Uma Abordagem de Teste de Integração com Planejamento em Inteligência Artificial

Luis Felipe de Lima[1]

[1]Programa de Pós-Graduação em Informática (PPGInf)
Departamento de Informática (DInf)
Universidade Federal do Paraná (UFPR)
Curitiba - PR - Brasil

lflima@inf.ufpr.br

Abstract. *This paper presents ongoing doctoral research that proposes an approach based on artificial intelligence (AI) planning techniques to support integration testing. AI planning is used in integration testing representations that result in testing plans indicating an order for software systems integration and testing. The approach is structured in an architecture to generate the testing plans and execute the tests from the plan information. The research method includes bibliographic reviews, feasibility studies, case studies, and experiments to evaluate the representation and approach. It is expected to reduce the effort of planning and executing the integration testing.*

Resumo. *Este artigo apresenta uma pesquisa de doutorado em andamento que propõe uma abordagem baseada em técnicas de planejamento em inteligência artificial (IA) para apoiar o teste de integração. O planejamento em IA é utilizado em representações do teste de integração que resultam em planos de teste que indicam uma ordem para integrar e testar sistemas de software. A abordagem é estruturada em uma arquitetura para gerar os planos de teste e executar o teste a partir das informações dos planos. O método de pesquisa inclui revisões bibliográficas, estudos de viabilidade, estudos de caso e experimentos de avaliação da representação e da abordagem. Espera-se contribuir com a redução do esforço do planejamento e da execução do teste de integração.*

1. Introdução

Diante do aumento da complexidade e escala dos sistemas de software, os testes tornaram-se essenciais para garantir produtos mais confiáveis, com maior qualidade e menores custos [Garousi et al. 2020]. Um teste consiste na execução de um software[1] em um ambiente controlado com a intenção de revelar defeitos [Myers 1979]. A literatura de engenharia de software (ES) [Wazlawick 2013, Sommerville 2019, Pressman and Maxim 2021] destaca que esses benefícios são alcançados com a realização dos testes em fases distintas ao longo do desenvolvimento do software.

As fases de teste se iniciam com testes individuais de cada unidade, ou módulo de programação, do software. Após o teste unitário, as unidades são integradas conforme a especificação. Nessa fase, chamada teste de integração, os testes são conduzidos à

[1]Neste trabalho, um sistema de software em teste é chamado de sistema sob teste (SST).

medida que as unidades são gradualmente integradas para revelar defeitos ocasionados pela interação entre as unidades. Com o software totalmente integrado, ocorrem testes de verificação das funcionalidades e validação com usuários [Delamaro et al. 2016].

A integração das unidades é feita geralmente conforme uma ordenação predefinida e pode ser direcionada por estratégias como a *top-down* e a *bottom-up* [Sommerville 2019]. As estratégias de integração consideram a estrutura hierárquica de controle do software que é dividida em níveis e estabelecida pelas chamadas de operações (ou dependências) entre as unidades. A estratégia *top-down* inicia a integração com as unidades de nível de hierarquia mais alto e a *bottom-up* com unidades de nível mais baixo.

Essas características do teste de integração podem impactar a complexidade do planejamento e, consequentemente, contribuir para o aumento do esforço requerido na execução do teste. Assim, caracteriza-se a necessidade do apoio de técnicas para otimizar recursos e reduzir o esforço durante o planejamento desses testes.

As técnicas de inteligência artificial (IA) têm sido amplamente estudadas e aplicadas na ES [Mashkoor et al. 2022]. Nesse contexto, as técnicas de planejamento em IA são usadas em várias áreas da ES, como teste de segurança [Lima 2020] e alocação de requisitos [Pereira et al. 2022]. Informalmente, o planejamento em IA consiste na geração de um conjunto de ações, ou plano, para a resolução de um problema a partir de uma representação do conhecimento sobre esse problema [Russell and Norvig 2016]. Como o planejamento em IA pode ser custoso no que diz respeito a tempo e recursos, busca-se a geração de planos factíveis e não necessariamente planos ótimos [Ghallab et al. 2004].

As representações com planejamento em IA são formalizadas em linguagens formais e os planos são gerados por ferramentas chamadas planejadores. As técnicas de planejamento em IA não-clássico potencializam a expressividade da representação ao possibilitar o uso de valorações numéricas [Ghallab et al. 2004]. Esses aspectos contribuem para representar problemas configuráveis, flexíveis e com critérios.

A *Planning Domain Definition Language* (PDDL) [McDermott et al. 1998] é a linguagem-padrão em competições de planejamento em IA não-clássico. As representações em PDDL são especificadas por um modelo composto por um domínio e um problema. O domínio contém o conhecimento do problema a ser resolvido e o problema compreende uma instância com definições válidas para um problema específico. O domínio e o problema são descritos em arquivos distintos que são usados como entrada do planejador para a geração dos planos. O Metric-FF[2] [Hoffmann 2003] é um planejador geralmente utilizado para a resolução de problemas de PDDL.

Diante dessas características, o planejamento em IA permite orquestrar atividades e serviços a partir da parametrização de comandos e da definição de condições. Um teste de software, que contém uma situação inicial e objetivos bem definidos, pode ser diretamente representado como um problema de planejamento em IA. Assim, o uso do planejamento em IA no teste de software também é motivado pela possibilidade de associação entre problemas com ênfase em objetivo. [Bozic and Wotawa 2019, Lima et al. 2020]. Dessa forma, o problema de pesquisa que direciona este trabalho é como o planejamento em IA pode contribuir com o planejamento e a execução do teste de integração.

[2]Detalhes sobre o funcionamento do Metric-FF são encontrados em `https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html`.

Este artigo apresenta uma pesquisa de doutorado em andamento que propõe uma abordagem que busca contribuir com a redução do esforço do teste de integração usando técnicas de planejamento em IA não-clássico. Os usuários pretendidos são gerentes de teste e testadores. Espera-se que a abordagem possa ser utilizada por leigos em IA, devido a modelagem ser pré-configurada e instanciada dinamicamente. O uso da abordagem é esperado para cenários de desenvolvimento prescritivo ou ágil e com possibilidade de direcionamento para ambientes de integração contínua. Acredita-se que a abordagem não seja limitada a algum tipo de linguagem ou *framework* de desenvolvimento específicos.

O restante do artigo está estruturado nas seguintes seções: a Seção 2 apresenta os objetivos e as questões de pesquisa. A Seção 3 descreve trabalhos relacionados. A Seção 4 relata o método de pesquisa. A Seção 5 destaca os resultados preliminares. Por fim, a Seção 6 conclui o artigo e aponta as próximas etapas da pesquisa.

2. Objetivos e questões de pesquisa

Com base na contextualização apresentada, o objetivo geral desta pesquisa é:

- Propor uma abordagem baseada em técnicas de planejamento em IA não-clássico que gera planos de teste de integração que indicam uma ordem para integrar e testar sistemas de software de acordo com as estratégias *top-down* e *bottom-up*.

Para alcançar o objetivo geral, são propostos os seguintes objetivos específicos:

a) Definir uma arquitetura que associa o planejamento em IA ao teste de integração;
b) Representar características do teste de integração em linguagem de planejamento;
c) Avaliar a representação e a abordagem em estudos experimentais.

Diante desses objetivos, espera-se responder as seguintes questões de pesquisa:

- QP1. Quais são os elementos necessários para gerar planos de teste de integração com planejamento em IA?
- QP2. Quais são os benefícios da aplicação do planejamento em IA no contexto de teste de integração?

3. Trabalhos relacionados

Esta seção apresenta alguns trabalhos relacionados ao tema da presente pesquisa.

Estudos recentes de revisão bibliográfica destacam a aplicação do planejamento em IA no teste de software [Bozic and Wotawa 2019, Lima et al. 2020]. Os trabalhos mencionados nessas revisões incluem propostas para as fases de teste de unidade [Galler et al. 2010] e teste de sistema [Simos et al. 2019, Bozic et al. 2019]. Esta pesquisa se difere ao aplicar o planejamento em IA na fase de teste de integração.

Outras técnicas de IA, como algoritmos de otimização e genéticos, são usadas para apoiar o planejamento do teste de integração [Zhang et al. 2019] [Zhang et al. 2022]. Esses técnicas utilizam características dos grafos que representam os sistemas para realizar a ordenação de unidades. A presente pesquisa se baseia em planejamento em IA para gerar planos que indicam uma ordem para a integração. Nesse sentido, o principal diferencial do planejamento em IA é a possibilidade de representar mais características do ambiente de teste e adaptar a ordenação de acordo com diferentes estratégias de integração.

O trabalho [Silva and Lemos 2011] propõe uma representação com planejamento em IA que gera planos indicando uma ordem de integração e teste de componentes de sistemas auto-adaptativos. Esta pesquisa representa alguns aspectos adicionais associados ao teste, como prioridades, níveis hierárquicos e estratégias de integração.

4. Método de pesquisa

O *Design Science* [Engström et al. 2020] é um paradigma que auxilia a elaborar e apresentar soluções de pesquisas aplicadas e contribui para sumarizar, justificar, avaliar e comunicar suas contribuições, além de indicar o que se espera de sua relevância prática, rigor e inovação. O método de pesquisa é descrito nesta seção a partir dos elementos do *Design Science* apresentados na Figura 1.

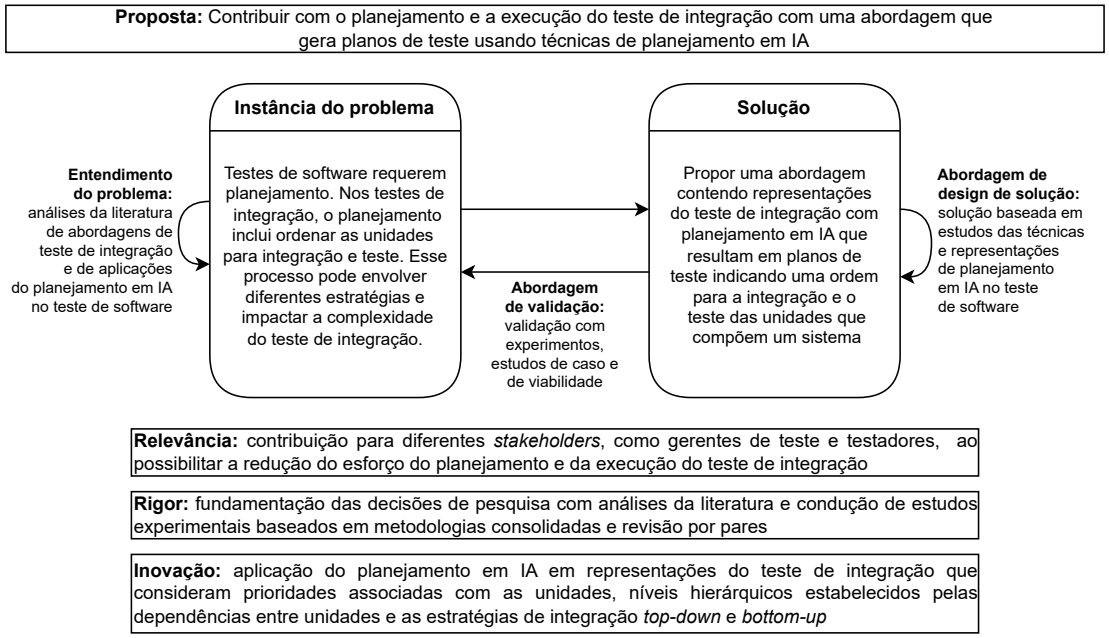

Figura 1. Abordagem proposta estruturada com o paradigma *Design Science*.

A **proposta** é referente ao objetivo geral da pesquisa apresentado na Seção 2. A **instância do problema** são aspectos do planejamento do teste de integração que motivaram o desenvolvimento da solução. O **entendimento do problema** abrange as análises da literatura conduzidas para aprofundar o embasamento teórico do escopo da pesquisa. A **abordagem de design de solução** inclui estudos que contribuíram para a definição da abordagem. Diante desses elementos, foi proposta como **solução** a abordagem usando técnicas de planejamento em IA para gerar planos de teste que auxiliam o planejamento e a execução do teste de integração. A **abordagem de validação** destaca os estudos esperados para validar a abordagem proposta.

A partir desse delineamento, a pesquisa é baseada nas seguintes atividades:

(i) Realizar revisões bibliográficas sobre o teste de integração e o planejamento em IA a fim de identificar lacunas para posicionar a pesquisa na literatura e obter contribuições para fundamentar e complementar a abordagem proposta;

(ii) Definir módulos de planejamento e execução do teste para compor uma arquitetura que que associa o teste de integração ao planejamento em IA com base em procedimentos de teoria de teste de software consolidados na literatura;

(iii) Representar o teste de integração como um problema de planejamento em IA;
(iv) Modelar o problema de planejamento em IA em PDDL;
(v) Conduzir um estudo de viabilidade com simulações de SSTs para verificar evidências da efetividade do planejamento em IA na geração dos planos a partir da representação proposta;
(vi) Realizar revisões na representação e na modelagem com base nos resultados da estudo de viabilidade;
(vii) Conduzir um estudo de caso usando especificações reais de ambientes de teste integração para instanciar a modelagem revisada e gerar planos de teste;
(viii) Implementar os componentes de software e definir as ferramentas de apoio para a arquitetura da abordagem;
(ix) Definir métricas para a medição do esforço da abordagem;
(x) Conduzir experimentos para avaliar a representação e a abordagem. Estudos experimentais podem avaliar as métricas definidas e realizar comparações com trabalhos correlatos. Estudos com usuários podem avaliar aspectos das representações como legibilidade, benefícios observados no teste como transparência e rastreabilidade, além da aplicação da abordagem na prática de desenvolvimento.

5. Resultados preliminares

Esta pesquisa foi iniciada em abril de 2020 com a realização das revisões bibliográficas (atividade i) que contribuíram para as definições da arquitetura do teste de integração (atividade ii), dos elementos contidos na representação do teste (atividade iii) e da linguagem de planejamento e do planejador considerados na atividade (iv).

Na atividade (ii) foi definida uma arquitetura para a abordagem com módulos de planejamento e execução do teste, como ilustra a Figura 2. O módulo de planejamento do teste tem como principal *stakeholder* o gerente de teste. A entrada desse módulo é um arquivo contendo informações obtidas em especificações sobre o SST. As informações desse arquivo são usadas na instanciação dinâmica dos arquivos PDDL que são encaminhadas ao planejador Metric-FF para gerar os planos de teste. O tratamento dos arquivos e a automatização do planejador são realizados por um *framework*.

O módulo de execução do teste contém mecanismos para a execução do teste baseados na teoria de teste de software de [Delamaro et al. 2016]. Para isso, o *framework* mapeia o plano de teste para um *script* que, ao ser executado pelo testador, realiza a integração das unidades e os testes no SST conforme a ordenação indicada. A abordagem prevê uma ferramenta de apoio que gera dados de teste adequados para os testes com base na especificação do SST. A saída desse módulo é um relatório de resultados do teste. A atividade (ii) é associada ao objetivo específico *a*.

Na atividade (iii) foram definidas formalizações considerando um SST representado por um conjunto U de unidades, um conjunto N de níveis hierárquicos e um conjunto P_r de prioridades. Essas formalizações incluem a descrição de como cada unidade $u_i \in U$ está organizada nos níveis $n_j \in N$ e sua respectiva associação com uma prioridade $p_{r_i} \in Pr$ de precedência de integração. Ainda, foi estabelecido como essas definições são usadas na representação das estratégias *top-down* e *bottom-up*.

Na atividade (iv), o problema de planejamento correspondente foi modelado em PDDL seguindo as definições da atividade (iii). Essas atividades contribuem com o obje-

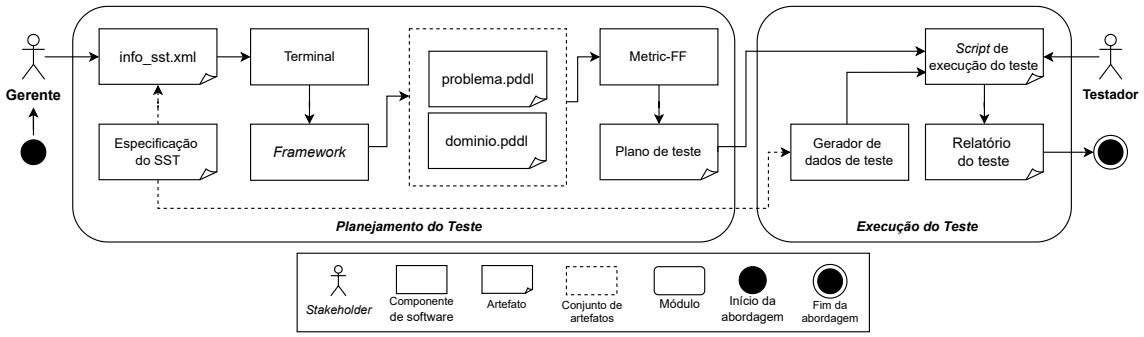

Figura 2. Visão geral da arquitetura da abordagem.

tivo específico *b*. O domínio PDDL foi ajustado para as estratégias *top-down* e *bottom-up* e contém a descrição de quatro ações. A ação **INTEGRA_UNIDADE** representa a integração das unidades, a ação **TESTA_SUBSISTEMA** indica o teste do subsistema gerado após cada integração, a ação **INCREMENTA_PRIORIDADE** realiza o incremento das prioridades e a ação **TROCA_NIVEL** realiza a troca dos níveis hierárquicos.

O problema PDDL representa um cenário de teste estabelecido pelas características do SST e da estratégia de integração. O estado inicial equivale ao momento anterior ao início do teste de integração. O estado final é alcançado após a integração de todas as unidades de U e de pelo menos uma execução do teste no SST a cada integração.

Os planos são compostos por ações que indicam a ordem de integração das unidades e a execução do teste do SST. Após cada execução do teste é indicado o incremento da prioridade. Quando todas as unidades de um determinado nível foram integradas é indicada a troca para o próximo nível. Os planos estabelecem o início da integração pelas unidades do primeiro nível para a estratégia *top-down* e pelas unidades do último nível para a estratégia *bottom-up*.

Exemplos da modelagem PDDL ajustada para as estratégias *top-down* e *bottom-up* estão disponíveis em um repositório (*link*). Esses exemplos foram elaboradas durante um estudo de viabilidade (atividade v) baseado na metodologia de [Wohlin et al. 2000]. Os exemplos consideram uma simulação de SST com cinco unidades (*A, B, C, D, E*), estruturadas em três níveis (*N1, N2, N3*) e associadas a duas prioridades (*P1, P2*). O Código 1 apresenta excertos dos planos de teste gerados pelo Metric-FF para cada exemplo.

Código 1. Planos de teste gerados pelo Metric-FF.

```
1  Estratégia top-down                Estratégia bottom-up
2
3  INTEGRA_UNIDADE A N1 P1            INTEGRA_UNIDADE E N3 P1
4  TESTA_SUBSISTEMA                   TESTA_SUBSISTEMA
5  INCREMENTA_PRIORIDADE              INCREMENTA_PRIORIDADE
6  INCREMENTA_PRIORIDADE              INTEGRA_UNIDADE D N3 P2
7  TROCA_NIVEL                        TESTA_SUBSISTEMA
8  INTEGRA_UNIDADE B N2 P1            INCREMENTA_PRIORIDADE
9  TESTA_SUBSISTEMA                   TROCA_NIVEL
10 INCREMENTA_PRIORIDADE              INTEGRA_UNIDADE B N2 P1
11 INTEGRA_UNIDADE C N2 P2            TESTA_SUBSISTEMA
12 TESTA_SUBSISTEMA                   INCREMENTA_PRIORIDADE
13 INCREMENTA_PRIORIDADE              INTEGRA_UNIDADE C N2 P2
```

```
14 TROCA_NIVEL                          TESTA_SUBSISTEMA
15 INTEGRA_UNIDADE E N3 P1              INCREMENTA_PRIORIDADE
16 TESTA_SUBSISTEMA                     TROCA_NIVEL
17 INCREMENTA_PRIORIDADE                INTEGRA_UNIDADE A N1 P1
18 INTEGRA_UNIDADE D N3 P2              TESTA_SUBSISTEMA
19 TESTA_SUBSISTEMA
```

O estudo de viabilidade mostrou que foi possível gerar planos de teste factíveis a partir da modelagem proposta, indicando que a representação do teste de integração com planejamento em IA foi efetiva. O estudo também destacou possíveis atualizações na representação que serão consideradas nas atividades futuras da pesquisa. A atividade (v) contribuiu para o objetivo específico *c*.

6. Considerações finais

Este artigo apresentou uma pesquisa de doutorado em andamento que objetiva contribuir com o teste de integração. A principal contribuição da pesquisa é gerar planos de teste a partir de representações com planejamento em IA. Os planos de teste são artefatos que podem ser usados por diferentes *stakeholders* para auxiliar o gerenciamento e a redução do esforço do teste de integração.

Atualmente estão sendo conduzidas as atividades (vi) e (vii) como parte do objetivo específico *b*. As próximas etapas da pesquisa são as atividades (viii), (ix) e (x) buscando atingir o objetivo específico *c*. Paralelamente, está prevista a produção de artigos com os resultados alcançados para publicação em eventos e revistas das áreas de teste de software, ES e IA para destacar o caráter multidisciplinar da pesquisa.

Agradecimentos

Este artigo foi realizado com o apoio do PPGInf-CAPES/MEC e C3SL-DInf/UFPR.

Referências

Bozic, J., Tazl, O. A., and Wotawa, F. (2019). Chatbot testing using AI planning. In *International Conference On Artificial Intelligence Testing*, pages 37–44.

Bozic, J. and Wotawa, F. (2019). Software testing: According to plan! In *International Conference on Software Testing, Verification and Validation Workshops*, pages 23–31.

Delamaro, M., Jino, M., and Maldonado, J. (2016). *Introdução ao teste de software. 2. ed.* Elsevier.

Engström, E., Storey, M.-A., Runeson, P., Höst, M., and Baldassarre, M. T. (2020). How software engineering research aligns with design science: a review. *Empirical Software Engineering*, 25(4):2630–2660.

Galler, S. J., Zehentner, C., and Wotawa, F. (2010). AIana: An AI planning system for test data generation. In *Workshop on Testing Object-Oriented Systems*, pages 1–8.

Garousi, V., Rainer, A., Lauvås Jr, P., and Arcuri, A. (2020). Software-testing education: A systematic literature mapping. *Journal of Systems and Software*, 165:110570.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

Hoffmann, J. (2003). The Metric-FF planning system: translating "ignoring delete lists" to numeric state variables. *Journal of artificial intelligence research*, 20:291–341.

Lima, L. F. (2020). Teste de intrusão para aplicações web: um método com planejamento em inteligência artificial. Dissertação (Mestrado em Informática) - Programa de Pós-graduação em Informática, Universidade Federal do Paraná.

Lima, L. F., Silva, F., Grégio, A. R. A., and Peres, L. M. (2020). A systematic literature mapping of artificial intelligence planning in software testing. In *International Conference on Software Technologies*, pages 152–159.

Mashkoor, A., Menzies, T., Egyed, A., and Ramler, R. (2022). Artificial intelligence and software engineering: Are we ready? *Computer*, 55(3):24–28.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL - The Planning Domain Definition Language. *Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational*.

Myers, G. J. (1979). The art of software testing. *ISBN: 0-471-04328-1*.

Pereira, F. C., Neto, G. B., Lima, L. F. d., Silva, F., and Peres, L. M. (2022). A tool for software requirement allocation using artificial intelligence planning. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*, pages 257–258.

Pressman, R. S. and Maxim, B. R. (2021). *Engenharia de software - 9ª edição*. AMGH. ISBN: 978-6558040101.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: A modern approach*. Malaysia; Pearson Education Limited.

Silva, C. E. d. and Lemos, R. d. (2011). Dynamic plans for integration testing of self-adaptive software systems. In *International Symposium on Software Engineering for Adaptive and Self-managing Systems*, pages 148–157.

Simos, D. E., Bozic, J., Garn, B., Leithner, M., Duan, F., Kleine, K., Lei, Y., and Wotawa, F. (2019). Testing TLS using planning-based combinatorial methods and execution framework. *Software Quality Journal*, 27(2):703–729.

Sommerville, I. (2019). *Engenharia de software. 10ª edição*. Pearson Universidades, ISBN: 978-8543024974.

Wazlawick, R. (2013). *Engenharia de software: Conceitos e práticas*, volume 1. Elsevier Brasil.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., and Regnell, B. (2000). Experimentation in Software Engineering - An Introduction. *Doedrecht the Netherlands*.

Zhang, Y., Jiang, S., Ding, Y., Yuan, G., Liu, J., Lu, D., and Qian, J. (2022). Generating optimal class integration test orders using genetic algorithms. *International Journal of Software Engineering and Knowledge Engineering*, 32(06):871–892.

Zhang, Y., Jiang, S., Wang, X., Chen, R., and Zhang, M. (2019). An optimization algorithm applied to the class integration and test order problem. *Soft Computing*, 23(12):4239–4253.