Revista Jornada Acadêmica

Ciências Exatas e da Terra

A TRANSIÇÃO DE METODOLOGIAS TRADICIONAIS PARA METODOLOGIAS ORIENTADAS A OBJETOS – ARTEFATOS DA ANÁLISE DE REQUISITOS

SILVEIRA, Luciana Almeida da[1]*

[1]Centro Universitário Fieo, Osasco, São Paulo, Brasil.

*e-mail de correspondência: lualmeidasouza@hotmail.com - Luciana Almeida da Silveira

Recebido: 05/01/2026 - Aceito: 11/03/2026 - Publicado online: 16/03/2026

Resumo

O processo de evolução de sistemas legados é um tópico de recentes pesquisas na área de engenharia de requisitos. Organizações vêm se defrontando continuamente com a necessidade de mudar e/ou melhorar seus sistemas computacionais. Neste processo de evolução, as maiores mudanças envolvem a transição do uso de metodologias tradicionais tais como a Análise Estruturada e Essencial para Metodologias Orientadas a Objetos tais como o Processo Unificado (UP). É consensual que um processo de evolução de sistemas legados não deve se basear unicamente na avaliação do código fonte, mas também nos demais artefatos que possam facilitar o processo de elicitação, análise e documentação de requisitos. Neste contexto, uma das principais dificuldades reside na falta de diretrizes que permitam mapear informações de sistemas legados representadas em modelos tais como Diagramas de Fluxos de Dados para novos artefatos utilizados em metodologias orientadas a objetos, como Casos de Uso em UML. Neste trabalho, é apresentado um breve estudo das principais metodologias de desenvolvimento de software utilizadas atualmente tanto no âmbito acadêmico quanto industrial, bem como uma proposta de um conjunto de diretrizes para apoiar desenvolvedores no processo de evolução de sistemas legados mapeando informações contidas em Diagramas de Fluxos de Dados e código fonte para Casos de Uso em UML.

Palavras-chave: Análise de Requisitos; Metodologia de Desenvolvimento de Software; Análise Estruturada; Processo Unificado; Metodologias Orientadas.

Abstract

The process of legacy system evolution has become a topic of recent research in the field of requirements engineering. Organizations are continually facing the need to change and/or improve their computational systems. In this evolution process, the most significant changes involve the transition from traditional methodologies, such as Structured and Essential Analysis, to Object-Oriented methodologies, such as the Unified Process (UP). It is widely acknowledged that the evolution of legacy systems should

not rely solely on source code evaluation, but also on other artifacts that may facilitate the elicitation, analysis, and documentation of requirements. In this context, one of the main challenges lies in the lack of guidelines that allow the mapping of information from legacy systems represented by models such as Data Flow Diagrams to new artifacts used in object-oriented methodologies, such as UML Use Cases. This paper presents a brief study of the main software development methodologies currently used in both academic and industrial contexts, as well as a proposal of a set of guidelines to support developers in the legacy system evolution process by mapping information contained in Data Flow Diagrams and source code to UML Use Cases.

Keywords: *Requirements Analysis; Software Development Methodology; Structured Analysis; Unified Process; Oriented Methodologies.*

INTRODUÇÃO

Diante da crescente complexidade dos sistemas computacionais e da necessidade constante de sua evolução, observa-se que a qualidade do produto de software está fortemente associada à adequação do processo de desenvolvimento adotado. No entanto, muitos projetos ainda enfrentam dificuldades decorrentes da escolha inadequada ou da execução incompleta de processos e metodologias de desenvolvimento, especialmente no que se refere às atividades de Engenharia de Requisitos.

Problemas relacionados à elicitação, análise e documentação de requisitos, bem como à ausência ou inadequação da documentação do sistema, figuram entre as principais causas de falhas em projetos de software e comprometem diretamente sua manutenibilidade e evolução. Esse cenário torna-se ainda mais crítico no contexto de sistemas legados, nos quais organizações necessitam promover a transição de metodologias tradicionais, como as abordagens estruturadas e essencial, para metodologias orientadas a objetos mais modernos, como o Processo Unificado.

Nesse contexto, este trabalho tem como objetivo analisar os principais processos e metodologias de desenvolvimento de software atualmente utilizados e propor um conjunto de diretrizes que auxiliem a evolução de sistemas legados, permitindo o mapeamento de informações presentes em Diagramas de Fluxo de Dados

e no código-fonte para Casos de Uso em UML, apoiando a migração para abordagens orientadas a objetos de forma sistemática e estruturada.

Referencial Teórico

O estudo fundamenta-se em livros clássicos e artigos científicos relacionados às metodologias de análise e modelagem de sistemas, incluindo:

- Análise Estruturada
- Análise Essencial
- Engenharia Reversa
- Processo Unificado (UP)
- UML e modelagem por Casos de Uso

Entre os autores que embasam a discussão destacam-se:

- Tom DeMarco
- Gane & Sarson
- McMenamin & Palmer
- Grady Booch
- Ivar Jacobson
- Roger Pressman

MATERIAIS E MÉTODOS

Trata-se de uma pesquisa aplicada, de natureza qualitativa, caracterizada como um estudo analítico e propositivo, desenvolvido com base em revisão bibliográfica e análise de artefatos de software

O estudo insere-se no campo da Engenharia de Requisitos e da evolução de sistemas legados.

Artefatos de sistemas legados

Foram analisados diferentes artefatos de sistemas legados, incluindo:

- Diagramas de Fluxo de Dados (DFDs):
  - ➔ Diagrama de Contexto (nível 0)
  - ➔ DFDs de níveis inferiores
- Especificações de processos (Português Estruturado)
- Documentação existente de sistemas legados (em diferentes estados: inexistente, precária ou atualizada)
- Código-fonte (como apoio à engenharia reversa)

Notações e Padrões Utilizados

A modelagem orientada a objetos utiliza:

- UML (Unified Modeling Language)
- Casos de uso como artefato central da abordagem orientada a objetos para representação funcional do sistema.

Procedimentos Metodológicos

Inicialmente foi realizado um estudo das principais abordagens de desenvolvimento de software, incluindo:

1. **Levantamento e Análise Comparativa de Metodologias**

- Estudo das metodologias tradicionais (Análise Estruturada e Essencial)
- Estudo das metodologias orientadas a objetos (com foco no Processo Unificado)
- Comparação dos artefatos produzidos por cada abordagem

2. Classificação dos Sistemas Legados

Os sistemas foram classificados conforme o estado da documentação existente:

- sistemas sem documentação
- sistemas com documentação precária
- sistemas com documentação atualizada

Após a classificação, houve definição de estratégias distintas de transição conforme cada cenário.

3. Engenharia Reversa Conceitual

Para compreensão da estrutura e funcionamento dos sistemas legados foram utilizadas técnicas de engenharia reversa, incluindo:

- análise do código-fonte (quando necessário)
- análise das entradas e saídas do sistema
- validação das informações com stakeholders

Essa etapa possibilitou organizar a base de informações necessária para a modelagem do sistema.

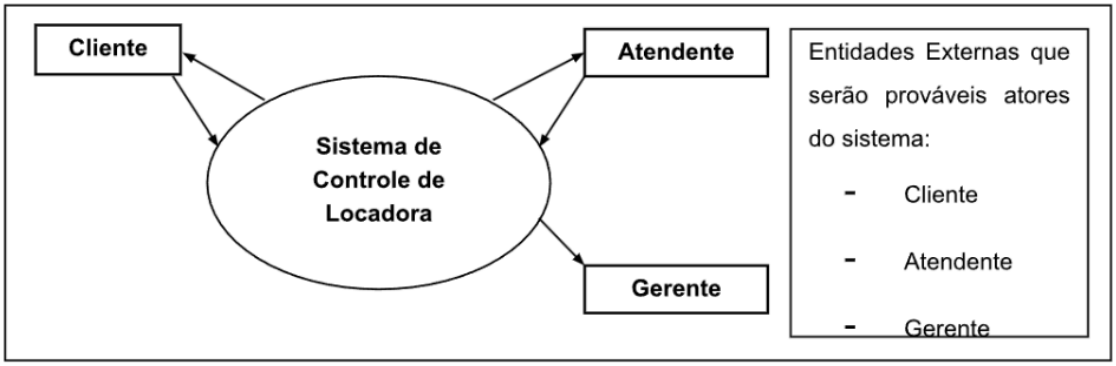
RESULTADOS E DISCUSSÃO

A partir das análises realizadas, foi desenvolvido um conjunto de diretrizes metodológicas destinado a orientar a conversão de artefatos de metodologias estruturadas para modelos orientados a objetos.

Diretriz A – Identificação de atores a partir do Diagrama de Contexto

Esta diretriz tem por objetivo relacionar as entidades externas como prováveis atores do sistema legado, conforme pode-se observar na Figura 1. Esta informação deverá *ser* validada com a Diretriz E. Nessa diretriz todas as entidades externas estarão relacionadas como possíveis atores do sistema.

Figura 1 – Diretriz A – Análise do Diagrama de Contexto



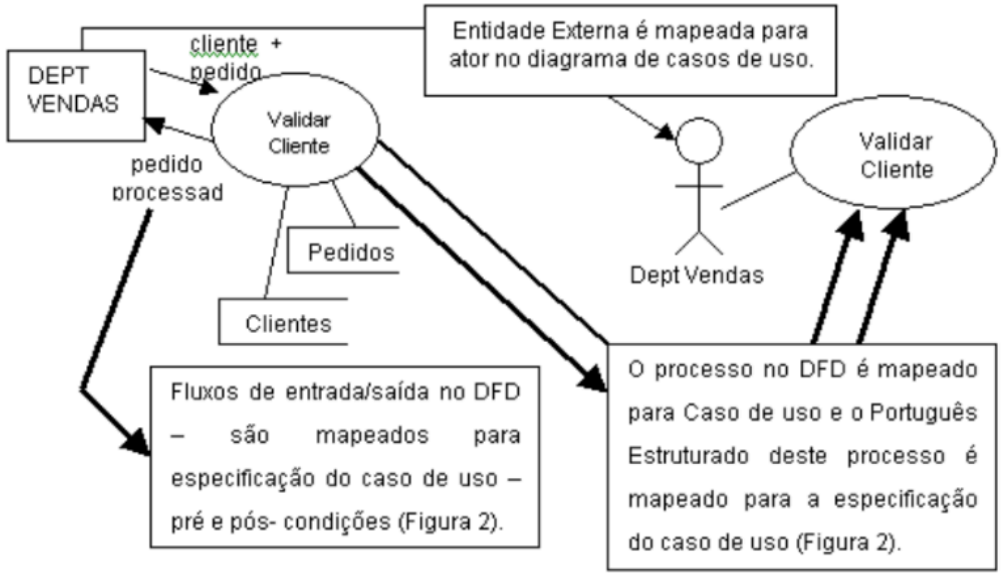
Fonte: O autor, 2026.

Diretriz B – Conversão de DFDs em Casos de Uso

Esta diretriz propõe a conversão dos Diagramas de Fluxo de Dados (DFDs) em Casos de Uso em UML. O procedimento inclui três subdiretrizes:

- **Subdiretriz B1** - Para cada DFD de último nível do sistema legado deve ser elaborado um caso de uso correspondente, como exemplificado na Figura 2.

Figura 2 – Conversão de DFD para Caso de Uso

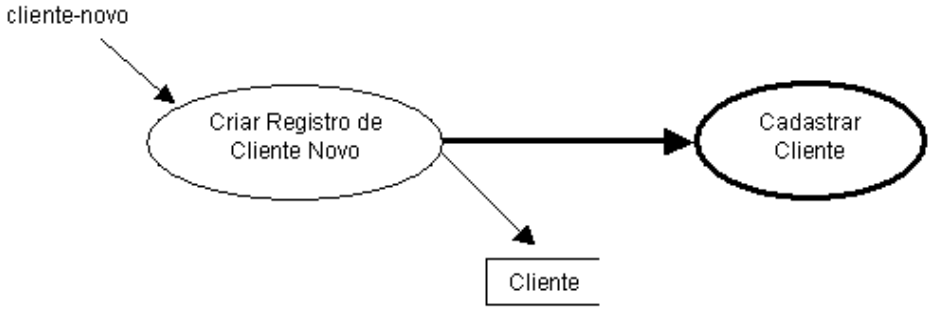


Fonte: O autor, 2026.

- **Subdiretriz B2** - Deve-se analisar a especificação do processo do DFD.

Caso o DFD não possua Entidade Externa, ele é o resultado de um outro processo anterior que o originou. Neste caso, deve ser definido apenas como um caso de uso do tipo <<include>> ou <<extend>>, o qual será chamado em outro caso de uso. A validação deve ser realizada considerando o processo que originou o fluxo, ou seja, deve-se ter o cuidado de validá-lo com o DFD que o chamou, e assim incluí-lo no processo de conversão (Figuras 3 e 4).

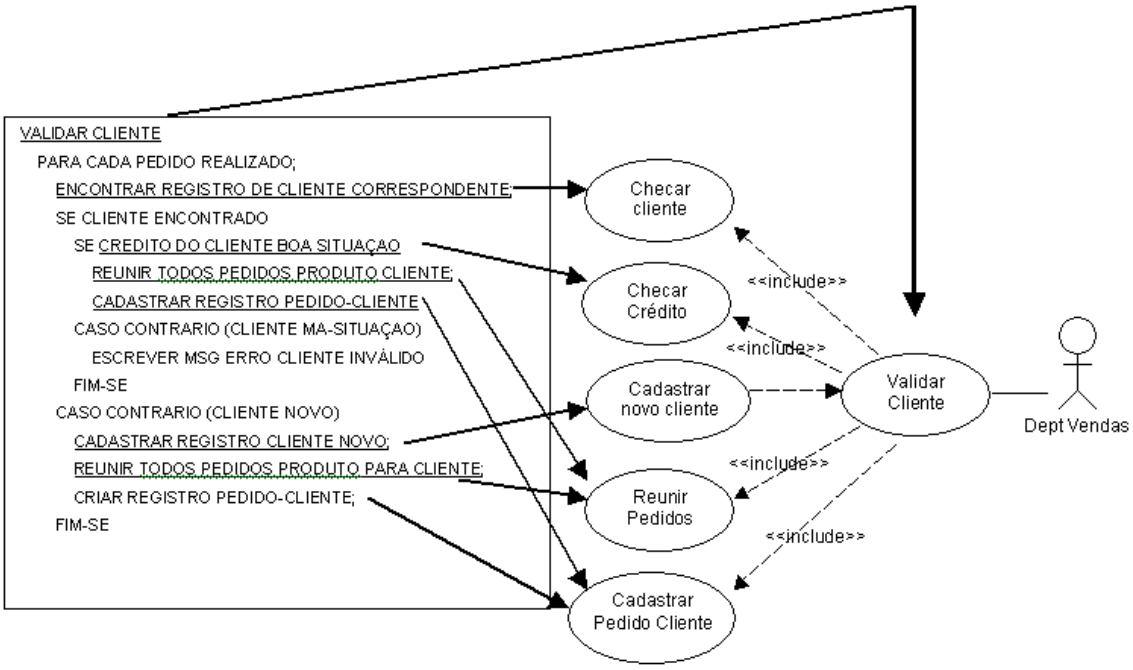
Figura 3 – Conversão de DFD para Caso de Uso – Especial



Fonte: O autor, 2026.

Normalmente, a especificação do processo indica quais são os passos necessários para se atingir um determinado objetivo. Deste modo, pode-se a partir da especificação do processo analisado já incluir no caso de uso correspondente as colaborações e generalizações indicadas de acordo com a Figura 4.

Figura 4 – Análise e conversão da especificação do processo Validar Cliente.



Fonte: O autor, 2026.

- **Subdiretriz B3 -** Analisar os fluxos de entrada e saída do DFD, bem como a especificação do processo do DFD.

Este passo é importante para validar os depósitos de dados identificados no DFD (Figuras 2, 4 e Tabela 1). Observa-se, na Tabela 1, que o caso de uso Validar Cliente é descrito com base nas informações contidas na especificação do processo do DFD (Figura 4).

Tabela 1 – Especificação do Caso de Uso

Caso de Uso	**Validar Cliente**
Objetivo	Este caso de uso descreve o procedimento para que um cliente seja validado de acordo com sua situação decorrente do crédito.
Prioridade	(x) Essencial - ( ) Secundário
Pré-Condições	Dados do Cliente e Pedido do cliente incluído.
Pós-Condições	Pedido do Cliente Processado.
Cenário Principal	1. O caso de uso inicia quando Dept Vendas requer a validação do cliente 2. O Sistema verifica a existência do cliente (caso de uso checar cliente é incluído) 3. O Sistema verifica o crédito do cliente (caso de uso checar crédito é incluído) 4. O sistema reúne todos os pedidos do cliente (caso de uso reunir pedidos é incluído) 5. O sistema cadastra todos os pedidos do cliente (caso de uso criar pedido cliente é incluído)
Fluxos Secundários	-
Caso de Uso Pai	-
Caso de Uso Subordinado	Checar Cliente Checar Crédito Reunir Pedidos Criar Pedido

Fonte: O autor, 2026.

Definiu-se o objetivo do caso de uso analisado, sua prioridade, indicando se sua execução é essencial ao sistema, as pré-condições existentes para sua execução, as pós-condições após a sua execução e o cenário principal juntamente com os fluxos secundários, indicando alternativas de execução.

Descreve-se também os casos de uso pai e subordinados, os quais são detectados no decorrer da conversão dos DFDs.

No passo 1 do cenário principal (*O caso de uso inicia Quando DeptVendas requer a validação do* cliente) deve-se indicar o responsável pela execução do mesmo. Apesar da especificação do processo do DFD não indicar o ator, o DFD possui como Entidade Externa o *DeptVendas*, logo, ele é incluído como ator do Caso de Uso.

Nos outros passos é realizada uma referência a especificação do processo, como por exemplo, em *Se cliente encontrado* no DFD, que foi representado pelo passo 2 *O sistema verifica a existência do cliente*.

Diretriz C – Organização das Entradas e Saídas do Sistema

Esta diretriz tem por objetivo relacionar e validar as entradas e saídas do sistema. Este passo requer o auxílio da equipe ou responsável pela manutenção do sistema, que indicará para cada entrada (em tela) e saída (relatório existente) o usuário responsável pela informação. Deste modo, o engenheiro de software terá uma relação de todos os atores (*stakeholders*) que são afetados.

Diretriz D – Validação dos *Stakeholders*

Esta diretriz tem por objetivo validar os Stakeholders apontados pela diretriz D. Deste modo, o engenheiro de software deve comparar com os atores já identificados pela diretriz B e realizar as alterações necessárias nos casos de uso identificados.

Diretriz E – Organização e Estruturação dos Casos de Uso (*include*, extend, generalização)

Esta diretriz tem por objetivo organizar os diagramas de caso de uso de acordo com as notações propostas pela UML. Casos de uso devem ser agrupados em pacotes ou pela especificação de relacionamentos de generalização, inclusão e extensão, existentes entre os mesmos (BOOCH et al., 1999).

O trabalho apresentou inicialmente um estudo das principais metodologias de desenvolvimento de sistemas, com o objetivo de compreender os fatores envolvidos no processo de migração de projetos desenvolvidos por meio da Análise Estruturada e da Análise Essencial para metodologias orientadas a objetos. A partir desse estudo, foi proposta a elaboração de um conjunto de diretrizes destinadas a apoiar engenheiros de

software e de requisitos no processo de transição entre metodologias tradicionais e orientadas a objetos.

Os resultados evidenciam que as informações existentes na documentação de sistemas legados, em especial aquelas representadas por Diagramas de Fluxo de Dados (DFDs), podem ser reutilizadas de forma sistemática para apoiar a construção e a descrição de Casos de Uso em UML. Observou-se ainda que a aplicação das diretrizes propostas deve considerar o grau e a qualidade da documentação disponível, uma vez que sistemas legados apresentam diferentes níveis de detalhamento e consistência documental.

Além disso, verificou-se que o processo de transição entre abordagens tradicionais e orientadas a objetos envolve não apenas aspectos técnicos, mas também uma mudança cultural significativa dentro das organizações. Aspectos como rastreamento de requisitos, modelagem organizacional e tratamento de requisitos não funcionais, pouco explorados nas abordagens estruturadas e essenciais, emergem como fatores relevantes a serem incorporados gradualmente ao processo de desenvolvimento.

CONCLUSÃO

Conclui-se que a evolução de sistemas legados para metodologias orientadas a objetos representa um desafio complexo, que vai além da simples adoção de novas técnicas de modelagem. As diretrizes propostas neste trabalho demonstraram-se relevantes ao oferecer um apoio sistemático para o reaproveitamento de informações existentes em DFDs, contribuindo para a construção de Casos de Uso em UML e para a migração gradual entre paradigmas de desenvolvimento.

Os achados reforçam a importância de considerar a documentação existente como um ativo estratégico no processo de evolução de sistemas legados, bem como a necessidade de amadurecimento organizacional para a adoção bem-sucedida de novas abordagens. Como contribuição adicional, este trabalho estabelece uma base conceitual para pesquisas futuras mais abrangentes, que incluem a associação entre modelagem organizacional e funcional, bem como o desenvolvimento de ferramentas

automatizadas de apoio à conversão de DFDs para Casos de Uso, ampliando o potencial de aplicação prática das diretrizes propostas.

DECLARAÇÃO DE DISPONIBILIDADE DE DADOS

Os dados utilizados e gerados neste estudo encontram-se integralmente apresentados no corpo do artigo. A pesquisa possui caráter teórico-metodológico, baseada em revisão bibliográfica, análise de artefatos conceituais e proposição de diretrizes, não envolvendo a geração ou o uso de conjuntos de dados experimentais externos.

AGRADECIMENTOS

Agradeço ao Professor Dr. Paulo Sérgio Muniz Silva do Centro Universitário FIEO, pelo auxílio na revisão do trabalho.

REFERÊNCIAS

1. Booch G, Rumbaugh J, Jacobson I. **The Unified Modeling Language**. Reading: Addison-Wesley; 1999.
2. Chikofsky EJ. **Reverse engineering and design recovery: a taxonomy**. IEEE Software. 1990 Jan.
3. DeMarco T. **Análise estruturada e especificação de sistemas**. Rio de Janeiro: Campus; 1989.
4. Gane C, Sarson T. **Structured systems analysis**. New York: Improved System Technologies; 1977.
5. Jacobson I. **Object-oriented software engineering: a use case driven approach**. California: Addison-Wesley; 1992.
6. Jacobson I, Booch G, Rumbaugh J. **The unified software development process**. Reading: Addison-Wesley; 1999.

7. McMenamin S, Palmer JF. **Análise essencial de sistemas**. São Paulo: Makron Books; 1984.
8. Mittra SS. **A road map for migrating legacy systems to client/server**. Journal of Software Maintenance: Research and Practice. 1995;8(2):117-130. Disponível em: <http://www.dei.unicap.br/~almir/seminarios/2004.2/ts04/reengenharia/reeng_software.htm>. Acesso em: 20 dez 2025.
9. Newcomb P, Kotik G. **Reengineering procedural into object-oriented systems**. In: Proceedings of the Second Conference on Reverse Engineering; 1995; Toronto, Canada. p. 237-249. Disponível em: <http://www.unicentro.br/editora/revistas/recen/v1n2/Reengenharia.pdf>. Acesso em: 22 dez 2025.
10. Pressman RS. **Engenharia de software**. São Paulo: Makron Books; 1995.
11. Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Campus; 1994.
12. Scott K. **O processo unificado explicado**. Porto Alegre: Bookman; 2003.
13. Sneed HM. **Migration of procedurally oriented COBOL programs in an object-oriented architecture**. In: Proceedings of the Conference on Software Maintenance; 1992; Orlando, USA. p.105-116. Disponível em: <http://www.dei.unicap.br/~almir/seminarios/2004.2/ts04/reengenharia/reeng_software.htm>. Acesso em: 20 dez 2006.
14. Kontoya G, Sommerville I. **Requirements engineering: processes and techniques**. Chichester: Wiley; 1998.
15. Yourdon E. **Análise estruturada moderna.** Rio de Janeiro: Campus; 1992.