TAILORING ORIENTADO À REUTILIZAÇÃO EM PROCESSOS DE SOFTWARE:

UMA PROPOSTA DE INTEGRAÇÃO DE PADRÕES E BOAS PRÁTICAS EM TODAS AS FASES DO CICLO DE VIDA[1]

Resumo

O Tailoring de Processos de Software adapta metodologias e artefatos às necessidades de cada projeto, mas raramente integra mecanismos formais de reutilização. Este artigo propõe um modelo de Tailoring Orientado à Reutilização, com padrões de requisitos, projeto, arquitetura e teste em todas as fases do desenvolvimento, a partir de lacunas identificadas por revisão de literatura, como padronização, rastreabilidade e adaptação contextual. O modelo é situado frente a normas como SWEBOK, ISO/IEC/IEEE 12207, CMMI e MPS.BR, e discute seus limites de escala no Scrum, indicando Scrum of Scrums, LeSS e SAFe como complementos entre equipes. Por fim, ilustra-se sua aplicação em um cenário de *help desk*, com métricas de acompanhamento e uma estimativa incremental de implantação.

Palavras-chave: Reuso. Rastreabilidade. Padronização. Aprendizado contínuo. Modelo.

1. Introdução

O desenvolvimento de software envolve múltiplas atividades e decisões ao longo do ciclo de vida do sistema, e cada projeto possui características que exigem adaptação das práticas de engenharia de software. Esse processo, conhecido como *Software Process Tailoring*, ajusta metodologias e artefatos às necessidades do projeto. Na maioria das organizações, contudo, o tailoring é aplicado apenas para simplificar etapas, deixando de lado o reuso sistemático de conhecimento.

A reutilização de software reduz esforço, custo e risco (Pressman e Maxim, 2020), abrangendo requisitos, modelos e decisões de projeto — ainda que de forma fragmentada (Irshad et al., 2018; Wedyan e Abufakher, 2020). Surge, portanto, a necessidade de integrar reutilização e processos de software de forma explícita: o

[1] Versão completa do artigo disponível em: https://github.com/IgorBandeira/Artigo-completo-Tailoring-Orientado-para-Reutilizacao-em-Processos-de-Software

Tailoring Orientado à Reutilização propõe configurar o desenvolvimento, desde o início, para promover reuso sistemático em todas as fases do ciclo de vida, transformando-o em um ciclo de aprendizagem contínua que retroalimenta novos projetos.

2. Materiais e Métodos

Este estudo seguiu duas etapas: (1) uma revisão de literatura exploratória (RLE), para identificar práticas e lacunas de reutilização no ciclo de vida do software; e (2) a modelagem de um processo de Tailoring Orientado à Reutilização, fundamentada nessas evidências.

a) Protocolo da revisão de literatura exploratória

O processo seguiu duas questões norteadoras principais: (QP1) Quais práticas de reutilização são reportadas nas diferentes fases do ciclo de vida de software? (QP2) Quais lacunas e limitações são identificadas na adoção sistemática de reuso nesses contextos?

As buscas foram realizadas no Google Scholar, cobrindo o período de 1995 a 2024, com strings combinando termos como "*software reuse*", "*requirements reuse*", "*design patterns*", "*test reuse*" e "*software evolution*" com "*systematic review*" ou "*empirical study*", conforme os critérios de inclusão e exclusão sintetizados na Tabela 1.

Reconhece-se como limitação o uso exclusivo do Google Scholar, o que pode ter omitido estudos relevantes não indexados — razão pela qual esta revisão é caracterizada como exploratória, e não sistemática no sentido estrito: o objetivo foi reunir evidências suficientes para fundamentar a modelagem proposta, não esgotar a literatura disponível sobre o tema.

A busca resultou em aproximadamente 50 registros, dos quais 28 foram considerados elegíveis após triagem por título e resumo; da leitura completa, 17 foram excluídos por especificidade excessiva ou duplicação, restando 11 trabalhos na síntese. A análise foi qualitativa, organizada por fase do ciclo de vida, fundamentando o referencial teórico e a modelagem do processo.

Tabela 1. Síntese do protocolo da RLE

Dimensão	Descrição
Bases de dados	Google Scholar
String de busca	Combinações de "*software reuse*", "*design patterns*", "*requirements reuse*", "*test reuse*" e "*software evolution*" com "*systematic review*" ou "*empirical study*", variando por fase
Período	1995–2024
Critérios de inclusão	Artigos e livros em inglês, revisados por pares ou de editoras acadêmicas reconhecidas, sobre reuso em alguma fase do ciclo de vida
Critérios de exclusão	Estudos sem relação com processos de software; duplicados; ou restritos a domínios muito específicos sem generalização
Seleção de estudos	Triagem por título e resumo seguida de leitura completa dos estudos elegíveis
Limitação	Cobertura restrita ao Google Scholar; revisão caracterizada como exploratória

Fonte: Elaborado pelo autor (2026).

A Tabela 2 sintetiza os 11 estudos incluídos, classificados por tipo de estudo, fase do ciclo de vida e principal contribuição extraída para a modelagem do processo proposto.

Tabela 2. Síntese analítica dos estudos incluídos na revisão de literatura exploratória

Estudo	**Tipo de estudo**	**Fase do ciclo de vida**	**Principal achado/contribuição**
IRSHAD; PETERSEN; POULDING (2018)	Revisão sistemática (SLR)	Requisitos	Reuso de requisitos exige unidades de especificação reaproveitáveis; falta de padronização é barreira central
POOLEY; WARREN (2008)	Proposta de mecanismo	Requisitos	Propõe o RTRT; ausência de rastreabilidade entre versões compromete a confiabilidade do reuso
ZOWGHI;	Survey	Requisitos	Elicitação é atividade

Estudo	Tipo de estudo	Fase do ciclo de vida	Principal achado/contribuição
COULIN (2005)			social e iterativa; reuso envolve transferência de conhecimento tácito
GAMMA et al. (1995)	Catálogo técnico	Projeto e Implementação	Documenta os padrões GoF como vocabulário comum para design orientado a objetos
BUSCHMANN et al. (1996)	Catálogo técnico	Projeto e Implementação / Validação	Sistematiza padrões arquiteturais (camadas, pipelines, brokers) reutilizáveis para a organização global do sistema
WEDYAN; ABUFAKHER (2020)	Revisão sistemática (SLR)	Projeto e Implementação	Uso ad hoc de padrões, dependente da experiência individual, compromete a uniformidade arquitetural
IZURIETA; REIMANIS (2019)	Estudo empírico	Projeto e Implementação / Evolução	Padrões implementados sofrem degradação comportamental progressiva sem governança formal
SINGHAL et al. (2021)	Revisão terciária	Validação	Maioria dos estudos foca priorização e seleção, não catalogação, gerando duplicação de scripts
EL-HALAWANY; ELMINIR; EL-BAKRY (2024)	Estudo empírico	Validação	Testes reaproveitados com vínculo a requisitos geram ganhos de eficiência e rastreabilidade
ARCURI et al. (2020)	Proposta de framework	Validação	Apresenta o framework FrUITeR: reuso de testes de interface pode ser medido sistematicamente
PRESSMAN; MAXIM (2020)	Livro-texto	Transversal (todas as fases)	Formalização de processo e reutilização são eixos complementares; evolução inclui atividades corretivas, adaptativas e perfectivas

Fonte: Elaborado pelo autor (2026).

A Tabela 2 evidencia um corpus heterogêneo e majoritariamente não empírico — apenas 3 dos 11 trabalhos são revisões sistemáticas ou terciárias —, confirmando o caráter exploratório desta revisão, sem aplicação de instrumentos padronizados de avaliação crítica como o CASP ou o MMAT. Trabalhos futuros que busquem revisão sistemática devem ampliar a busca a bases como Scopus e Web of Science, adotando o protocolo PRISMA completo.

b) Modelagem do processo

Com base nas evidências consolidadas pela RLE, procedeu-se à modelagem do Tailoring Orientado à Reutilização: para cada fase do ciclo de vida, a literatura apontou problemas recorrentes cujas soluções existentes mostraram-se insuficientes, e o modelo propõe mecanismos específicos para endereçá-los de forma estruturada.

Na fase de requisitos (Irshad et al., 2018; Pooley e Warren, 2008), motivou-se o catálogo vivo de requisitos; no projeto e implementação (Wedyan e Abufakher, 2020; Izurieta e Reimanis, 2019), o catálogo de design patterns; na validação (Singhal et al., 2021; El-Halawany et al., 2024), a biblioteca de testes; e na evolução (Pressman e Maxim, 2020), os Maintenance Patterns.

3. Referencial Teórico

A integração entre reutilização e processos de software é pouco explorada na literatura, que trata o reuso de modo pontual, sem mecanismos integradores, embora Pressman e Maxim (2020) considerem os dois eixos complementares. Este referencial aprofunda quatro dimensões — requisitos, projeto e implementação, validação e evolução — articulando-as com o Tailoring Orientado à Reutilização.

Importa situar essa lacuna frente às normas: o SWEBOK (IEEE Computer Society; ISO/IEC/IEEE 24765, 2017) trata o reuso como tópico transversal, sem tailoring próprio; a ABNT NBR ISO/IEC/IEEE 12207 (2021) não detalha sua adaptação em organizações ágeis; o CMMI Institute (2018) o trata como gestão de ativos de processo; e o MPS.BR (SOFTEX, 2023) inclui Gerência de Reutilização — sem conectar reuso e cerimônias ágeis no nível operacional da equipe.

3.1. Especificação de requisitos

A Engenharia de Requisitos é a base conceitual do desenvolvimento; nesta fase, a reutilização reduz a complexidade e aumenta a consistência entre projetos. Irshad et al. (2018) apontam que requisitos já validados podem ser adaptados a novos contextos, reduzindo retrabalho, e Zowghi e Coulin (2005) destacam que a elicitação é social e iterativa, envolvendo transferência de conhecimento tácito.

Apesar desses benefícios, Irshad et al. (2018) identificam obstáculos — falta de padronização documental e ausência de mecanismos formais de adaptação —, e Pooley e Warren (2008) mostram que a carência de rastreabilidade compromete a confiabilidade do reuso. A Engenharia de Requisitos foi priorizada por essa natureza encadeada: inconsistências propagam-se às etapas seguintes, e a rastreabilidade só se sustenta como parte de um processo formal.

3.2. Projeto e implementação

O projeto de software traduz decisões abstratas de requisitos em estruturas concretas, por meio dos padrões de Gamma et al. (1995) — soluções que reduzem o esforço cognitivo do projetista — e dos padrões arquiteturais de Buschmann et al. (1996), que organizam o sistema globalmente. Pressman e Maxim (2020) destacam que esses padrões permitem internalizar práticas consolidadas.

Wedyan e Abufakher (2020), porém, observam que padrões são frequentemente usados de maneira ad hoc, comprometendo a uniformidade arquitetural, e Izurieta e Reimanis (2019) demonstram que padrões implementados se degradam ao longo do tempo sem governança formal. Por isso, o modelo institucionaliza a identificação, seleção e validação de padrões como etapa explícita do processo.

3.3. Validação

A validação assegura que o software atende aos requisitos estabelecidos, e a reutilização permite reaproveitar testes já validados — associado por El-Halawany et al. (2024) a ganhos de eficiência e rastreabilidade. A automação reforça essa importância: testes automatizados protegem contra regressões, e padrões de teste oferecem estruturas reutilizáveis para problemas recorrentes (Buschmann et al., 1996), como demonstra o framework FrUITeR de Arcuri et al. (2020).

Singhal et al. (2021), todavia, identificam desafios: a maioria dos estudos foca priorização e seleção, não catalogação, levando à duplicação de scripts. Essa

lacuna motiva a prioridade dada à catalogação e modularização de testes no modelo, que define processos e artefatos capazes de sustentar a rastreabilidade entre testes e requisitos apontada por El-Halawany et al. (2024).

3.4. Evolução

A evolução é o momento em que o software responde a mudanças técnicas, organizacionais ou de negócio, exigindo compreensão do histórico de decisões (Pressman e Maxim, 2020). Izurieta e Reimanis (2019) mostram que padrões implementados degradam-se progressivamente sem governança formal, elevando os custos de manutenção. Por isso, a evolução foi incluída como fase explícita, institucionalizando análise de impacto e registro formal de decisões.

4. Proposta de Tailoring

O objetivo central desta proposta é transformar a reutilização em princípio estruturante do processo, de modo que o conhecimento produzido pelo time circule entre as fases de desenvolvimento. Essa visão fundamenta o Modelo Sistêmico de Tailoring com Foco em Reutilização, que organiza práticas de reuso na rotina de equipes ágeis.

4.1. Racionalidade e fundamentação da proposta

Embora o trabalho não tenha como objetivo validar empiricamente o processo proposto, buscou-se reduzir ambiguidades quanto à origem de seus elementos. A inclusão de fases, artefatos, papéis e diretrizes foi orientada por dois eixos complementares.

O primeiro eixo é a fundamentação na literatura: a reutilização se concretiza com padronização, registro de decisões e disseminação do conhecimento — papel dos catálogos, ADRs e padrões propostos. O segundo são as lacunas práticas: equipes ágeis perdem conhecimento entre sprints, problema que os elementos do modelo endereçam via mecanismos integrados às cerimônias do Scrum.

4.2. Princípio fundamental

O modelo parte de uma premissa central: equipes ágeis aprendem e esquecem rapidamente, e o conhecimento fragmentado raramente retorna ao processo. Três princípios sustentam a proposta: conhecimento tratado como ativo reutilizável; reuso incorporado às cerimônias cotidianas; e padrões funcionando como guias para decisões antes arbitrárias.

4.3. Visão geral do modelo

O Modelo Sistêmico organiza-se em quatro fases, correspondentes aos momentos do ciclo de vida: requisitos, projeto e implementação, validação e evolução. Cada fase incorpora mecanismos de reutilização definidos a partir das lacunas da literatura, integrando-se às cerimônias do Scrum sem impor sobrecarga às equipes.

As seções seguintes detalham cada fase, descrevendo componentes, artefatos gerados e justificativas de inclusão.

Tabela 3. Visão geral do Modelo Sistêmico de Tailoring com Foco em Reutilização

Fase	Principais artefatos	Integração Scrum
1 – Requisitos	Catálogo de requisitos, User Story Patterns, Critérios de aceite, Mini-ADRs	Refinamento e Sprint Planning
2 – Projeto e Implementação	Catálogo de design patterns, ADRs completos, Biblioteca de componentes, Templates	Spike de reuso, Code Review, Daily Scrum
3 – Validação	Biblioteca de testes, Test Patterns, Pipelines CI/CD padronizados	Code Review, Retrospectiva
4 – Evolução	Maintenance Patterns, Change Records, Catálogo de refatorações	Retrospectiva, Sprint Planning

Fonte: Elaborado pelo autor (2026).

4.4. Estrutura detalhada das fases

4.4.1. Fase 1 – Análise de Requisitos Orientada à Reutilização

Esta fase enfrenta as lacunas da seção 3.1 — padronização documental e rastreabilidade (Irshad et al., 2018; Pooley e Warren, 2008) — por meio do Catálogo Vivo de Requisitos Reutilizáveis, repositório evolutivo de requisitos funcionais padrão. Inserir no Refinamento verificações de reuso transforma essa cerimônia em detector de redundâncias, complementado pelos User Story Patterns, que evitam a perda de rastreabilidade.

4.4.2. Fase 2 – Projeto e Implementação Orientada a Padrões

Esta fase responde às lacunas da seção 3.2 — uso ad hoc de padrões e degradação arquitetural (Wedyan e Abufakher, 2020; Izurieta e Reimanis, 2019) — por meio do Catálogo Oficial de Design Patterns e do Spike de Reuso, que verifica formalmente componentes semelhantes antes de criar algo novo. Os ADRs preservam o raciocínio das escolhas técnicas, e o Code Review verifica aderência arquitetural.

4.4.3. Fase 3 – Validação Orientada à Reutilização

Esta fase endereça as lacunas da seção 3.3 — ausência de catalogação formal de testes (Singhal et al., 2021) e perda de rastreabilidade (El-Halawany et al., 2024) — por meio da Biblioteca de Testes Reutilizáveis e dos Test Patterns, que organizam estratégias para problemas recorrentes (Buschmann et al., 1996), institucionalizando o reuso em pipelines de CI/CD (Arcuri et al., 2020).

4.4.4. Fase 4 – Evolução e Manutenção Orientada à Reutilização

Esta fase responde às lacunas da seção 3.4 — degradação de padrões sem rastreabilidade de decisões (Izurieta e Reimanis, 2019) e dificuldade de manutenção sem histórico (Pressman e Maxim, 2020) — por meio dos Maintenance Patterns e dos Change Records, que registram motivação e alternativas descartadas. O Catálogo de Refatorações torna a reestruturação uma capacidade organizacional, garantindo evolução rastreável do produto.

4.5. Integração com o Scrum

Os papéis complementares propostos surgem da necessidade de sustentar o modelo no cotidiano: catálogos e padrões tendem a se degradar sem responsabilidades explícitas. Três funções são definidas sem implicar criação de novos cargos, podendo ser exercidas por pessoas já existentes na equipe.

O Pattern Steward define os padrões técnicos como referência de qualidade arquitetural; o Knowledge Curator mantém atualizado o conhecimento produzido — catálogos, ADRs, bibliotecas —; e o Reusability Coach identifica oportunidades de reuso na rotina das equipes. Nas cerimônias, o Refinamento é o ponto preferencial de intervenção; o Sprint Planning formaliza a seleção de padrões; e a Retrospectiva fecha o ciclo, atualizando catálogos para que o conhecimento retorne ao processo.

Figura 1. Diagrama do Modelo de Tailoring Orientado à Reutilização.

Fonte: Elaborado pelo autor (2026).

4.5.1. Limites do Scrum e Escalonamento Ágil para Reuso Entre Projetos

É preciso reconhecer uma limitação estrutural: o Scrum, em sua formulação padrão, é desenhado para a escala de um único time. Os catálogos e bibliotecas propostos neste modelo, contudo, geram ativos organizacionais, reaproveitáveis por equipes distintas — tensão que o Scrum isolado não resolve, por não ter papel ou cerimônia para sincronizar catálogos entre times ou arbitrar conflitos de adoção entre *squads*.

Frameworks de escalonamento ágil podem suprir essa lacuna: o Scrum of Scrums (Sutherland, 2001) sincroniza catálogos via fórum periódico de representantes. O LeSS (*Large-Scale Scrum*, Larman e Vodde, 2016) introduz um Backlog de Produto único compartilhado por múltiplas equipes, alinhando-se ao Catálogo Vivo de Requisitos único. Já o SAFe (*Scaled Agile Framework*, Knaster e Leffingwell, 2020) oferece a estrutura mais robusta para institucionalizar ADRs e o Catálogo de Design Patterns como ativos de Arquitetura Corporativa, ainda que com maior peso processual.

Recomenda-se, portanto, que a adoção do modelo em múltiplas equipes seja acompanhada de um desses mecanismos: Scrum of Scrums para poucas equipes, LeSS para porte médio, e SAFe para organizações maiores. Essa escolha não altera as quatro fases do modelo, mas define se os papéis de Pattern Steward, Knowledge Curator e Reusability Coach operam em escopo de time único ou multiequipe.

4.6. Benefícios Esperados

Com base nas evidências da literatura e na estrutura do modelo proposto, esperam-se: redução do retrabalho; maior previsibilidade das entregas; aprendizado organizacional contínuo; aceleração na criação de funcionalidades; menor dependência de especialistas isolados; documentação viva; e governança mais auditável.

4.7. Aplicação Ilustrativa, Métricas de Acompanhamento e Estimativa de Implantação

Ilustra-se a aplicação do modelo em um sistema web de *help desk* para abertura, triagem e resolução de chamados. Na Fase 1, o Catálogo Vivo receberia User

Story Patterns para “abertura de chamado” e “atribuição de responsável”. Na Fase 2, o Spike de Reuso verificaria um componente de login reutilizável. Na Fase 3, a Biblioteca de Testes forneceria *fixtures* padronizadas, e na Fase 4 retrabalhos de performance seriam registrados como Maintenance Patterns.

A aplicação, ainda que não validada empiricamente, sugere métricas como a taxa de reaproveitamento de requisitos, a densidade de ADRs por entrega, e a taxa de acerto do Spike de Reuso (proporção de *spikes* com reaproveitamento efetivo) e o tempo médio de resolução de defeitos. Coletados do *board* da equipe, esses indicadores forneceriam a base quantitativa que pesquisas futuras poderiam consolidar.

Quanto ao tempo de implantação, estima-se um horizonte de quatro a seis meses para implantação plena, distribuído incrementalmente (Tabela 4). Essas durações são estimativas ilustrativas, e não medições empíricas, devendo ser calibradas conforme o contexto organizacional.

Tabela 4. Estimativa incremental de implantação do modelo proposto

Período	Foco da etapa	Ações principais	Métrica de acompanhamento inicial
Mês 1	Fase 1 (Requisitos)	Criar Catálogo Vivo de Requisitos e definir User Story Patterns iniciais	Taxa de reaproveitamento de requisitos
Meses 2-3	Fase 2 (Projeto e Implementação)	Estruturar Catálogo de Design Patterns e instituir o Spike de Reuso	Densidade de ADRs por entrega; taxa de acerto do Spike de Reuso
Mês 4	Fase 3 (Validação)	Consolidar Biblioteca de Testes e Test Patterns documentados	Tempo médio de resolução de defeitos recorrentes
Meses 5-6	Fase 4 (Evolução) e consolidação	Formalizar Change Records e papéis complementares (Steward, Curator, Coach)	Consolidação dos quatro indicadores em painel único de retrospectiva

Fonte: Elaborado pelo autor (2026).

5. Considerações Finais

Este trabalho investigou como integrar a reutilização sistematicamente ao desenvolvimento de software. As práticas de reutilização mostraram-se presentes em todas as fases do ciclo de vida, mas fragmentadas e dependentes do conhecimento tácito das equipes. Propôs-se o Modelo de Tailoring com Foco em Reutilização, em quatro fases integradas às cerimônias do Scrum, que institucionaliza catálogos, ADRs e padrões de manutenção para que o processo acumule conhecimento rastreável entre projetos.

Reconhece-se que o modelo não foi validado empiricamente — sua principal limitação —, possivelmente via estudo de caso longitudinal em organizações que já adotem Scrum. Constitui também limitação a amplitude do recorte: a cobertura horizontal das quatro fases restringe o aprofundamento individual de cada componente, apresentado em nível conceitual. Recomenda-se que versões futuras aprofundem cada fase por meio de apêndices técnicos dedicados.

Em síntese, tratar o reuso como princípio estruturante do processo, e não como prática emergente, é condição necessária para que o conhecimento técnico produzido por equipes ágeis se converta em ativo organizacional duradouro, capaz de aumentar consistência e sustentar a qualidade do software.

6. Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. ABNT NBR ISO/IEC/IEEE 12207: engenharia de sistemas e software - processos de ciclo de vida de software. Rio de Janeiro: ABNT, 2021.

ARCURI, Andrea et al. FruITeR: a framework for evaluating UI test reuse. arXiv e-Prints, [s. l.], 2020. Disponível em: https://arxiv.org/pdf/2008.03427. Acesso em: 01 dez. 2025.

BUSCHMANN, Frank et al. Pattern-oriented software architecture: a system of patterns. Chichester: Wiley, 1996.

CMMI INSTITUTE. CMMI for development, version 2.0. Pittsburgh: ISACA, 2018.

EL-HALAWANY, Ahmed M.; ELMINIR, Hamdy K.; EL-BAKRY, Hazem. Improving reuse during the development process for web systems. Scientific Reports, [s. l.], v. 14, n. 1, art. 24023, out. 2024. DOI: 10.1038/s41598-024-74643-7.

GAMMA, Erich et al. Design patterns: elements of reusable object-oriented software. Boston: Addison-Wesley, 1995.

IRSHAD, Mohsin; PETERSEN, Kai; POULDING, Simon. A systematic literature review of software requirements reuse approaches. Information and Software Technology, [s. l.], v. 93, p. 223–245, 2018. DOI: 10.1016/j.infsof.2017.09.009.

IZURIETA, Clemente; REIMANIS, Derek. Behavioral evolution of design patterns: understanding software reuse through the evolution of pattern behavior. In: INTERNATIONAL CONFERENCE ON SOFTWARE REUSE (ICSR), 2019, [S. l.]. Proceedings [...]. Cham: Springer, 2019. p. 77–93.

ISO/IEC/IEEE. ISO/IEC/IEEE 24765: systems and software engineering - vocabulary. Geneva: ISO, 2017.

KNASTER, Richard; LEFFINGWELL, Dean. SAFe 5.0 distilled: achieving business agility with the Scaled Agile Framework. Boston: Addison-Wesley, 2020.

LARMAN, Craig; VODDE, Bas. Large-scale Scrum: more with LeSS. Boston: Addison-Wesley, 2016.

POOLEY, Rob; WARREN, Chris. Reuse through requirements traceability (RTRT). In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING ADVANCES (ICSEA), 3., 2008, [S. l.]. Proceedings [...]. [S. l.: s. n.], 2008. p. 65–70.

PRESSMAN, Roger S.; MAXIM, Bruce R. Software engineering: a practitioner's approach. 9. ed. New York: McGraw-Hill Education, 2020.

SINGHAL, Sushma et al. A systematic literature review on test case selection and prioritization: a tertiary study. Applied Sciences, [s. l.], v. 11, n. 24, art. 12121, 2021. DOI: 10.3390/app112412121.

SOFTEX. MPS.BR: guia geral MPS de software 2023. São Paulo: Associação para Promoção da Excelência do Software Brasileiro, 2023.

SUTHERLAND, Jeff. Agile can scale: inventing and reinventing SCRUM in five companies. Cutter IT Journal, [s. l.], v. 14, n. 12, p. 5–11, dez. 2001.

WEDYAN, Fadi; ABUFAKHER, Somia. Impact of design patterns on software quality: a systematic literature review. IET Software, [s. l.], v. 14, n. 1, p. 1–17, fev. 2020. DOI: 10.1049/iet-sen.2018.5446.

ZOWGHI, Didar; COULIN, Chad. Requirements elicitation: a survey of techniques, approaches, and tools. In: AURUM, Aybüke; WOHLIN, Claes (ed.). Engineering and managing software requirements. Berlin: Springer, 2005. p. 19–46.