Exploração de Estados Implícitos em Requisitos de Software como Estratégia Pedagógica no Ensino de Engenharia de Requisitos e Testes *

Christopher Ric Coelho Saar[1], **Daricélio Moreira Soares**[1], **Manoel Limeira de Lima Júnior Almeida**[1]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal do Acre (UFAC) – Rio Branco, AC – Brazil

christopher.saar@sou.ufac.br, {daricelio.soares, manoel.almeida}@ufac.br

No ensino de Engenharia de Software, estudantes frequentemente produzem requisitos e critérios de aceitação que, embora sintaticamente adequados, são estruturalmente incompletos [Ouhbi and Pombo 2020]. Uma manifestação recorrente dessa lacuna é a omissão de estados e pré-condições essenciais (como autenticação ou permissões de negócio) que, ao permanecerem implícitos, indicam falhas no modelo mental sobre o comportamento dinâmico do sistema [Ferrari et al. 2022]. Esse fenômeno é explicado pela alta carga cognitiva envolvida na modelagem inicial de sistemas complexos [Sweller 1988], o que compromete atividades críticas subsequentes, como a derivação rigorosa de casos de teste [Garousi et al. 2020]. Com a crescente adoção de Inteligência Artificial (IA) generativa, observa-se que ferramentas automáticas tendem a reproduzir ou amplificar esses pressupostos implícitos presentes nos requisitos originais, oferecendo soluções que aparentam corretude mas carecem de profundidade lógica [Siddiq et al. 2024, White et al. 2023].

Esta pesquisa propõe uma estratégia pedagógica que utiliza a identificação sistemática de estados implícitos como instrumento de desenvolvimento conceitual [Bhatia and Kumar 2023]. A abordagem é apoiada por uma ferramenta que integra a **Semântica de Frames** [Fillmore 1976] para o mapeamento de papéis lógicos (Agente e Objeto) a uma **Base de Estados** estruturada em ontologias de domínio pré-definidas. Concebida como um apoio metacognitivo, a ferramenta atua como um "espelho": em vez de corrigir automaticamente o texto, ela sinaliza inconsistências e slots vazios na estrutura do requisito, provocando o estudante a refletir sobre a completude de suas especificações. Esse processo busca reduzir a carga cognitiva ao direcionar o foco do aluno para as transições de estado críticas que ele não processou inicialmente.

A avaliação do impacto pedagógico ocorre por meio de um desenho quase-experimental [Campbell and Stanley 1963], utilizando a **Rubrica de Completude Estrutural (RCE)** detalhada na Tabela 1. Mede-se a evolução do aprendizado comparando a frequência de estados implícitos e a qualidade estrutural de artefatos produzidos manualmente e com o suporte da ferramenta.

Espera-se que esta intervenção resulte na consolidação de modelos mentais mais robustos, permitindo que os alunos identifiquem falhas lógicas precocemente. A principal contribuição reside na articulação entre a análise técnica via *frames* e a teoria da carga

*__Link para o vídeo de apresentação:__ https://www.youtube.com/watch?v=oRf1uwJktcI

Tabela 1. Rubrica de Completude Estrutural (RCE) para Avaliação de Requisitos

Elemento	Critério de Avaliação	Impacto no Modelo Mental
Ator (Agente)	Identificação do papel autorizado na ação.	Definição de Responsabilidade.
Ação (Verbo)	Uso de verbo de operação claro (CRUD/Regra).	Precisão Procedimental.
Objeto	Especificação da entidade do domínio afetada.	Rastreabilidade de Dados.
Estado	**Explicitação de pré-condições de transição.**	**Redução da Carga Cognitiva.**

cognitiva, oferecendo um método para o uso crítico de IA no ensino de Engenharia de Software. Ao explicitar o que antes era tácito, a pesquisa favorece uma compreensão rigorosa da relação entre requisitos e testes, essencial para a formação de profissionais capazes de validar comportamentos complexos em cenários industriais modernos, superando a tendência de negligenciar a verificação em prol da mera elicitação [Daun et al. 2023].

Referências

Bhatia, S. and Kumar, R. (2023). Identifying implicit assumptions in software requirements: An nlp-based approach. *Journal of Systems and Software*, 195:111516.

Campbell, D. T. and Stanley, J. C. (1963). *Experimental and quasi-experimental designs for research*. Rand McNally, Chicago.

Daun, M., Grubb, A. M., Stenkova, V., and Tenbergen, B. (2023). A systematic literature review of requirements engineering education. *Requirements Engineering*, 28(1):145–175.

Ferrari, A., Lipitakis, E., and Gnesi, S. (2022). Detecting ambiguity and incompleteness in natural language requirements: A survey and taxonomy. *IEEE Transactions on Software Engineering*, 48(8):2800–2821.

Fillmore, C. J. (1976). Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Garousi, V., Giray, G., Tuzun, E., Catal, C., and Felderer, M. (2020). Software engineering education: Challenges and future directions. *IEEE Transactions on Education*, 63(4):245–254.

Ouhbi, S. and Pombo, N. (2020). Software engineering education: Challenges and perspectives. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 202–209.

Siddiq, M. S., Santos, J. C., Cuadrado, J. S., and Hussain, S. (2024). Empirical study on the effectiveness of large language models in software testing: A systematic literature review. *Information and Software Technology*, page 107386.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance code quality and requirements. In *Proceedings of the 45th International Conference on Software Engineering (ICSE)*, pages 1–11. IEEE.