uma primeira causa.

Segundo Araújo e Gouveia (2016), a Teoria Geral dos Sistemas começou a ser estudada por Ludwig von Bertalanffy e buscava um modelo científico explicativo do comportamento de um organismo vivo, tratando-se de uma teoria interdisciplinar aplicada a diversas áreas. No mesmo artigo, Araújo e Gouveia (2016) cita o conceito de sistema, como sendo um conjunto de elementos ou componentes que interagem para se atingir os objetivos. Entre as várias definições da Teoria Geral dos Sistemas, podemos destacar, o conceito de sistema aberto, como sendo aquele que interage com o exterior, processando as trocas de informação através de entradas e saídas, também chamadas de inputs e outputs.

3.3 Relações da Ferramenta com a Interdisciplinaridade

Conforme descrito no início do texto, a pesquisa tem por objetivo estudar e propor uma ferramenta que auxilie na medição da expertise de um desenvolvedor de software, indicando qual seu nível de conhecimento em determinado contexto ou tecnologia. Para isso, utiliza-se o histórico de alterações em código realizado pelo próprio desenvolvedor ao longo da sua carreira ou em um projeto específico. Este histórico é possível de obter, já que a maioria das aplicações utiliza os chamados Sistemas de Controle de Versões, ou *Version Control Systems* (VCS), nos quais ficam armazenados todas as alterações do projeto ao longo do tempo, juntamente com quem foi o responsável.

De forma mais específica, a ferramenta visa medir a expertise de um desenvolvedor em um framework em específico. Para configurar um framework na ferramenta, é preciso fornecer um conjunto de comandos que caracterizam a utilização daquele framework. A ferramenta realiza uma busca projetos que utilizem o framework em questão em um VCS de código aberto, o GitHub, e depois calcula algumas métricas de utilização, gerando assim uma base de comparação.

Com a base de comparação pronta, a ferramenta está apta a analisar o perfil do GitHub de desenvolvedores, e comparar a sua utilização do framework com os demais. A ferramenta gera um relatório contendo métricas de utilização daquele framework pelo desenvolvedor, de forma geral e detalhada por comandos, posicionando-o em relação a base de comparação gerada anteriormente.

Em outras palavras, a ferramenta é um software que analisa software, representado pelo seu código fonte. Pensando desta forma, podemos notar uma relação direta com a Cibernética de Segunda Ordem, pois o próprio software é utilizado para estudar ele mesmo e o seu autor. Também podemos notar a presença do conceito da auto-referência da Cibernética de Segunda Ordem, já que a estabilidade do sistema vem de dentro dele mesmo, com ele se sustentando sozinho. Isso ocorre no trabalho, pois para que a técnica a ser proposta possa avaliar o nível de conhecimento do desenvolvedor naquele contexto, são necessários dados anteriores a respeito de alterações já realizadas, como o que foi alterado no código-fonte do sistema ao longo do tempo. Ou seja, a

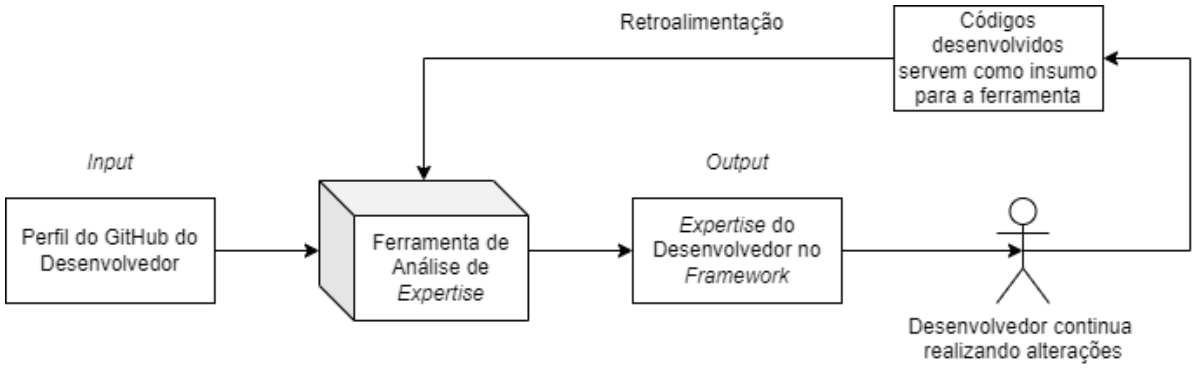
própria evolução natural do software é o insumo necessário para que a ferramenta consiga operar, e ele se retroalimenta, sendo capaz de se sustentar sozinho e de ir aprimorando seus resultados com o decorrer do tempo.

Um dos conceitos desenvolvidos na teoria da Cibernética de Primeira Ordem foi, segundo Gomes *et al.* (2014), o de feedback, ou também chamado de retroalimentação, e que consiste na circulação de informações entre elementos do sistema, de forma que o sistema é retroalimentado por informações dele próprio.

Neste cenário, podemos traçar uma relação com o tema da pesquisa, pois na ferramenta as próprias informações geradas ao longo do desenvolvimento de um sistema, como por exemplo os artefatos de código modificados, são utilizados como insumos para as avaliações e comparações. Ou seja, o sistema é retroalimentado, fazendo uso assim do conceito de feedback proveniente da Cibernética.

Também podemos encontrar relações do tema estudado com a Teoria Geral dos Sistemas. Inicialmente podemos citar a característica da ferramenta, que exige um conjunto de entradas, também chamado de input, e após determinado processamento devolve um conjunto de saídas, também chamado de output. O input é o histórico de alterações em código fonte do desenvolvedor, enquanto o output é a avaliação do conhecimento do desenvolvedor naquele contexto. A medida que o desenvolvedor realiza mais alterações, o sistema pode ser retroalimentado para realizar a atualização de sua expertise. A figura 1 abaixo ilustra esse funcionamento.

Figura 1 – Funcionamento da ferramenta



Fonte: Elaborada pelos autores.

4 CONSIDERAÇÕES FINAIS

A Ciência da Informação, a Ciência da Computação e a Engenharia de Software como subárea, são interdisciplinares por vários motivos. Um software é desenvolvido visando a solução de determinado problema. Esses problemas podem estar em qualquer área do conhecimento, sendo assim, essencial a interação entre áreas para que a solução elaborada seja mais aderente possível ao problema em questão. A equipe envolvida na concepção de um software, desde o levantamento inicial dos requisitos, a análise das funcionalidades, passando pelo desenvolvimento, testes, implantação final e até a entrega para o cliente, exige um conhecimento interdisciplinar para que o sucesso seja atingido.

Para que a ferramenta seja utilizada com sucesso em uma organização, é necessário a interação de demais pessoas de demais áreas, e não somente de desenvolvedores. Isso ocorre, pois quem irá utilizar a ferramenta na maioria das vezes é alguém que está avaliando a expertise do desenvolvedor em determinada tecnologia, seja para um processo seletivo ou para a alocação de tarefas, e esse processo nem sempre é realizado por um desenvolvedor, podendo ser realizado por uma pessoa de recursos humanos ou por um gerente de projetos, por exemplo. Ou seja, isso pode ser realizado por pessoas de diferentes equipes e diferentes disciplinas, não ficando limitados a Ciência da Computação. Neste cenário, as teorias da Interdisciplinaridade que foram destacadas, em especial a Cibernética de Primeira e Segunda Ordem e a Teoria Geral dos Sistemas, fornecem uma base teórica essencial, já que possuem ligações históricas com o tema estudado.

REFERÊNCIAS

ARAÚJO, A. C. M.; GOUVEIA, L. B. Uma revisão sobre os princípios da teoria geral dos sistemas. **Revista Estação Científica**, n. 16, p.1-14, jul./dez. 2016.

BALTES, S.; DIEHL, S. Towards a theory of software development expertise. *In*: PROCEEDINGS OF THE 2018 26TH ACM JOINT MEETING ON EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING, Lake Buena Vista FL USA November 4 - 9, 2018. Proceeding... Lake Buena Vista : ESEC/FSE, 2018.p. 187-200. Disponível em: https://dl.acm.org/doi/proceedings/10.1145/3236024. Acesso em 11 nov 2022.

BEDARD, J.; CHI, M. T. Expertise. **Current directions in psychological science**, v. 1, n. 4, p.135-139, 1992.

CROSS, N. Expertise in design: an overview. **Design studies**, v. 25, n. 5, p.427-441, 2004.

CAFEZEIRO, I.; COSTA, LC.; KUBRUSLY, R. S. Ciência da Computação, Ciência da Informação, Sistemas de Informação: uma reflexão sobre o papel da informação e da interdisciplinaridade na configuração das tecnologias e das ciências. **Perspectivas em Ciência da Informação** [online], v. 21, n. 3, p. 111-133, 2016. Acesso em: 8 maio 2022. Disponível em: https://doi.org/10.1590/1981-5344/2681.

DAVENPORT, T. H.; PRUSAK, L. **Working knowledge**: How organizations manage what they know. Boston: Harvard Business Press, 1998.

EDWIN, N. M. Software frameworks, architectural and design patterns. **Journal of Software Engineering and Applications**, v. 7, n. 8, p. 670-678, 2014.

ERICSSON, K. A.; TOWNE, T. J. Expertise. **WIREs Cognitive Science**, n. 3, p. 404-416, 2010.

EXPERT. *In:* MERRIAM-WEBSTER dictionary. Springfield: Merriam-Webster Inc, 2022. Disponível em: https://www.merriam-webster.com/dictionary/expert . Acesso em: 5 maio 2022.

GLANVILLE, R. Second order cybernetics. **Systems Science and Cybernetics**, v.3, p. 59-85, 2002.

GIL, A.C. **Métodos e técnicas de pesquisa social**. 4. ed. São Paulo: Atlas, 1994. 207 p.

GOMES, L. B.; *et al.* As origens do pensamento sistêmico: das partes para o todo. **Pensando famílias**, v. 18, n. 2, p. 3-16, 2014.

HEYLIGHEN, F.; JOSLYN, C. Cybernetics and second-order cybernetics. **Encyclopedia of physical science & technology**, v. 4, p. 155-170, 2001.

JACOBY, J.; TROUTMAN, T.; KUSS, A.; MAZURSKY, D. Experience and expertise in complex decision making. **NA – Advances in Consumer Research**, v. 13, p. 469-472, 1986.

JAVEED, F.; *et al.* Discovering software developer's coding expertise through deep learning. **IET Software**, v. 14, n. 3, p. 213-220, 2020.

MORADI DAKHEL, A. C.; DESMARAIS, M.; KHOMH, F. Assessing Developer Expertise from the Statistical Distribution of Programming Syntax Patterns. *In*: **Evaluation and Assessment in Software Engineering**, p. 90-99, 2021.

PRIKLADNICKI, R.; AUDY, J. L. N. Interdisciplinaridade na engenharia de software. **Scientia**, v. 19, n. 2, p. 117-127, 2008.

RICK, E.; KNIGHT, K. **Inteligência Artificial**. 2. ed. New York: McCraw-Hill, 1991.

RIEHLE, D. **Framework design**: A role modeling approach. 2000. Tese (Doutorado) - ETH Zurich, Zurique, Suíça, 2000.

ROBILLARD, M.; WALKER, R.; ZIMMERMANN, T. Recommendation systems for software engineering. **IEEE software**, v. 27, n. 4, p. 80-86, 2009.

SARMA, A.; *et al.* Hiring in the global stage: Profiles of online contributions. *In*: IEEE INTERNATIONAL CONFERENCE

ON GLOBAL SOFTWARE ENGINEERING (ICGSE), 11, 2016. **Proceedings...** Orange County, CA, USA: IEEE, 2016. p.1-10. Disponível em: https://ieeexplore.ieee.org/document/7577412. Acesso em: 11 nov 2022.

SARACEVIC, Tefko. Ciência da informação: origem, evolução e relações. **Perspectivas em Ciência da Informação**, Belo Horizonte, v. 1, n. 1, jan./jun. 1996. ISSN 19815344. Disponível em: http://portaldeperiodicos.eci.ufmg.br/index.php/pci/article/view/235/22. Acesso em: 25 maio 2022.

SOMMERMAN, A. **Inter ou transdisciplinaridade.** Da fragmentação disciplinar ao novo diálogo entre os saberes. São Paulo: Paulus, 2006.

SOMMERVILLE, I. **Software engineering**. 9. ed. Boston: Addison-Wesley, 2011.

YAN, J.; *et al.* Profiling developer expertise across software communities with heterogeneous information network analysis. *In*: PROCEEDINGS OF THE TENTH ASIA-PACIFIC SYMPOSIUM ON INTERNETWARE. Beijing China 16 September 2018.**Proceedings....** New York: Association for Computing Machinery, 2018. p.1-9. Disponível em : https://dl.acm.org/doi/10.1145/3275219.3275226. Acesso em:11 nov 2022.