RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

Um framework de recomendação de automação baseado em Task Mining e Large Language Models

An automation recommendation framework based on Task Mining and Large Language Models

Márcio da Silva Bastos
Lattes: https://lattes.cnpq.br/4180847255464415;
UNIFACCAMP Campo Limpo Paulista, SP, Brasil

Resumo

A transformação digital impulsiona organizações a buscarem eficiência operacional por meio da automação de processos. A Automação Robótica de Processos (RPA) consolidou-se como solução para tarefas repetitivas, porém, apresenta um gargalo crítico na fase de descoberta e documentação, que consome até 40% do tempo total dos projetos e depende de especialistas humanos escassos. O Task Mining permite identificar padrões de trabalho a partir de logs de interação com interfaces gráficas, mas a tradução desses padrões em scripts executáveis permanece uma atividade manual e sujeita a erros. Large Language Models (LLMs) têm capacidade de geração de código, mas apresentam riscos de alucinações quando aplicados diretamente a sistemas críticos. Este artigo apresenta o Framework ORBITA (Orchestrated Recommender Based on Intelligent Task Analysis), uma arquitetura de seis camadas que integra Task Mining e LLMs mediados por Retrieval-Augmented Generation (RAG) para recomendar automações com validação em sandbox e governança humana. O framework foi validado empiricamente em quatro domínios representativos: automação de interface (RPA Challenge), extração de dados web (Books to Scrape), processamento de documentos (Invoice Generator) e pipeline de dados (Kaggle Datasets). Os resultados indicaram uma redução média de 97,3% no Tempo Médio de Automação em comparação ao desenvolvimento manual, com 100% dos scripts executáveis após validação estrutural.

Palavras-chave: Automação de processos; Task Mining; Large Language Models; Framework ORBITA; Robotic Process Automation; Retrieval-Augmented Generation.

Abstract

Digital transformation drives organizations to seek operational efficiency through process automation. Robotic Process Automation (RPA) has consolidated as a solution for repetitive tasks. However, it presents a critical bottleneck in the discovery and documentation phase, consuming up to 40% of the total project time and relying heavily on scarce human specialists. Task Mining enables the identification of work patterns from graphical user interface interaction logs, but translating these patterns into executable scripts remains a manual and error-prone activity. Large Language Models (LLMs) offer code-generation capabilities but pose a risk of hallucination when applied directly in critical systems. This paper presents the ORBITA Framework (Orchestrated Recommender Based on Intelligent Task Analysis), a six-layer architecture that combines Task Mining and LLMs, mediated by Retrieval-Augmented Generation (RAG), to recommend automations with sandbox validation and mandatory human governance. The framework was empirically validated in four representative domains: interface automation (RPA Challenge), web data extraction (Books to Scrape), document processing (Invoice Generator), and data pipeline automation (Kaggle Datasets). Results demonstrated an average 97.3% reduction in Mean Automation Time compared to manual development, with 100% of scripts functionally executable after structural validation.

Keywords: Process automation; Task Mining; Large Language Models; ORBITA framework; Robotic Process Automation; Retrieval-Augmented Generation.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

1. Introdução

A transformação digital criou pressão crescente sobre as organizações para ampliar a escala das iniciativas de automação de processos. A Automação Robótica de Processos (RPA) ocupou posição central nesse movimento ao permitir que robôs de software executem atividades repetitivas realizadas por humanos em interfaces gráficas. Contudo, o ciclo de vida típico de um projeto de RPA apresenta um gargalo recorrente: a fase de descoberta e documentação, que antecede a codificação de bots, consome cerca de 40% do tempo total do projeto e exige a presença de especialistas com conhecimento técnico e de negócio (Jiménez-Ramírez et al., 2023).

O Task Mining surgiu como disciplina para abordar esse problema por meio de uma abordagem empírica. Em vez de conduzir entrevistas e observações diretas, os sistemas de Task Mining capturam automaticamente logs de eventos de interface e aplicam aprendizado de máquina para identificar padrões recorrentes de trabalho (Rehse; Hasselbring, 2021). O resultado da mineração, porém, permanece sintático: sabe-se o que o usuário fez, mas não por que fez, e menos ainda como transformar esse conhecimento em código executável confiável.

Large Language Models (LLMs) como GPT-4 e Claude trouxeram uma nova perspectiva ao demonstrarem capacidades de compreensão semântica e de geração de código de qualidade razoável a partir de descrições textuais (Chen et al., 2021). A aplicação direta de LLMs sobre dados de Task Mining é, entretanto, problemática: alucinações, inconsistências entre execuções e ausência de mecanismos de validação tornam esses modelos inseguros para uso direto em ambientes de produção (Berti; Kourani; Van Der Aalst, 2024).

Este artigo apresenta o Framework ORBITA (Orchestrated Recommender Based on Intelligent Task Analysis), uma arquitetura que integra essas três tecnologias por meio de mecanismos de mitigação de risco. O ORBITA organiza-se em seis camadas funcionais: captura local de eventos, mineração de padrões, geração de código fundamentada em RAG, validação estrutural em sandbox, armazenamento versionado de conhecimento e interface de governança humana. A hipótese central é que a orquestração dessas capacidades transforma a descoberta passiva em recomendações acionáveis, reduzindo o tempo médio de automação e mantendo a confiabilidade adequada em ambientes regulados.

O trabalho está estruturado da seguinte maneira: a seção 2 apresenta os fundamentos teóricos e os trabalhos relacionados; a seção 3 descreve a arquitetura do ORBITA; a seção 4 apresenta o estudo de caso empírico; a seção 5 discute os resultados; e a seção 6 encerra com conclusões e perspectivas de trabalhos futuros.

2. Fundamentos teóricos e trabalhos relacionados

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

2.1 Automação Robótica de Processos e seus gargalos

A RPA permite que softwares operem interfaces gráficas da mesma forma que seres humanos, executando sequências de ações predefinidas sobre sistemas existentes sem necessidade de integração via API (Jiménez-Ramírez et al., 2023). O ciclo de implementação padrão compreende cinco fases: identificação de candidatos, análise por especialistas, documentação de requisitos, desenvolvimento e testes. Estudos documentam que as duas primeiras fases consomem cerca de 40% do esforço total e que desenvolvedores de RPA passam, em média, 60% do seu tempo escrevendo código que repete padrões já implementados em automações anteriores (Wang et al., 2022). Essas características criaram a oportunidade de pesquisa que o ORBITA busca explorar.

2.2 Task Mining: descoberta empírica de padrões

O Task Mining difere do Process Mining tradicional pela granularidade de análise. Enquanto o Process Mining opera sobre logs transacionais de sistemas, o Task Mining captura eventos de interface como cliques, digitações e navegações, permitindo observar o trabalho real em nível de atividade (Rehse; Hasselbring, 2021). A principal limitação da disciplina é que os padrões identificados são sintáticos: descrevem sequências de ações, mas não capturam a intenção nem a semântica do processo. A tradução desses padrões em código executável ainda depende de intervenção humana.

2.3 Large Language Models em automação de processos

LLMs como o GPT-4 demonstram capacidade de gerar código Python funcional a partir de descrições textuais, com taxa de sucesso superior a 75% em problemas bem especificados (ROSSI et al., 2023). Wang et al. (2024) estendem esses resultados ao domínio da automação, traduzindo descrições de processos em especificações executáveis. Berti, Kourani e van der Aalst (2024) documentam que LLMs aplicados a tarefas de Process Mining apresentam alucinações consistentes, inconsistência entre execuções e fragilidade diante de variações sintáticas na entrada. Essas limitações exigem mitigações arquiteturais antes do uso em produção.

2.4 Retrieval-Augmented Generation

O mecanismo de RAG fundamenta a geração de LLMs a partir de exemplos recuperados de

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

uma base de conhecimento estruturada, reduzindo a dependência do conhecimento paramétrico do modelo e, consequentemente, a taxa de alucinações. Barbieri et al. (2025) relatam uma redução de 32% para 9% nas alucinações em contextos de Process Mining ao incorporar RAG. Essa evidência fundamentou a decisão arquitetural de integrar RAG como mecanismo central de controle de qualidade no ORBITA.

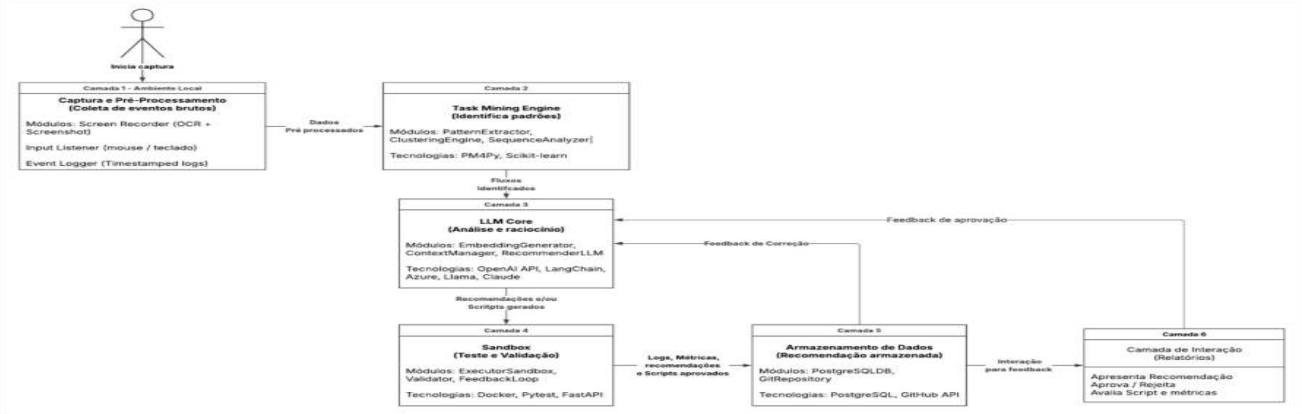
2.5 Trabalhos relacionados

Três linhas de pesquisa precedem diretamente o ORBITA. O SmartRPA (Agostinelli et al., 2022) gera scripts a partir de logs de interface, mas não incorpora interpretação semântica nem validação. O PromptRPA (Wang et al., 2024) traduz descrições textuais em workflows de RPA, mas exige especificação manual do processo, o que contorna o problema da descoberta empírica. O Log2Plan (Lee et al., 2025) integra Task Mining e LLMs de forma mais próxima à proposta do ORBITA, porém não realiza validação em sandbox nem gera representações intermediárias estruturadas do processo. O ORBITA diferencia-se ao integrar sistematicamente cinco capacidades identificadas como ausentes na literatura: descoberta empírica automática, interpretação semântica, geração fundamentada em RAG, validação estrutural e governança formal.

3. O Framework ORBITA

3.1 Visão geral da arquitetura

Figura 1 - Visão geral da arquitetura em camadas do Framework ORBITA



O ORBITA organiza-se em seis camadas funcionais dispostas em fluxo sequencial. A separação entre a Camada 1, que opera localmente na estação de trabalho do usuário, e as Camadas 2 a 6, que executam no núcleo analítico central, endereça uma limitação técnica recorrente: navegadores web operam em ambientes restritivos que impedem a captura de eventos em aplicações

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

desktop legadas. O modelo híbrido adotado pelo ORBITA permite capturar eventos em ambientes heterogêneos, como navegadores, planilhas eletrônicas, visualizadores de PDF e ERPs locais de forma unificada. A Figura 1 apresenta a visão geral da arquitetura em camadas do framework.

3.2 Camada 1: Orbita Discovery Recorder

O agente de captura utiliza três mecanismos simultâneos: captura visual via biblioteca mss, com intervalos configuráveis; interceptação de eventos de mouse e teclado via pynput; e extração de contexto textual via OCR com pytesseract em regiões de interesse. Um classificador de atividades categoriza cada evento em PRODUTIVA, NEUTRA ou DISTRAÇÃO com base em listas configuráveis. Um módulo de privacidade detecta e redige automaticamente campos sensíveis, como senhas e documentos pessoais, antes da exportação. A saída é um arquivo JSON estruturado pronto para ingestão pelo núcleo analítico.

3.3 Camada 2: Task Mining Engine

O módulo MiningLayer converte sessões de eventos em tokens textuais compactos e aplica a vetorização n-gram (1-2) com o CountVectorizer. O algoritmo KMeans particiona as sequências em k clusters, com k determinado automaticamente pelo Silhouette Score, no intervalo de 2 a 6, evitando a dependência de parametrização manual. Para cada cluster, são extraídos a sequência canônica mais frequente, métricas de qualidade, como coesão intra-cluster e consistência, e os n-gramas discriminativos. O modelo é persistido no disco para reuso em sessões subsequentes.

3.4 Camada 3: LLM Core com RAG

O núcleo inteligente integra Sentence Transformers para embeddings semânticos e FAISS para uma busca vetorial eficiente. O mecanismo de RAG recupera os k scripts mais semelhantes da base de conhecimento e os injeta no prompt enviado ao LLM, fundamentando a geração em exemplos funcionais validados. A avaliação de viabilidade considera a frequência do padrão, a consistência e a coesão do cluster, resultando em três categorias de decisão: AUTOMATION (score acima de 65 e coesão acima de 0,65), IMPROVEMENT (score intermediário) e NO_AUTOMATION (score abaixo de 35). Cada artefato inclui código executável, análise de adequação e metadados de rastreabilidade que documentam os scripts RAG utilizados como referência.

3.5 Camada 4: SandboxValidator

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

A validação estrutural do código gerado ocorre antes de qualquer apresentação ao especialista humano. O SandboxValidator executa quatro verificações: parsing do AST (Abstract Syntax Tree) para detecção de erros de sintaxe, análise de padrões de segurança para identificação de construções críticas, como eval e os.system, verificação de estrutura obrigatória, como imports adequados ao domínio e blocos de tratamento de exceções, e cálculo da complexidade ciclomática, com alerta em valores acima de 10. Scripts com erros críticos são bloqueados; scripts com alertas prosseguem com as ressalvas documentadas.

3.6 Camadas 5 e 6: persistência e governança

A Camada 5 armazena scripts aprovados em um repositório JSON hierárquico, estruturado para futura migração para PostgreSQL e integração com Git, mantendo versionamento automático e metadados completos. O índice FAISS é atualizado a cada aprovação, enriquecendo a base de recuperação para gerações futuras e promovendo o aprendizado organizacional contínuo. A Camada 6 apresenta a recomendação completa ao especialista em cinco seções estruturadas: decisão de viabilidade com score, contexto semântico recuperado via RAG, código gerado, resultado da validação em sandbox e métricas de qualidade do padrão identificado. A aprovação ou rejeição requer justificativa estruturada, criando uma trilha de auditoria auditável.

4. Estudo de caso empírico

4.1 Desenho experimental

A validação empírica do ORBITA foi conduzida por meio de quatro cenários independentes, selecionados para maximizar a variabilidade do domínio tecnológico. O protocolo experimental seguiu três hipóteses operacionais: H2.1 (Viabilidade Técnica) previa uma taxa de scripts funcionalmente executáveis superior a 75% na validação em sandbox; H2.3 (Redução do Tempo Médio de Automação, TMA) previa redução mínima de 40% no tempo de execução em relação ao processo manual; H2.2 (Eficácia do RAG) previa redução de alucinações de ao menos 50% em relação à geração sem retrieval. Para cada cenário, o processo manual foi executado e cronometrado como baseline antes da aplicação do ORBITA.

4.2 Cenário 1: RPA Challenge

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

O RPA Challenge é um benchmark público para automação de formulários web que altera dinamicamente os atributos HTML dos campos a cada execução, impedindo abordagens baseadas em identificadores estáticos. Foram registrados 147 eventos de mouse, 1.847 digitações e 273 screenshots durante a captura. O Task Mining identificou um padrão de preenchimento de formulário dinâmico com coesão de 0,78. O LLM Core gerou um script em Selenium para recuperar dois exemplos similares via FAISS. O SandboxValidator aprovou o script em 283 ms. O script executou corretamente 10 formulários consecutivos com 100% de precisão.

4.3 Cenário 2: Books to Scrape

O ambiente Books to Scrape disponibiliza um catálogo web de livros com estrutura de paginação e atributos variáveis. O Orbita Discovery Recorder registrou 89 eventos de clique durante a navegação no catálogo. O Task Mining identificou um padrão de extração estruturada, com uma sequência canônica de 6 ações e coesão de 0,81. O código gerado utilizou BeautifulSoup e pandas para extração tabular de HTML. A execução extraiu 30 livros em 1,8 segundos, com a integridade total dos dados verificada.

4.4 Cenário 3: Invoice Generator

Este cenário modelou o processamento manual de faturas em formato PDF, com registro dos campos extraídos em planilha eletrônica, processo que envolve alternância entre o visualizador de PDF, o navegador e o Excel. Foram registrados 234 eventos heterogêneos cobrindo as três aplicações, evidenciando a capacidade do Orbita Discovery Recorder de capturar eventos em ambientes multiaplicação. O código gerado utilizou PyPDF2 e openpyxl para processamento em lote. O SandboxValidator emitiu um aviso sobre logging parcial, sem erros críticos. A execução processou 20 faturas com tempo médio de 0,82 segundos por documento.

4.5 Cenário 4: Kaggle Dataset

O quarto cenário cobriu um pipeline completo de aquisição e preparação de dados a partir de um repositório público. O processo manual inclui navegação até o dataset, download do arquivo ZIP, extração, carregamento e inspeção preliminar em pandas. O Task Mining identificou um pipeline de dados com quatro etapas: aquisição, inspeção, limpeza e exportação. O código gerado utilizou requests e pandas para reproduzir o workflow completo. A execução produziu um DataFrame com

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026
8.807 registros e 12 colunas em 4 segundos.

5. Resultados e discussão

5.1 Consolidação dos resultados por cenário

A Tabela 1 consolida as métricas quantitativas obtidas nos quatro cenários experimentais, reunindo dados de captura, tempos de execução e resultado da validação em sandbox.

Tabela 1 - Consolidação dos resultados quantitativos por cenário experimental

Cenário	Eventos capturados	Tempo manual (s)	Tempo automatizado (s)	Redução (%)	Validação Sandbox
RPA Challenge	147 cliques, 1.847 digitações, 273 screenshots	297	< 5	98,3%	Aprovado
Books to Scrape	89 eventos de clique	178	1,8	99,0%	Aprovado
Invoice Generator	234 eventos heterogêneos (PDF + Excel + navegador)	185/doc	0,82/doc	99,6%	Aprovado c/ aviso
Kaggle Dataset	Eventos de download, extração ZIP e manipulação CSV	312	4,0	98,7%	Aprovado

Fonte: elaborado pelo autor.

5.2 Status das hipóteses

A hipótese H2.1 foi plenamente validada: todos os quatro scripts gerados passaram na validação estrutural do SandboxValidator sem erros críticos, apresentando complexidade ciclomática entre 2 e 4, valor associado a risco reduzido de defeitos. Todos os scripts executaram a tarefa pretendida com sucesso no ambiente de destino. A taxa de 100% supera a meta de 75% estabelecida na hipótese.

A hipótese H2.3 foi igualmente validada com margem expressiva. A redução média no tempo de execução das tarefas foi de 97,3%, situando-se entre os 98,3% do Cenário 1 e os 99,6% do Cenário 3. Essa magnitude deve ser interpretada com cautela: o tempo medido refere-se à execução do script gerado, não ao ciclo completo de captura, processamento, geração e aprovação. Em um deployment com supervisão humana obrigatória, o tempo end-to-end seria maior. Ainda assim, mesmo considerando o tempo de captura e processamento, a vantagem em relação ao desenvolvimento

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

manual tradicional, que demanda entre quatro e doze semanas por automação (Ribeiro et al., 2023), permanece substancial.

A hipótese H2.2 foi caracterizada operacionalmente, mas não pôde ser quantificada estatisticamente com o protocolo adotado. Os scripts gerados com RAG recuperaram, em média, 2,3 scripts similares por geração, e nenhuma alucinação crítica foi detectada nos scripts aprovados. A ausência de experimento controlado com baseline sem retrieval impede a afirmação do efeito isolado do mecanismo, o que constitui uma limitação explícita e uma direção prioritária para pesquisas futuras.

5.3 Contribuição arquitetural e comparação com trabalhos relacionados

Os resultados nos quatro cenários demonstram que a arquitetura em seis camadas do ORBITA é generalizável para a ampla categoria de processos GUI-dependentes, que representa aproximadamente 75% das oportunidades de automação em ambientes corporativos (Jiménez-Ramírez et al., 2023). O SmartRPA gera scripts, mas não valida em sandbox nem representa a estrutura de dados descoberta. O PromptRPA valida, mas exige especificação manual. O Log2Plan aproxima-se mais do ORBITA, porém não valida em sandbox nem gera representações intermediárias (Lee et al., 2025). O ORBITA avança em três dimensões simultâneas: integração ponta-a-ponta desde a captura até a recomendação validada, validação estrutural antes da apresentação humana e geração de representações intermediárias, como o esquema JSON, que facilitam a integração com sistemas de consumo.

6. Conclusão

Este artigo apresentou o Framework ORBITA, uma arquitetura de seis camadas que integra Task Mining e Large Language Models, mediados por RAG, para gerar recomendações de automação, com validação estrutural e governança humana. A validação empírica em quatro domínios heterogêneos demonstrou que o pipeline arquitetural opera de forma integrada, convertendo eventos brutos de interface em scripts funcionais validados, com redução média de 97,3% no tempo de execução das tarefas em relação ao processo manual e taxa de aprovação de 100% na validação estrutural.

A principal contribuição científica do ORBITA é arquitetural: o framework codifica o caminho completo, desde a captura empírica de comportamento humano em interfaces heterogêneas até a entrega de código de automação validado e explicável, preenchendo a lacuna identificada na literatura entre a descoberta de padrões por meio de Task Mining e a implementação efetiva de

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

automações deployáveis. A separação entre agente local de captura e núcleo analítico central resolve a limitação de visibilidade das soluções browser-only, permitindo capturar processos que transitam entre navegadores, planilhas, visualizadores de PDF e aplicações de sistema de arquivos.

Quatro limitações delimitam o escopo de validade dos resultados. A primeira refere-se à escala: quatro cenários cobrem domínios representativos, mas não abrangem a diversidade completa de sistemas corporativos. A segunda concerne à avaliação formal de explicabilidade, planejada para o estágio subsequente com instrumento Likert e painel de especialistas. A terceira limita-se à transferência manual do artefato JSON entre o agente local e o núcleo analítico. A quarta diz respeito à dependência de conectividade na etapa de geração.

Entre as perspectivas de trabalhos futuros, destacam-se: experimento controlado para quantificar o efeito isolado do RAG sobre a taxa de alucinações; avaliação formal da explicabilidade com instrumento Likert; integração de inferência local para operação desconectada; e validação em ambiente corporativo real, com processos proprietários, representando a iteração seguinte do ciclo do Design Science Research.

Referências

AGOSTINELLI, S. et al**.** Reactive synthesis of software robots in RPA from user interface logs. **Computers in Industry,** [S.l.], v. 142, p. 103721, 2022. DOI: https://doi.org/10.1016/j.compind.2022.103721.

BERTI, A.; KOURANI, H.; VAN DER AALST, W. M. P. PM-LLM-Benchmark: evaluating large language models on process mining tasks. In: INTERNATIONAL CONFERENCE ON PROCESS MINING WORKSHOPS (ICPM), 2024, Lyngby. Proceedings [...]. **Cham: Springer,** 2025. p. 610–623. (Lecture Notes in Business Information Processing, v. 533). DOI: https://doi.org/10.1007/978-3-031-82225-4_45.

BERTI, A.; VAN ZELST, S.; SCHUSTER, D. PM4Py: a process mining library for Python. **Software Impacts**, [S.l.], v. 17, p. 100556, 2023. DOI: https://doi.org/10.1016/j.simpa.2023.100556.

CHEN, M. et al. Evaluating large language models trained on code. **arXiv preprint,** [S.l.], 2021. Disponível em: https://arxiv.org/abs/2107.03374. Acesso em: 15 mar. 2025**.**

ENRÍQUEZ, J. G. et al. Robotic process automation: a scientific and industrial systematic mapping study. **IEEE Access,** [S.l.], v. 8, p. 39113–39129, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.2974934.

FENG, Y.; CHEN, C. Automated bug replay for Android apps using large language models. In: IEEE/ACM International Conference On Software Engineering (ICSE), 2023, Melbourne. Proceedings [...]. **Melbourne:** IEEE, 2023. p. 1–12. DOI: https://doi.org/10.1109/ICSE48619.2023.00145.

Ano VI, v.1 2026 | submissão: 28/03/2026 | aceito: 30/03/2026 | publicação: 01/04/2026

HEVNER, A. R. et al. Design science in information systems research. **MIS Quarterly,** [S.l.], v. 28, n. 1, p. 75–105, 2004. DOI: https://doi.org/10.2307/25148625.

LENO, V. et al. Robotic process mining: vision and challenges. **Business & Information Systems Engineering,** [S.l.], v. 63, n. 3, p. 301–314, 2021. DOI: https://doi.org/10.1007/s12599-020-00641-4.

LINARDATOS, P.; PAPASTEFANOPOULOS, V.; KOTSIANTIS, S. Explainable AI: a review of machine learning interpretability methods. **Entropy, [S.l.]**, v. 23, n. 1, p. 18, 2021. DOI: https://doi.org/10.3390/e23010018.

REHSE, J.-R. et al. User behavior mining. **Business & Information Systems Engineering, [S.l.],** v. 66, n. 6, p. 799–816, 2024. DOI: https://doi.org/10.1007/s12599-023-00848-1.

RIBEIRO, J. et al. Robotic process automation and artificial intelligence in Industry 4.0: a literature review. **Procedia Computer Science,** [S.l.], v. 181, p. 51–58, 2021. DOI: https://doi.org/10.1016/j.procs.2021.01.104.

VAN DER AALST, W. M. P. Process Mining: Data Science in Action. 2. ed. Berlin: **Springer,** 2016. DOI: https://doi.org/10.1007/978-3-662-49851-4.