CAPÍTULO 4

O USO DA UML 2.5.1 *UNIFIED MODELING LANGUAGE* (UML) NA MODELAGEM DE *SOFTWARE* COM AS METODOLOGIAS *SCRUM*, *EXTREME PROGRAMMING* (XP), *RATIONAL UNIFIED PROCESS* (RUP) E POCESSO UNIFICADO (PU)

https://doi.org/10.22533/at.ed.394122608014

Henderson Matsuura Sanches
Centro Universitário Processus – UniProcessus
ORCID: 0000-0003-2354-3393

RESUMO: A ***Unified Modeling Language*** (UML) emergiu como uma linguagem padrão para a modelagem de sistemas de software, consolidando diversas abordagens anteriores. Este artigo explora a relevância da UML 2.5.1, a versão atualizada pela ***Object Management Group*** (OMG), no contexto da análise e desenvolvimento de software, com um foco especial em sua integração e aplicação em metodologias de desenvolvimento ágeis e tradicionais, como ***Scrum, Extreme Programming*** (XP), ***Rational Unified Process*** (RUP) e Processo Unificado (PU). Serão abordados os principais conceitos, a categorização dos diagramas e a importância da UML na visualização, especificação, construção e documentação de artefatos de sistemas distribuídos. A metodologia empregada baseia-se em pesquisa bibliográfica em fontes acadêmicas e especializadas, incluindo a Biblioteca Digital Brasileira de Teses e Dissertações (BDTD), a Sociedade Brasileira de Computação (SBC) e a especificação oficial da OMG.

PALAVRAS-CHAVE: UML, Modelagem de Software, UML 2.5.1, Scrum, XP, RUP, Processo Unificado

The use of UML 2.5.1 Unified Modeling Language (UML) in software modeling with the Scrum, Extreme Programming (XP), Rational Unified Process (RUP), and Unified Process (UP) methodologies

ABSTRACT: The Unified Modeling Language (UML) emerged as a standard language for software systems modeling, consolidating various previous approaches. This article explores the relevance of UML 2.5.1, the current version updated by the Object Management Group (OMG), in the context of software analysis and development, with a special focus on its integration and application in agile and traditional development methodologies, such as Scrum, Extreme Programming (XP), Rational Unified Process (RUP), and Unified Process (UP). Key concepts, diagram categorization, and the importance of UML in visualizing, specifying, constructing, and documenting distributed system artifacts will be addressed. The methodology employed is based on bibliographic research in academic and specialized sources, including the Brazilian Digital Library of Theses and Dissertations (BDTD), the Brazilian Computer Society (SBC), and the official OMG specification.

KEYWORDS: UML, Software Modeling, UML 2.5.1, Scrum, XP, RUP, Unified Process

INTRODUÇÃO:

No cenário complexo do desenvolvimento de software, a capacidade de visualizar, especificar, construir e documentar sistemas de forma clara e padronizada é fundamental. A Unified Modeling Language (UML) surgiu como uma resposta a essa necessidade, unificando as notações de modelagem existentes e proporcionando uma linguagem gráfica universalmente aceita [1]. Desde sua concepção, a UML tem evoluído para acompanhar as demandas da indústria, culminando na versão 2.5.1, que é o foco deste trabalho [2].

Este artigo tem como objetivo apresentar a UML 2.5.1, destacando suas características, a categorização de seus diagramas e sua aplicação na modelagem de software, bem como sua integração com as principais metodologias de desenvolvimento de software, como o Processo Unificado (PU), Rational Unified Process (RUP), Scrum e Extreme Programming (XP). A pesquisa foi conduzida com base em fontes confiáveis, incluindo a especificação oficial da OMG, artigos científicos da Sociedade Brasileira de Computação (SBC), teses e dissertações da Biblioteca Digital Brasileira de Teses e Dissertações (BDTD), materiais de sites especializados e o livro "UML 2.5: Do Requisito à Solução"[1], [2], [3], [4] A estrutura do artigo seguirá as normas da Associação Brasileira de Normas Técnicas (ABNT), garantindo o rigor acadêmico e a padronização da apresentação do conteúdo.

O QUE É *UNIFIED MODELING LANGUAGE* (UML)

A UML é uma linguagem de modelagem visual que permite aos desenvolvedores e analistas representar a arquitetura, o design e o comportamento de sistemas de software. Ela não é uma metodologia de desenvolvimento, mas sim uma ferramenta que pode ser utilizada em diversas metodologias, como as orientadas a objetos. A principal vantagem da UML reside em sua capacidade de fornecer uma representação gráfica padronizada, facilitando a comunicação entre as equipes de desenvolvimento e as partes interessadas [1].

UML 2.5.1: A Versão Atual

(OMG), representa a mais recente iteração da linguagem. Esta versão é uma revisão menor da UML 2.5, que por sua vez, foi uma reescrita substancial da especificação anterior, visando simplificar e melhorar a clareza das definições semânticas. A UML 2.5.1 consolida a estrutura da linguagem em um único documento, eliminando redundâncias e aprimorando a consistência [2].

As Novidades e Melhorias da UML 2.5.1

A transição para a UML 2.5 e sua subsequente revisão na versão 2.5.1 trouxeram melhorias significativas com foco na simplificação, clareza e consistência da especificação. Embora a UML 2.5.1 seja primariamente uma versão de manutenção para corrigir erros e ambiguidades da 2.5, as mudanças consolidadas nesta fase são importantes. A Tabela 1 resume as principais novidades introduzidas a partir da UML 2.5.

Categoria da Mudança	Descrição da Melhoria
Simplificação da Especificação	A especificação, que antes era dividida em múltiplos documentos (Infraestrutura e Superestrutura), foi consolidada em um único documento, tornando-a mais coesa e fácil de consultar [2].
Clareza Semântica	Foram realizados esforços para tornar as definições dos elementos da UML mais precisas e menos ambíguas, reduzindo a margem para interpretações divergentes [2].
Redundância Removida	Elementos e conceitos duplicados ou sobrepostos que existiam entre a infraestrutura e a superestrutura foram eliminados ou fundidos, resultando em um metamodelo mais limpo [2].
Diagramas de Interação	Aprimoramentos na forma como os diagramas de interação (Sequência, Comunicação, etc.) são definidos, permitindo uma modelagem mais flexível e poderosa de cenários complexos [3].

Compatibilidade e Extensibilidade	Melhorias no mecanismo de perfis (Profiles), facilitando a adaptação da UML para domínios específicos (como sistemas de tempo real ou SOA) sem alterar o núcleo da linguagem [3].
Foco na Usabilidade	A reestruturação da especificação teve como um dos objetivos principais facilitar o aprendizado e o uso da UML por parte de desenvolvedores, analistas e ferramentas CASE (Computer-Aided Software Engineering) [2].

Tabela 1: Resumo das Principais Novidades da UML 2.5.1.

CATEGORIZAÇÃO DOS DIAGRAMAS UML 2.5.1

A UML 2.5.1 organiza seus diagramas em duas categorias principais: Diagramas de Estrutura e Diagramas de Comportamento. Essa categorização ajuda a compreender os diferentes aspectos de um sistema que podem ser modelados [3].

Diagramas de Estrutura

Os diagramas de estrutura representam os aspectos estáticos de um sistema, mostrando seus elementos e como eles se relacionam. Incluem:

- **Diagrama de Classe:** Essencial para a modelagem orientada a objetos, descreve as classes, seus atributos, métodos e os relacionamentos entre elas [3]. (Ver Figura 1)
- **Diagrama de Objeto:** Uma instância do diagrama de classe, que exibe objetos específicos e seus valores em um determinado momento. Na UML 2.5.1, é considerado uma variação do diagrama de classe [3].
- **Diagrama de Pacote:** Utilizado para organizar os elementos do modelo em grupos lógicos, facilitando a gestão de projetos grandes e complexos [3].
- **Diagrama de Estrutura Composta:** Detalha a estrutura interna de um classificador, mostrando suas partes, portas e conectores [3].
- **Diagrama de Componente:** Ilustra a organização e as dependências entre os componentes de um sistema, sendo fundamental para arquiteturas baseadas em componentes [3].
- **Diagrama de Implantação:** Representa a arquitetura física do sistema, mostrando os nós de hardware e como os artefatos de software são distribuídos e executados neles [3].

- **Diagrama de Perfil:** Permite estender a UML com estereótipos, valores marcados e restrições personalizadas para adaptar a linguagem a domínios ou plataformas específicas [3].

Diagramas de Comportamento

Os diagramas de comportamento modelam os aspectos dinâmicos de um sistema, descrevendo como os elementos interagem e respondem a eventos. Incluem:

- **Diagrama de Caso de Uso:** Descreve as funcionalidades do sistema do ponto de vista dos usuários (atores) e como eles interagem com o sistema para alcançar objetivos específicos [1], [3].tal para o sucesso dos Sprints do Scrum [10].
- **Diagrama de Atividade:** Representa o fluxo de trabalho ou o fluxo de controle de uma atividade, mostrando a sequência de ações e as decisões que as governam [1], [3].
- **Diagrama de Máquina de Estados:** Modelagem do comportamento de um objeto ao longo do tempo, mostrando seus estados e as transições entre eles em resposta a eventos [3].
- **Diagramas de Interação:** Uma subcategoria que foca na troca de mensagens entre objetos;
- **Diagrama de Sequência:** Enfatiza a ordem temporal das mensagens trocadas entre os objetos [1], [3].
- **Diagrama de Comunicação (ou Colaboração):** Foca na organização estrutural dos objetos que trocam mensagens [3].
- **Diagrama de Visão Geral da Interação:** Combina aspectos dos diagramas de atividade e sequência para fornecer uma visão de alto nível das interações [3].
- **Diagrama de Tempo:** Detalha as restrições de tempo e a duração dos eventos e estados dos objetos [3].

UML E O PROCESSO UNIFICADO (PU)

A UML, por ser uma linguagem de modelagem, atinge seu potencial máximo quando integrada a uma metodologia de desenvolvimento de software robusta. O Processo Unificado (PU) é amplamente reconhecido como a metodologia que melhor se alinha com a UML, tendo sido, inclusive, desenvolvido em conjunto com a própria linguagem por seus criadores (Booch, Rumbaugh e Jacobson). O PU é um processo iterativo e incremental, orientado a casos de uso, arquitetura e riscos, que se estrutura em quatro fases principais: Concepção, Elaboração, Construção e Transição [4]. Na Figura 1 é apresentado o Ciclo de Vida do Processo Unificado.

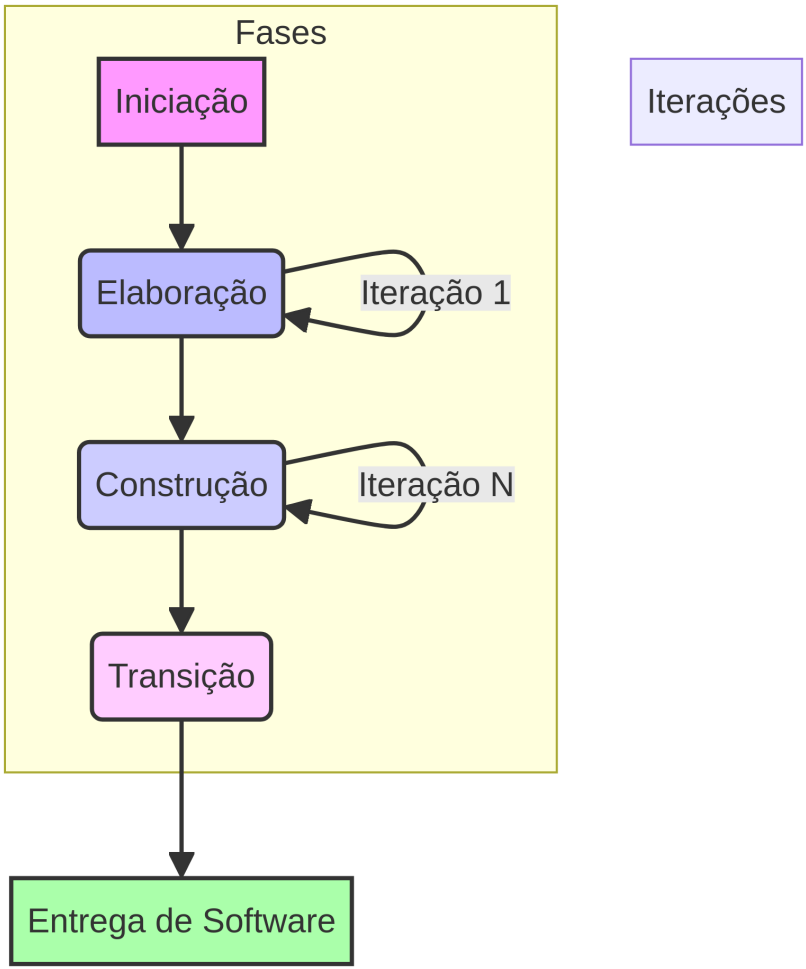

Figura 1: Ciclo de Vida do Processo Unificado.

A UML, por ser uma linguagem de modelagem, atinge seu potencial máximo quando integrada a uma metodologia de desenvolvimento de software robusta. O PU é amplamente reconhecido como a metodologia que melhor se alinha com a UML, tendo sido, inclusive, desenvolvido em conjunto com a própria linguagem por seus criadores (Booch, Rumbaugh e Jacobson). O PU é um processo iterativo e incremental, orientado a casos de uso, arquitetura e riscos, que se estrutura em quatro fases principais: Concepção, Elaboração, Construção e Transição [4].

Em cada fase do Processo Unificado, a UML desempenha um papel crucial:

- **Concepção:** Nesta fase inicial, os diagramas de Caso de Uso da UML são fundamentais para capturar os requisitos funcionais do sistema, definindo o escopo e as funcionalidades de alto nível. Diagramas de Atividade podem ser usados para modelar fluxos de trabalho de negócios [1], [4].
- **Elaboração:** A arquitetura do sistema é definida e refinada. Diagramas de Classe são utilizados para modelar a estrutura estática do sistema, enquanto Diagramas de Sequência e Comunicação detalham as interações entre os objetos. Diagramas de Componente e Implantação começam a ser esboçados para visualizar a estrutura física e de distribuição [3], [4].
- **Construção:** O foco é na implementação e testes. Os modelos UML são detalhados, e novos diagramas, como os de Máquina de Estados, podem ser empregados para modelar o comportamento complexo de objetos. A UML serve como um guia para a codificação e para a verificação da conformidade do código com o design [3], [4].
- **Transição:** O sistema é entregue aos usuários finais. Os modelos UML, especialmente os de Implantação, são atualizados para refletir a configuração final do sistema, servindo como documentação essencial para manutenção e futuras evoluções [3], [4].

A sinergia entre a UML e o Processo Unificado reside na capacidade da linguagem de fornecer as notações visuais necessárias para expressar os artefatos gerados em cada iteração do processo, desde a análise de requisitos até a implantação. Essa combinação promove uma abordagem disciplinada e iterativa para o desenvolvimento de software, resultando em sistemas mais bem compreendidos, projetados e documentados.

Rational Unified Process (RUP)

O **RUP** é uma adaptação comercial e detalhada do Processo Unificado, desenvolvido pela *Rational Software* (posteriormente adquirida pela IBM). O RUP é uma metodologia de engenharia de software configurável e iterativa que fornece uma abordagem disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Ele é fortemente orientado a casos de uso e à arquitetura, e sua estrutura é intrinsecamente ligada à UML [5]. Na Figura 2 é apresentado o ciclo de vida do *Rational Unified Process* (RUP),

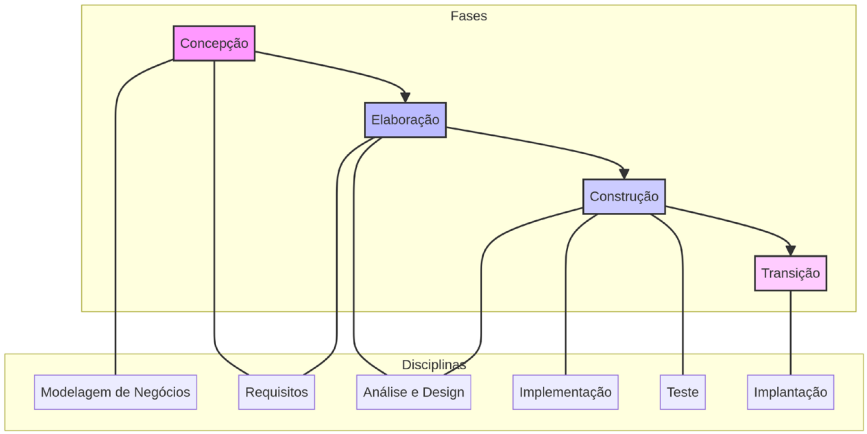

Figura 2: Ciclo de Vida do RUP.

O RUP define uma série de "disciplinas" (anteriormente chamadas de "fluxos de trabalho") que guiam o desenvolvimento, como Modelagem de Negócios, Requisitos, Análise e Design, Implementação, Teste e Implantação. Em todas essas disciplinas, a UML é a linguagem de modelagem primária. Por exemplo, na disciplina de Requisitos, os Diagramas de Caso de Uso são centrais; na Análise e Design, os Diagramas de Classe, Sequência e Componente são amplamente utilizados. O RUP, portanto, não apenas recomenda a UML, mas a integra profundamente em suas práticas e artefatos, tornando-se um exemplo paradigmático da aplicação da UML em um contexto de desenvolvimento de software industrial [5].

Embora o RUP tenha sido muito influente, sua natureza abrangente e, por vezes, burocrática, levou ao surgimento de adaptações mais leves, como o OpenUP, e à popularização de metodologias ágeis. No entanto, a base conceitual e a forte ligação com a UML permanecem como um legado importante para a engenharia de software.

COMPARATIVO ENTRE UML, PU, *SCRUM* E RUP

Para consolidar a compreensão das relações e diferenças entre a UML, o Processo Unificado (PU) e o *Rational Unified Process* (RUP), a Tabela 1 apresenta um comparativo de suas principais características. A **Tabela 2 temos um comparativo entr**e UML, Processo Unificado, RUP, Scrum e Extreme Programming (XP) [Fonte: Elaborado pelo autor com base em [1], [2], [4], [5], [6], [7]]

Característica	(UML)	PU	RUP	Scrum	XP
Natureza	Linguagem de modelagem gráfica	Metodologia de desenvolvimento de software (framework genérico)	Adaptação comercial e detalhada do UP (metodologia específica)	Framework ágil para gestão de projetos	Metodologia ágil com foco em práticas de engenharia de software
Objetivo Principal	Visualizar, especificar, construir e documentar sistemas	Fornecer uma abordagem iterativa e incremental para o desenvolvimento de software	Fornecer um processo configurável e disciplinado para o desenvolvimento de software, com suporte a ferramentas	Gerenciar projetos complexos de forma iterativa e incremental	Entregar software de alta qualidade rapidamente através de práticas específicas
Foco	Notação e semântica para modelos	Ciclo de vida do projeto, fases e iterações	Disciplinas, artefatos, papéis e ferramentas	Gestão de projetos, equipes auto-organizadas, entregas curtas	Práticas de engenharia (testes, refatoração, programação em pares)
Relação com a UML	É a linguagem utilizada por UP, RUP, Scrum e XP	Utiliza a UML como sua linguagem de modelagem principal	Integra a UML profundamente em todas as suas disciplinas e artefatos	Uso opcional e leve (UML as Sketch) para comunicação	Uso opcional e leve (UML as Sketch) para comunicação e design
Flexibilidade	Alta (pode ser usada com diversas metodologias)	Média (framework adaptável)	Média a Baixa (processo mais prescritivo, mas configurável)	Alta (framework leve e adaptável)	Alta (foco em adaptação e mudança)
Origem	Desenvolvida por Grady Booch, James Rumbaugh e Ivar Jacobson (Os Três Amigos)	Desenvolvido por Grady Booch, James Rumbaugh e Ivar Jacobson	Desenvolvido pela Rational Software (posteriormente IBM)	Ken Schwaber e Jeff Sutherland	Kent Beck
Principais Fases ou Disciplinas	Não possui fases, é uma notação	Concepção, Elaboração, Construção, Transição	Modelagem de Negócios, Requisitos, Análise e Design, Implementação, Teste, Implantação	Product Backlog, Sprint Planning, Sprint, Daily Scrum, Sprint Review, Sprint Retrospective	Planejamento, Design Simples, Testes (TDD), Programação em Pares, Refatoração, Integração Contínua

Tabela 2: Comparativo entre UML, PU, RUP, Scrum e XP.

METODOLOGIAS ÁGEIS E A UML: *SCRUM* E XP

Com o advento das metodologias ágeis, como Scrum e XP, a forma de utilizar a UML evoluiu de uma abordagem mais formal e prescritiva (como no RUP) para uma aplicação mais leve e adaptável, frequentemente referida como "UML as Sketch" (UML como Esboço) ou "UML as Blueprint" (UML como Projeto) [4]. Nessas abordagens, a UML é empregada para facilitar a comunicação e o entendimento, sem a necessidade de uma documentação exaustiva.

Scrum

O Scrum é um framework ágil para gerenciar projetos complexos, focado na entrega iterativa e incremental de produtos. Embora o Scrum não prescreva o uso de ferramentas ou técnicas específicas, a UML pode ser utilizada de forma complementar para visualizar e comunicar aspectos do sistema [4]. Na Figura 3 é apresentado o fluxo do Scrum.

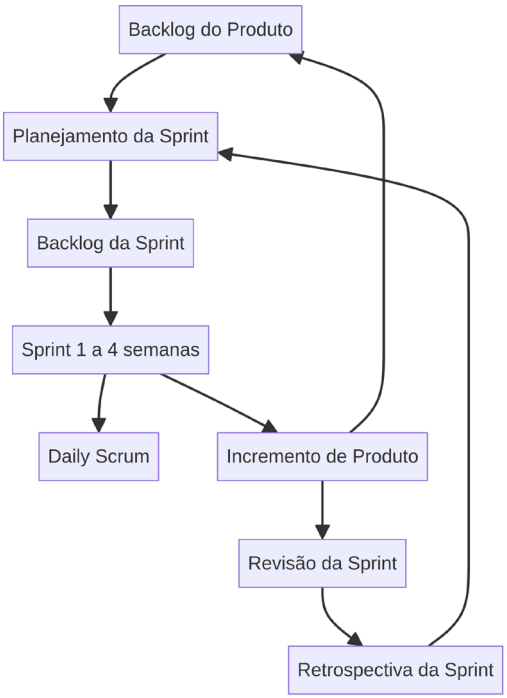

Figura 3: Fluxo do *Scrum*.

O Scrum é um framework ágil para gerenciar projetos complexos, focado na entrega iterativa e incremental de produtos. Embora o Scrum não prescreva o uso de ferramentas ou técnicas específicas, a UML pode ser utilizada de forma complementar para visualizar e comunicar aspectos do sistema [4].

- **Planejamento da *Sprint*:** Diagramas de Caso de Uso podem ajudar a refinar os itens do Product Backlog, enquanto Diagramas de Atividade podem modelar fluxos de trabalho de funcionalidades específicas.
- **Reuniões Diárias (*Daily Scrum*):** Embora não seja comum, diagramas simples podem ser usados para ilustrar rapidamente um problema ou uma solução.
- **Revisão da *Sprint (Sprint Review)*:** Diagramas de Componente ou de Implantação podem ser úteis para apresentar a arquitetura do incremento de software entregue.

O uso da UML no Scrum é pragmático: os diagramas são criados apenas quando agregam valor significativo à compreensão e comunicação da equipe, e são frequentemente descartados ou atualizados de forma mínima após cumprirem seu propósito [4]. Na **Tabela 3 temos a Relação entre Artefatos/Eventos do *Scrum* e o Uso da UML [Fonte: Elaborado pelo autor com base em [4], [6]].**

Artefato/Evento do Scrum	Uso da UML
Product Backlog	Diagramas de Caso de Uso para detalhar itens; Diagramas de Atividade para fluxos de negócio.
Sprint Planning	Diagramas de Sequência ou de Classe para discutir o design de funcionalidades.
Daily Scrum	Esboços rápidos de diagramas (UML as Sketch) para resolver problemas ou comunicar impedimentos.
Sprint Review	Diagramas de Componente ou de Implantação para apresentar o incremento de software.
Sprint Retrospective	Não há uso direto da UML, foco na melhoria do processo.
Incremento de Produto	Diagramas de Componente ou de Implantação para documentar a arquitetura da entrega.

Tabela 3: Relação entre Artefatos/Eventos do *Scrum* e o Uso da UML.

Extreme Programming (XP) e a UML

O XP é uma metodologia ágil que enfatiza a entrega contínua, a comunicação constante e a simplicidade. No XP, a documentação é mantida ao mínimo necessário, e a comunicação face a face é preferida. A UML, nesse contexto, é utilizada principalmente como uma ferramenta de comunicação rápida e informal [4].

Design Simples: Diagramas de Classe ou de Sequência podem ser desenhados rapidamente em um quadro branco para discutir uma solução de design com a equipe, sendo apagados ou fotografados após a discussão.

Cartões CRC (*Class-Responsibility-Collaboration*): Embora não sejam diagramas UML, os cartões CRC podem ser complementados com esboços de diagramas de classe para visualizar as interações.

Refatoração: Diagramas de Componente podem ajudar a visualizar a estrutura do sistema antes e depois de grandes refatorações.

No XP, a UML é vista como um meio para um fim – o de facilitar a compreensão e a colaboração – e não como um fim em si mesma. A ênfase está no código funcionando e na comunicação eficaz, com os diagramas servindo como auxiliares visuais temporários [4].

Na Figura 4 temos as Práticas do Extreme Programming (XP) e a Tabela 4 é apresentado a Relação entre Práticas do XP e o Uso da UML [Fonte: Elaborado pelo autor com base em [4], [7]].

Prática do XP	Uso da UML
Design Simples	Esboços rápidos de Diagramas de Classe ou Sequência para discutir soluções.
Programação em Pares	Diagramas informais para alinhar o entendimento entre os programadores.
Testes (TDD)	Diagramas de Atividade para modelar cenários de teste.
Refatoração	Diagramas de Componente para visualizar a estrutura antes e depois.
Metáfora	Diagramas de Pacote ou Componente para representar a arquitetura do sistema.
Integração Contínua	Diagramas de Implantação para visualizar o ambiente de integração.

Tabela 4: Relação entre Práticas do XP e o Uso da UML.

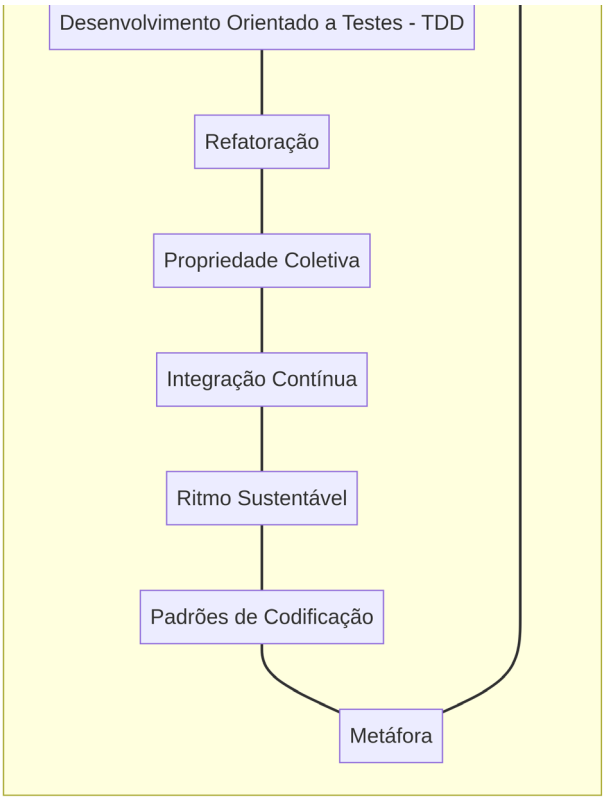

Figura 4: Práticas do Extreme Programming (XP);

ESTUDOS DE CASO E IMPACTO NA INDÚSTRIA

A UML não é apenas uma ferramenta teórica; sua aplicação prática tem demonstrado benefícios tangíveis em diversos setores da indústria de software. Empresas de grande porte e projetos complexos utilizam a UML para gerenciar a complexidade, melhorar a comunicação e otimizar o ciclo de vida do desenvolvimento.

Estatísticas de Impacto

Estudos e pesquisas na área de engenharia de software têm quantificado o impacto positivo da UML:

- **Redução no Tempo de Manutenção:** Projetos que mantêm uma documentação UML atualizada podem experimentar uma redução de até **30%** no tempo necessário para atividades de manutenção. Isso se deve

a melhor compreensão do código e da arquitetura do sistema, facilitada pelos diagramas [8].

- **Custos de Manutenção:** A fase de manutenção é notoriamente a mais cara do ciclo de vida do software, consumindo entre **60% e 70%** do custo total. O uso de diagramas de classe e sequência, por exemplo, melhora significativamente a compreensão do código, o que se traduz em menor esforço e custo de manutenção [8].
- **Adoção na Indústria:** A UML se estabeleceu como um padrão de fato na indústria. Já nos anos 2000, cerca de **30%** das pequenas empresas na Áustria utilizavam a UML, e essa adoção se expandiu globalmente, sendo fundamental para a comunicação e o design em equipes de desenvolvimento [9].

Casos de Sucesso Reais

Grandes organizações e projetos de alta complexidade têm se beneficiado da aplicação da UML:

- **NASA (Jet Propulsion Laboratory - JPL):** A NASA, através do seu Jet Propulsion Laboratory (JPL), desenvolveu um perfil UML específico para "State Analysis" (Análise de Estados). Isso permitiu que engenheiros de software utilizassem diagramas de estados e atividades da UML para refletir diretamente os requisitos de engenharia de sistemas em missões críticas. Essa abordagem melhorou a rastreabilidade dos requisitos e contribuiu para a redução de erros em sistemas de software de alta segurança [10].
- **Boeing:** A Boeing, uma das maiores fabricantes aeroespaciais do mundo, emprega a Model-Based Systems Engineering (MBSE) com perfis UML para integrar o design de sistemas complexos. Isso inclui o desenvolvimento de aeronaves comerciais e a participação em projetos como a Estação Espacial Internacional. A UML facilita a comunicação entre equipes multidisciplinares e a gestão da complexidade inerente a esses sistemas [10].
- **Indústria Automotiva e de Sistemas Embarcados:** Em setores onde a precisão e a segurança são primordiais, como na indústria automotiva e em sistemas embarcados, a UML é amplamente utilizada. Diagramas de máquina de estados e de atividade são cruciais para modelar o comportamento de sistemas de tempo real, garantindo que as transições de estados e as respostas a eventos ocorram conforme o esperado, o que é vital para a segurança e o desempenho [11].

Esses exemplos demonstram que a UML, quando aplicada corretamente, é uma ferramenta poderosa para o desenvolvimento de software, contribuindo para a qualidade, a manutenibilidade e o sucesso de projetos complexos.

CONCLUSÃO

A UML 2.5.1 continua sendo uma ferramenta indispensável para a modelagem de software, oferecendo uma linguagem rica e padronizada para representar sistemas complexos. Sua evolução, com a versão 2.5.1 consolidando e aprimorando a especificação, demonstra o compromisso da OMG em manter a linguagem relevante e eficaz para as necessidades da indústria. A compreensão e aplicação correta dos diversos diagramas da UML permitem uma comunicação mais eficiente, um design mais robusto e uma documentação mais clara, contribuindo significativamente para o sucesso de projetos de desenvolvimento de software.

REFERÊNCIAS

SANCHES, H. M. **O uso da Unified Modeling Language (UML) na modelagem de software**. Atena Editora, 2023. DOI: 10.22533/at.ed.1282321084.

OMG. **Unified Modeling Language (UML) Specification, Version 2.5.1**. 2017. Disponível em: https://www.omg.org/spec/UML/2.5.1/About-UML/. Acesso em: 24 fev. 2026.

UML-DIAGRAMS.ORG. **UML 2.5 Diagrams Overview**. Disponível em: https://www.uml-diagrams.org/uml-25-diagrams.html. Acesso em: 24 fev. 2026.

GUEDES, G. T. **UML 2.5: Do Requisito à Solução**. Novatec Editora, 2018.

KRUCHTEN, Philippe. **The Rational Unified Process: An Introduction**. 3. ed. Addison-Wesley Professional, 2004.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**. Scrum.org, 2020. Disponível em: https://scrumguides.org/scrum-guide.html. Acesso em: 24 fev. 2026.

BECK, Kent. **Extreme Programming Explained: Embrace Change**. 2. ed. Addison-Wesley Professional, 2004.

DESHMUKH, R.; KUMAR, S. **Unified Modeling Language (UML) and Its Contribution to Software Maintenance Efficiency**. *Advances in Software Engineering*, Mar. 2023. Disponível em: https://www.researchgate.net/publication/389585787_Unified_Modeling_Language_UML_and_Its_Contribution_to_Software_Maintenance_Efficiency. Acesso em: 24 fev. 2026.

JCSPL. **UML é popular na produção de software**. 31 mar. 2006. Disponível em: https://jcspl.net/2006/03/31/uml-e-popular-na-producao-de-software/. Acesso em: 24 fev. 2026.

NASA. **NASA Study on Flight Software Complexity / UML for State Analysis**. 2015. Disponível em: https://www.nasa.gov/uploads/2015/04. Acesso em: 24 fev. 2026.