Rastreabilidade de Requisitos na Engenharia de Software

Rastreabilidade de Requisitos serve para orientar desde a escrita dos requisitos até a implementação do código, para garantia da qualidade do software.

Sergio Fred Andrade

RASTREABILIDADE DE REQUISITOS NA ENGENHARIA DE SOFTWARE

RESUMO

O processo de desenvolvimento de software segundo a engenharia de software, inicia com a elicitação e análise dos requisitos. A qualidade do software depende das especificações conforme a análise dos requisitos e o respeito às regras do negócio e suas retrições. Portanto, o tratamento eficiente dos requisitos e a capacidade de rastrear sua implementação ao longo de todo o ciclo de vida do desenvolvimento de sistema são etapas importantes para a garantia de qualidade do software. O objetivo deste e-book é explorar em profundidade um dos instrumentos mais poderosos para conectar os requisitos do software à sua especificação no processo de implementação, usando o paradigma da orientação a objetos. Assim, apresentamos a Matriz de Rastreabilidade de Requisitos (MRR), um instrumento para ilustrar e orientar desde a escrita dos requisitos funcionais e não funcionais até o seu uso com implementação do código com linguagem de programação válida. Ao longo destas páginas, desvendaremos não apenas o que é uma MRR e como construí-la, mas também sua intrínseca relação com a qualidade de software, demonstrando como sua aplicação correta pode prevenir defeitos, facilitar a gestão de mudanças, garantir a cobertura completa dos requisitos e, em última análise, entregar produtos que realmente atendam as expectativas dos usuários e das partes interessadas.

1. Apresentação

Requisitos de software são divididos em duas amplas categorias: requisito funcional e requisito não funcional. O primeiro é vinculado às tarefas que o software faz de acordo com as transações com os dados do domínio da aplicação, como inserção, atualização, remoção, consultas e o processamento dessas informações. A segunda categoria representa funcionalidades ou aspectos que não tem vínculo direto com os dados do domínio da aplicação, como autorização e permissão para acesso ao sistema, tempo de inicialização, segurança das informações, outros recursos como relógio, calculadora, interoperabilidade de sistemas e outros.

Todos esses requisitos devem ser analisados, modelados e implementados nas especificações para que se tenha garantia de funcionalidade e consistência dos dados, respeitando as regras do negócio e restrições do sistema organizacional. Essa correspondência alia-se à qualidade aceitável do software.

Os requisitos devem ser bem definidos e representar as operações do mundo real que serão retratados no software e essa garantia pode estar representada na MRR. A Matriz de Rastreabilidade de Requisitos é uma tabela que detalha os aspectos dos requisitos para servir de orientação ao codificador sobre os elementos fundamentais na construção das classes, com seus atributos, métodos e interfaces necessárias.

Então, qualidade do software é diretamente dependente da qualidade dos requisitos que o definem. Requisitos devem ser representados na especificação implementada para resolver o problema da operação organizacional do domínio, a exemplo de: vendas, finanças, gestão escolar ou outros. Se essa especificação for falha, ambígua, incompleta ou inconsistente, é praticamente impossível construir um software que atenda às expectativas e possua os atributos de qualidade desejados.

Imagina-se um requisito funcional descrito vagamente como "O sistema deve ser ágil de operar". Não existem critérios objetivos no requisito, como essa "agilidade" será implementada e, principalmente, verificada? A falta de clareza compromete diretamente a característica de Usabilidade (ISO 25000, 2022). Da mesma forma, um requisito não funcional de desempenho que retrata apenas "O sistema deve ser rápido" é vago. Qual a especificação de rápido? Quais as situações aplicadas? A ausência de especificidade impede a garantia da eficiência de desempenho. Se requisitos de segurança forem omitidos ou mal descritos, a característica de segurança estará comprometida. Se os requisitos conflitantes não forem identificados e resolvidos na fase de análise, a correção funcional e a credibilidade do sistema serão afetadas.

A incompletude é outro inimigo da qualidade. Se um requisito crucial para uma funcionalidade não for capturado, o software entregue ao cliente não atenderá plenamente as necessidades do usuário, impactando a adequação funcional. A inconsistência entre requisitos e as regras do negócio do sistema pode levar a comportamentos inesperados e falhas, prejudicando a confiabilidade.

Portanto, a engenharia de requisitos eficaz, que produz especificações claras, completas, consistentes, corretas, viáveis e testáveis, é o alicerce sobre o qual a qualidade do software é construída. É essencial que tanto os requisitos funcionais (o quê o sistema faz) quanto os não funcionais (como o sistema faz, abrangendo as características de qualidade da ISO 25010) sejam elicitados, analisados, especificados e validados conforme a necessidade do cliente.

Para guiá-lo nesta jornada de aprendizado, organizamos o conteúdo de forma lógica e informativa. Começamos estabelecendo as bases conceituais da Rastreabiilidade de requisitos (Capítulo 2), de Requisitos funcionais e não funcionais e O processo de rastreabilidade de requisitos (Capítulo 2), abordando o que são requisitos, seus tipos, ciclo de vida e os desafios de sua análise. Em seguida, está descrito no e-book, os Elementos da matriz de rastreabilidade (Capítulo 3) detalhando sua definição, importância, tipos e construção. Os capítulos seguintes exploram a integração da rastreabilidade com diferentes métodos de desenvolvimento e processos de Análise de requisito pela técnica da semântica (Capítulo 4), mostra o processo da análise de requisitos e a Construção da matriz por semântica do requisito funcional (Capítulo 4) e a especificação adequada de software (Capítulo 4).

O principal objetivo deste documento é servir como um guia técnico abrangente sobre a Matriz de Rastreabilidade de Requisitos, conceitos e orientação prática. Pretende-se que este e-book seja útil para estudantes, analistas de requisitos, gerentes de projeto, desenvolvedores e testadores que buscam aprimorar seus processos de engenharia de requisitos.

Portanto, esperamos que sirva como um guia valioso, capacitando você a entender e aplicar a rastreabilidade de requisitos como um meio eficaz para construir software de alta qualidade. Boa leitura!

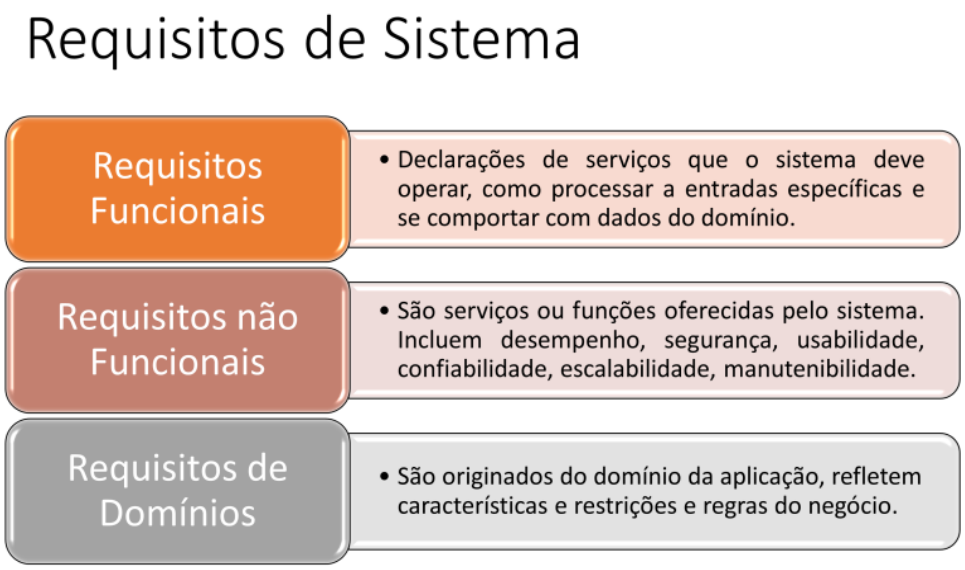
2. Rastreabilidade de Requisitos

Conceitos de domínio e requisitos de software

Em geral um sistema de software representa um domínio ou um mini-mundo real. Trata-se de um processo organizacional com limites ou fronteiras bem definidos, que tem uma operação e objetivo úteis para uma Organização ou Pessoa. Exemplos: 1) uma biblioteca universitária tem limites definidos da sua atuação que são as informações sobre livros, bibliotecários, estantes, salas de preparação de livros, empréstimos, devoluções, estudantes, professores, todos com suas devidas características próprias. Uma característica ou propriedade de um objeto citado, é somente dele e não mais de outro objeto. 2) uma clínica médica tem os objetos como, médico, recepcionistas, pacientes, salas de consultas, salas ambulatoriais, salas de exames clínicos e outros. Todos os objetos com suas características próprias. 3) agenda telefônica que tem objetos como pessoa de contato, número de telefone, endereço, evento, duração do evento, local do evento e outros, e todos esses objetos têm propriedades ou características que os descrevem.

Requisitos

A precisão e a clareza na descrição dos requisitos são cruciais, pois quaisquer ambiguidades ou omissão de atributos pode resultar em retrabalho significativo no modelo de dados e na implementação sistêmica, atrasos no cronograma e, em última instância, na entrega da aplicação que causará prejuízo ao usuário final. Os requisitos podem ser expressos de diversas notações, desde declarações em linguagem natural até modelos formais e diagramas, cada um com suas vantagens e desvantagens dependendo do contexto do projeto e da complexidade do sistema.



Requisitos Funcionais

Requisitos funcionais especificam as funções que um sistema deve executar. Eles descrevem as operações que o sistema realizará, detalhando as ações com dados e transações em base de dados, serviços ou tarefas que o software operacionalizará para o usuário. Esses requisitos são diretamente relacionados às funcionalidades do negócio e às interações do usuário com o sistema. Em outras palavras, eles especificam o comportamento do sistema em termos de entradas, processamento e saídas.

A clareza e a precisão são cruciais na especificação de requisitos funcionais. Um requisito funcional bem definido deve ser:

- **Atômico:** Deve descrever uma única funcionalidade, sem agrupar múltiplas operações.
- **Completo:** Deve conter todas as informações necessárias para sua implementação, sejam condicionais ou de processamento.
- **Consistente:** Não deve contradizer outros requisitos, e ser clara e objetiva.
- **Verificável:** Deve ser possível testar sua funcionalidade segundo sua especificação.
- **Não ambíguo:** Deve ter uma única interpretação, evitando termos vagos e dúbios.

Requisitos não Funcionais

Requisitos não funcionais (RNFs) especificam as operações ou restrições que o sistema deve possuir sem vínculo com os dados do seu domínio ou suas funcionalidades essenciais. Eles descrevem "como" o sistema deve se comportar, abordando aspectos como desempenho, segurança, usabilidade, confiabilidade, escalabilidade, manutenibilidade e compatibilidade. Embora não definam o que o sistema faz, os RNFs são importantes como complementos aos requisitos funcionais.

A norma ISO/IEC/IEEE 29148 (2018) trata da importância de tornar requisitos não funcionais mensuráveis e rastreáveis sempre que possível. As principais características dos RNFs são:

- **Desempenho:** Velocidade, tempo de resposta e utilização eficiente de recursos.
- **Segurança:** Proteção contra acesso não autorizado, integridade dos dados, autenticação, autorização, auditoria.
- **Usabilidade:** Facilidade de uso, aprendizado, eficiência de operação, satisfação do usuário, acessibilidade.
- **Confiabilidade:** Disponibilidade, tolerância a falhas, capacidade de recuperação.
- **Escalabilidade:** Capacidade de lidar com um aumento no volume de usuários, dados ou transações.
- **Manutenibilidade:** Facilidade de modificar, corrigir ou adaptar o software.
- **Portabilidade:** Capacidade do software de operar em diferentes ambientes (sistemas operacionais, navegadores, plataforma).
- **Compatibilidade:** Capacidade de coexistir e interagir com outros sistemas.

Conceito sobre rastreabilidade de requisitos

A rastreabilidade de requisitos no contexto da engenharia de software é a capacidade de documentar e seguir o ciclo de vida de um requisito, desde sua origem na análise inicial até sua especificação e implementação, sobre os testes e validação no sistema final. Não se trata apenas de criar documentos da análise, mas de estabelecer uma rede de conexão entre a análise do requisito e sua implementação, que permita uma compreensão holística do projeto e suas interdependências. Segundo Pressman (2021), a rastreabilidade não apenas garante que cada requisito seja implementado e testado, mas também facilita a análise de impacto e a gestão de mudanças em projetos dinâmicos.

A rastreabilidade é um processo para identificar e mostrar os elementos e etapas da análise de requisitos e aspectos de melhoria nos processos de especificação desses requisitos e projeto de software. Quando é realizado com critério técnico é considerado uma engenharia de requisitos, pois permite especificar e demonstrar a construção ou manutenção de requisitos apropriados à aplicação adequada com linguagens de programação.

O processo de rastreabilidade é demonstrado pela Matriz de Rastreabilidade de Requisitos (MRR), que não é apenas um documento técnico, mas uma estratégia gerencial que demonstra a tecnicidade da equipe de desenvolvimento na engenharia dos requisitos.

Infelizmente esse processo geralmente não aparece muito nos projetos de software, por desmerecimento de analistas em promover melhor documentação ao projeto, mas apresenta uma importância muito grande para garantia da qualidade do software, pois ajuda a estabelecer e manter ligações claras entre os requisitos e todas as outras etapas do ciclo de desenvolvimento. Ele oferece uma série de benefícios direcionados à qualidade do produto, a eficiência do processo e a satisfação dos usuários da aplicação.

Oferecem informações cruciais no processo de desenvolvimento que podem indicar uma análise de impacto de projeto, estimar o esforço, o custo e o tempo necessários para implementar o sistema e indicar decisões eficazes para viabilidade e a manutenção do produto.

A rastreabilidade também induz a reduzir o risco de novos defeitos ou de afetar funcionalidades existentes devido a alterações não programadas ou manutenções indevidas.

O processo de rastreabilidade de requistos

A MRR possibilita o processo de rastrear os elementos para especificação do código desde a fase de descrição do requisito, de projeto de software com a diagramação de casos de uso, de sequência e de classes, modelo de dados, até a definição lógica dos objetos relacionais, atributos e métodos operacionais, com padrões para referenciamentos de get/set, cardinalidades em instanciação de objetos e instruções CRUD (*Create, Read, Update, Delete*) em SQL para transações em bases de dados. Pois, o propósito fundamental da rastreabilidade é garantir que cada unidade programada de codificação tenha uma correspondência clara e objetiva vinculada ao requisito que o originou.

Portanto, possibilita uma visão completa e integrada das interdependências entre os requisitos e todas as especificações do projeto.

Trata-se de um processo binário e bidirecional (requisito X especificação), ou seja, na análise do requisito, no projeto de software e diagramação, para descrever conceitualmente a qualidade do produto, associado à lógica do modelo de dados, estrutura das classes e construção dos métodos, com objetivo não apenas verificar se um requisito foi implementado, mas também se uma parte do código ou um teste está realmente relacionado ao requisito referenciado ou a outro requisito incluso ou em extensão (*include/extend*).

Esse processo apresentado aqui neste e-book, é mostrado através da MRR, que representa os aspectos importantes entre a fase de análise de requisitos e o projeto de software, que irão nortear o trabalho para os implementadores na especificação do modelo de dados, das classes, métodos e interface do projeto codificado, fornecendo uma visão clara e concisa dos componentes implementados.

Embora possam parecer simples, sua eficácia reside na capacidade de revelar lacunas, inconsistências e impactos de mudanças de forma visual e organizada. Segundo Sommerville (2005), as matrizes de rastreabilidade são uma forma clara de relacionar requisitos e artefatos, além de apoiar na identificação de lacunas e inconsistências durante o desenvolvimento.

3. Elementos da matriz de rastreabilidade

A criação de uma Matriz de Rastreabilidade eficaz é um processo flexível que deve ser adaptado ao contexto e às necessidades do projeto. No entanto, alguns passos gerais podem guiar sua construção, ou, como mostrado neste e-book, uma matriz padrão é orientada para facilitar o processo de desenvolvimento, teste e manutenção de software.

O Objetivo da Rastreabilidade é determinar os elementos necessários para o processo da engenharia de requisitos e como os requisitos podem ser descritos com clareza e definir o escopo e o nível de detalhe necessários para a especificação implementada.

Os elementos indicados por esta metodologia para se chegar à garantia da qualidade do software, são os seguintes:

1) **Código do requisito:** é um identificador descrito alfanumérico que serve para representar o requisito nos diagramas, no projeto de software e para implementação;

2) **Caso de uso:** diagrama na Linguagem Unificada de Modelos (UML) que indica os processos dos requisitos, descritos em resumo, para ilustrar em leitura rápida o processo operacional do software;

3) **Entidades:** indicação da entidade do modelo conceitual ou lógico no Diagrama Entidade-Relacionamento, que representa um objeto concreto ou abstrato do domínio. Cada requisito deve indicar ao menos duas entidades relacionadas;

4) **Atributos:** indicação de atributos principais de cada entidade citada, que representam as propriedades de cada entidade, conforme o modelo lógico no Diagrama Entidade-Relacionamento. Cada entidade deve ter aproximadamente dez atributos descritivos - não chaves, inclusos;

5) **Condições (*include/extend*):** *include* é um aspecto na associação entre processos que pode representar uma inclusão – quando o novo processo contém uma diretiva obrigatória de funcionalidade, a exemplo de uma condicional como verificação ou comparação de dados. *Extend* é um aspecto na associação entre processos que representa um novo processo facultativo, como uma condição opcional de verificação ou comparação de dados.

6) **CRUD:** são operações representadas nos métodos implementados para cumprir as transações nas bases de dados. O (C) corresponde a uma instrução SQL de insert para inserir dados numa tabela; o (R) corresponde a uma instrução SQL de select para consulta ou recuperação de dados de tabelas; o (U) é uma instrução de SQL para update para editar dados em tabelas do banco de dados; e, o (D) representa uma instrução de SQL para delete ou remoção de registros no banco de dados.

7) **Get/Set:** em orientação a objetos usa-se propriedades com dois blocos de código: um acessador *get* e um modificador *set*. O método para o acessador *get* é executado quando a propriedade é lida, geralmente quando há uma instrução SQL em *select*. O método para o modificador *set* é executado quando um novo valor é atribuído à propriedade, em geral para cumprir instruções de SQL *uptdate* e *insert*.

8) **SQL:** são as instruções implementadas de forma embutida nos métodos para transações no banco de dados em SQL de acordo com CRUD e com *get/set*. Essas instruções não necessárias e permitem que se crie, acesse, modifique e remova dados em banco de dados, quando irá cumprir as transações na base de dados.

4. Análise de requisito pela técnica da semântica

A técnica usada para construção do requisito obedece a uma regra hierárquica baseada na semântica descritiva, mostrada na **Figura 1**. A aplicação dessa técnica de análise exige um aspecto conceitual na elaboração do enunciado do requisito, onde considera-se as partes integrantes do descritivo os elementos necessários para auxiliar na abstração do modelo de dados e na construção da classe especificada. Por exemplo:

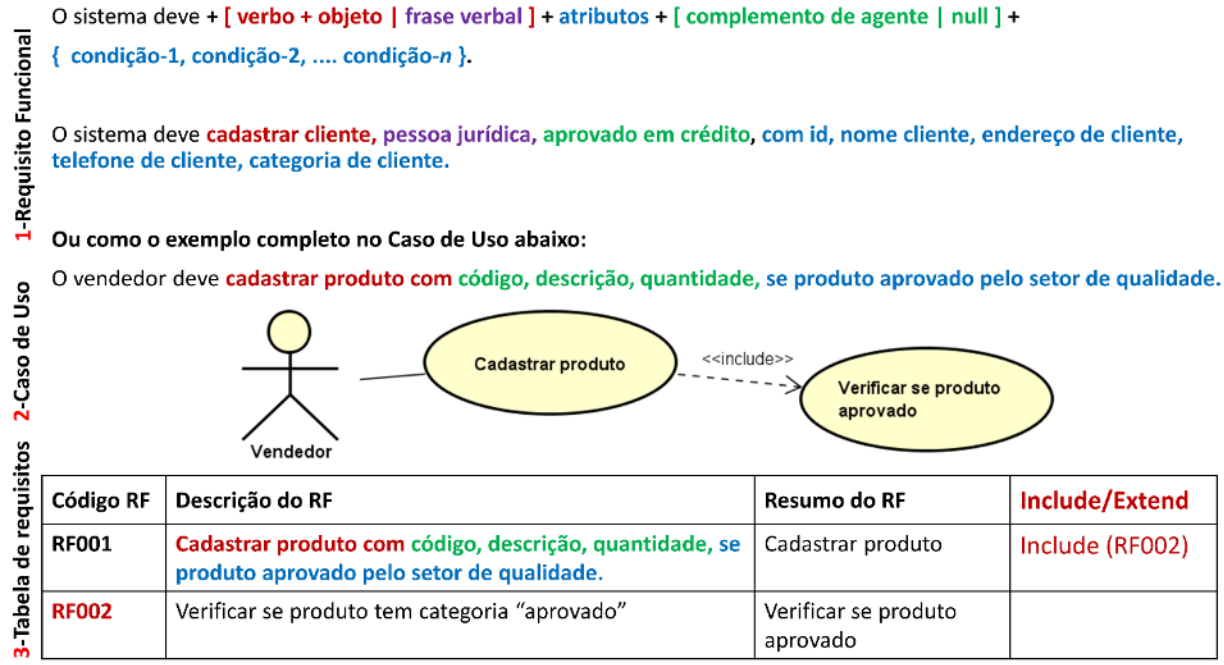
O sistema deve ***+ [verbo + objeto | frase verbal] + atributos + [complemento de agente | null] + {condição-1, condição-2, ... condição-n}.***

O sistema deve ***cadastrar cliente, pessoa jurídica, com id, nome cliente, endereço de cliente, telefone de cliente, categoria de cliente, se aprovado para crédito.***

Construção da matriz por semântica do requisito funcional

Conforme pode-se observar na Figura 1, a elaboração da MRR inicia-se na 1ª. Etapa **[1-Requisito Funcional]**, com o termo "*O sistema deve...*" ou "*A aplicação deve...*" ou termo parecido, seguido de um verbo no infinitivo que indica ação de fazer, mais um objeto para completar a sentença, mais os atributos desse objeto, mais o complemento dos atributos se necessário, mais as condições para ação desse requisito.

A 2ª. etapa **[2-Caso de Uso]**, é a representação desse requisito na linguagem UML com o diagrama Caso de Uso. Observa-se que cada ação está num processo (elipse). No exemplo abaixo temos um ator (vendedor) que opera duas ações que são interligadas por um *<<include>>* (inclusão).



Código RF	Descrição do RF	Resumo do RF	Include/Extend
RF001	Cadastrar produto com código, descrição, quantidade, se produto aprovado pelo setor de qualidade.	Cadastrar produto	Include (RF002)
RF002	Verificar se produto tem categoria "aprovado"	Verificar se produto aprovado	

Figura 1 – Descrição do requisito a partir do Diagrama caso de uso.

Esses elementos citados, são dispostos nas colunas da MRR pelo código do requisito e pode fazer referência a outro requisito em outra linha, ou não, a depender se há *include/extend*.

A MRR deve ser acompanhada por uma **[3-Tabela de Requisitos]** (da Figura 1) que traz a descrição completa dos requisitos com seu código conforme a técnica da semântica, narrada anteriormente. Essa necessidade justifica-se para se obter melhor organização na notação dos requisitos já que a matriz é apresentada com 8 colunas e torna-se muito extensa para tal.

Diagrama caso de uso e a matriz de rastreabilidade

Conforme está mostrado na Figura 2 a MRR é uma abstração de uma equação conceitual com as técnicas do Diagrama Caso de Uno mais a Tabela de Requisitos mais o Diagrama Entidade-Relacionamento que é igual ao CRUD mais mapeamentos *get/set* mais instruções de SQL. (Casos de uso + Diagrama de requisitos + DER = CRUD + get/set + SQL)

Essa junção é importante para aproximar a análise dos requisitos das fases de projeto e implementação das especificações com uso de linguagem de programação.

A ideia é que quando o programador for implementar ele deva seguir a MRR e isso facilite na codificação para especificar classes, atributos e métodos.

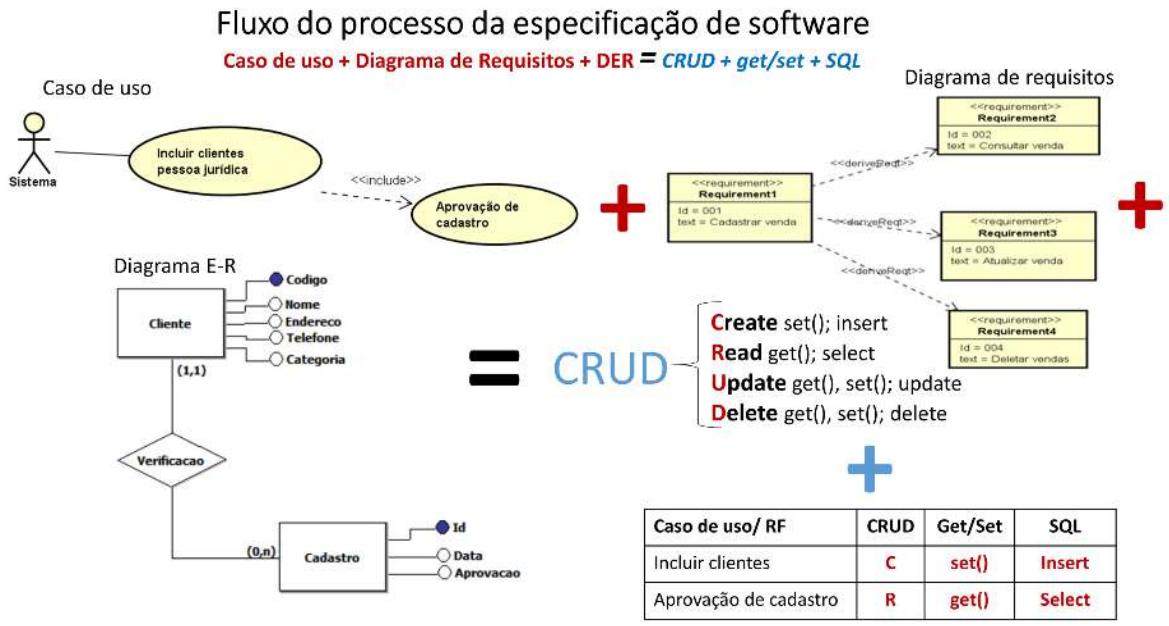
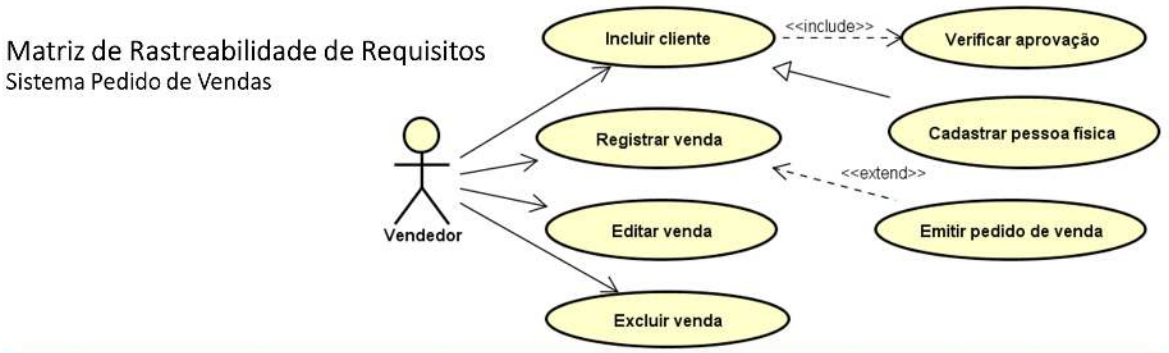


Figura 2 – Roteiro do processo da especificação de software.

Um exemplo completo de diagrama de caso de uso com processos *include* e *extend* está mostrado na Figura 3.

O diagrama de caso de uso mostra a funcionalidade em requisitos que fazem parte do sistema que será projetado, é uma excelente ferramenta indicativa dos requisitos funcionais e serve para documentar o início do processo da engenharia de requisitos. Portanto, os processos de caso de uso são vistos como requisitos e eles serão especificados na implementação. Isso ajuda no trabalho do programador porque já estão indicados nas colunas da MRR com modelo de dados, entidades, atributos e métodos que serão mapeados nas classes programadas.

Observa-se na Figura 3 que o diagrama caso de uso está totalmente referenciado na MRR, orientando o trabalho do programador quando este implementará o modelo de dados e as classes do sistema, com associação entre os objetos, o CRUD, *get/set* e instruções de SQL.



RF	Caso de uso/ ER	Entidade DER	Atributos DER	include/extend/ generalization	CRUD	Get/Set	SQL
RF001	Incluir clientes	Cliente	Idcliente, nome, endereco	Include(**RF002**)	C	set	Insert
RF002	Aprovação de cadastro	Cliente, Aprovacao	Idcliente, status_aprovacao	*	R,U	get,set	Select,Update
RF003	Editar clientes	Cliente	Idcliente, nome, endereco	*	R,U	get,set	Select, Update
RF004	Excluir clientes	Cliente	Idcliente, nome, endereco	*	R,D	get,set	Select, Delete
RF005	Cadastrar Cliente Pessoa Física	Cliente, PessoaFisica	Idcliente, nome, endereco, status, aprovacao	Generalizagion (**RF001**)	C	set	Insert
RF006	Registrar Pedido Venda	Cliente, PessoaFisica, Aprovacao, Pedido	Idcliente, nome, endereco, status, aprovacao, Pedido	Extend (**RF007**)	C	set	Insert
RF007	Emitir Pedido Venda	Cliente, PessoaFisica, Aprovacao	Idcliente, nome, endereco, status, aprovacao, Pedido	*	C,R	set,get	Insert, Select

Figura 3 – Matriz de Rastreabilidade de Requisitos completa.

A implementação segundo a matriz de rastreabilidade

Em um contexto de orientação a objetos, o mapeamento de objetos refere-se à forma como os dados de uma tabela no modelo relacional corresponde a um objeto e são representados e armazenados. *Getters* e *setters* são mecanismos que ajudam a controlar esse mapeamento, garantindo que o acesso e a modificação dos dados sejam feitos de maneira segura e consistente.

Getters (Acessadores):

Função: Recupera o valor de um atributo, na maioria das vezes de uma propriedade mapeada de atributos de tabelas em banco de dados.

Exemplo: Se uma classe tem um atributo <nome>, o método seria *getNome().*

Importância: Permite que outras partes do código acessem o valor do atributo de forma controlada, sem acesso direto.

Setters (Modificadores):

Função: Define ou altera o valor de um atributo, na maioria das vezes de uma propriedade mapeada de atributos de tabelas em banco de dados.

Exemplo: Se uma classe tem um atributo <nome>, o método seria *setNome(String novoNome).*

Importância: Permite que outras partes do código modifiquem o valor do atributo de forma segura, potencialmente com validações.

Cada entidade do modelo de dados, no DER – lógico, é mapeado para uma classe na especificação do código, a classe carrega os atributos mapeados da tabela do banco de dados com seu respectivos tipos de dados, e, deve ter os métodos modificadores *set()* e os acessadores *get()*, para cada atributo.

A Figura 4 mostra esse fluxo que inicia com o mapeamento dos atributos da tabela – *Cliente* (Nome, Idade, Credito), que são representados na classe *Cliente*, com dados carregados no objeto *cli* (instanciado na classe *Cliente*), com os métodos *getNome(), getIdade(), getCredito, setNome(), setIdade(), setCredito*.

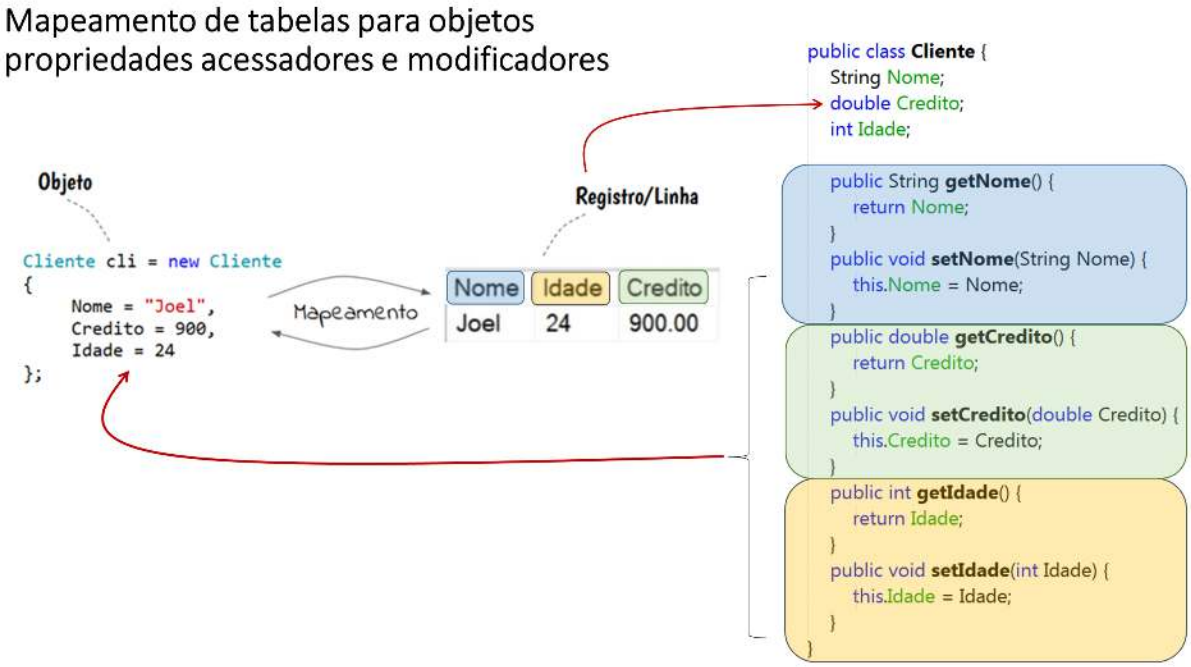


Figura 4 – Mapeamento de tabelas para objetos.

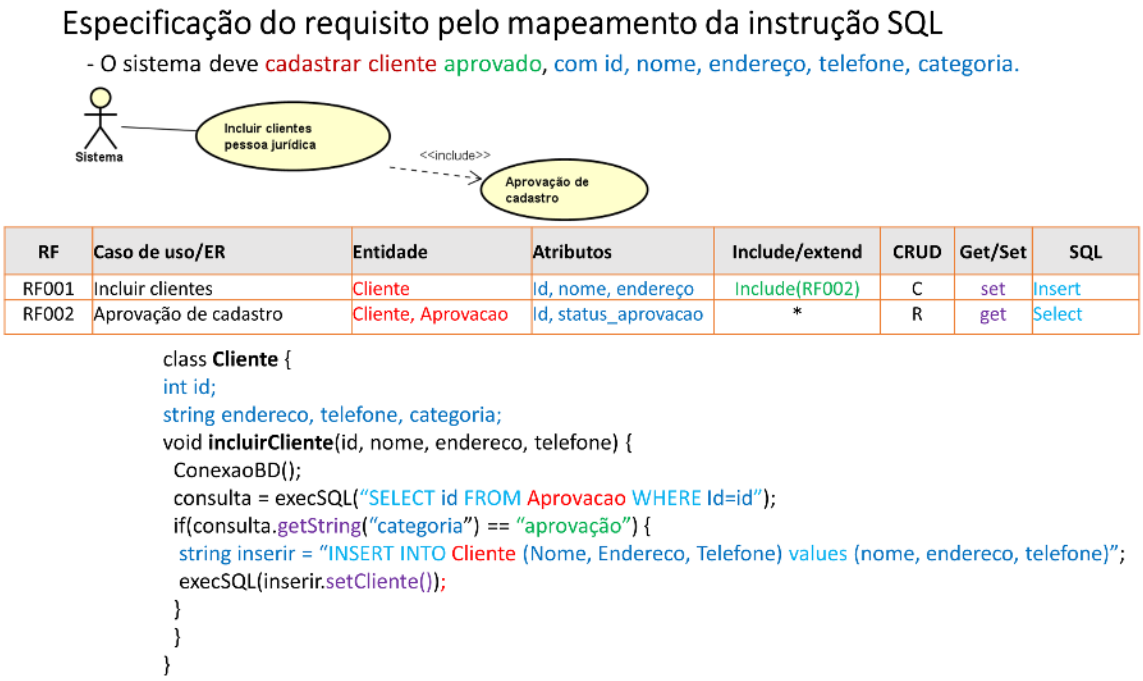
Por outro lado, a aplicação da linguagem para manipulação dos registros de banco de dados, o SQL, é necessária para inserir dados (insert), atualizar dados (update), remover dados (delete) e consultar dados (select). O que faz uma referência entre os getters/setters, as transações CRUD e as instruções SQL para manipulação de dados.

Especificação adequada de software

Genericamente a especificação de software é a conversão de algoritmos de acordo com os requisitos funcionais e não funcionais, o projeto de software e por linguagens de programação, para definir de forma técnica as propriedades dos dados, as funcionalidades e comportamentos dos objetos abstraídos, que serão instanciados durante a execução do programa.

É uma tarefa importante no desenvolvimento de software, pois estabelece a notação padronizada para o reconhecimento computacional e execução das tarefas, e a objetiva descrição para representação dos requisitos do domínio retratado no sistema.

Tecnicamente é um processo da computação lógica sobre a construção algoritímica e respeito aos requisitos e ao projeto de software, com uso da sintaxe/semântica das linguagens de programação, para definir a estrutura de dados, os atributos, comportamentos e a integração de componentes da aplicação que será desenvolvida. É uma etapa crucial no desenvolvimento de software, pois estabelece uma base clara e compreensível para todas as partes envolvidas no projeto – análise, projeto, codificação, teste e manutenção de software.



RF	Caso de uso/ER	Entidade	Atributos	Include/extend	CRUD	Get/Set	SQL
RF001	Incluir clientes	Cliente	Id, nome, endereço	Include(RF002)	C	set	Insert
RF002	Aprovação de cadastro	Cliente, Aprovacao	Id, status_aprovacao	*	R	get	Select

```
class Cliente {
int id;
string endereco, telefone, categoria;
void incluirCliente(id, nome, endereco, telefone) {
 ConexaoBD();
 consulta = execSQL("SELECT id FROM Aprovacao WHERE Id=id");
 if(consulta.getString("categoria") == "aprovação") {
  string inserir = "INSERT INTO Cliente (Nome, Endereco, Telefone) values (nome, endereco, telefone)";
  execSQL(inserir.setCliente());
 }
 }
}
```

Figura 5 – Especificação de requisitos com a MRR.

Na Figura 5 é mostrado o Diagrama Caso de Uso, a MRR e a especificação da implementação numa linguagem representativa didática para orientação a objeto, correspondendo ao requisito RF001 e sua inclusão na RF002.

Observa-se que todos os elementos formadores da classe *Cliente (atributos, método)* estão facilmente representados e aliados entre a MRR – com tabela do BD, CRUD, *get/set* e SQL, e a respectiva codificação dessa classe.

Vale lembrar que uma classe é um conceito fundamental que define uma categoria ou agrupamento de objetos afins para instanciação de ocorrências. Ela especifica quais são os atributos (características) e métodos (comportamentos) que a instanciação dos objetos desse agrupamento devem possuir. Atributos são as informações ou propriedades que caracterizam cada objeto da classe (por exemplo, nome, idade e endereço de um cliente), e, os métodos são as ações ou comportamentos que os objetos podem realizar (comprar um móvel, devolver o móvel, trocar o móvel).

5. Considerações finais

A ideia deste e-book busca contribuir de forma clara e objetiva para os procedimentos das etapas de análise e engenharia de requisitos no processo de desenvolvimento de software.

É sugerida a Matriz de Rastreabilidade de Requisitos (MRR) com o propósito de mostrar o entendimento da notação dos elementos necessários na engenharia de requisitos e como esses elementos podem ser implementados na especificação e construção das classes necessárias da orientação a objetos.

Foi mostrada a metodologia de como descrever semanticamente os requisitos funcionais e como podem ser codificados na prática com os elementos padrões. De acordo com as ilustrações nas figuras, é mostrado ao leitor exemplos de como proceder para obtenção do projeto e especificação do software para a garantia da qualidade esperada.

Em suma, espera-se que este trabalho contribua com as equipes de desenvolvimento de software para a indústria e possa apoiar no ambiente ensino-aprendizagem para cursos na área da computação com entendimento mais claro do processo de desenvolvimento de software.

REFERÊNCIAS

BOOCH, Grady. UML: guia do usuário. Elsevier Brasil, 2006.

CMMI (Capability Maturity Model Integration). CMMI® for Development, Version 1.3. Preface, SEI, CMU, 2006.

COUGO, Paulo Sérgio (1997). Modelagem conceitual e projeto de bancos de dados. Campus.

DEITEL, Harvey M.; DEITEL, Paul J. Cómo programar en Java. Pearson educación, 2003.

GODOY, Vinícius. Programação Orientada a Objetos I. IESDE BRASIL SA, 2019.

ISO/IEC/IEEE 29148:2018(en). Systems and software engineering — Life cycle processes requirements engineering. Disponível em: https://www.iso.org/obp/ui/en/#iso:std:72089:en

ISO/IEC 25000 and AI product quality measurement perspectives. In: IWESQ/APSEDEI@ APSEC. 2022. p. 21-29. Disponível: https://www.iso.org/standard/64764.html

MACHADO, Felipe Nery Rodrigues. Banco de Dados: Projeto e Implementação. São Paulo: 2004.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software-9. McGraw Hill Brasil, 2021.

Quest. Acessado em: https://www.quest.com/products/toad-edge/, Acesso em: 20.dez/2022.

SOMMERVILLE, Ian. Requerimientos del software. Ingeniería del software, 7a ed., PEARSON EDUCACIÓN, Madrid, SPA, 2005.

E-BOOK RASTREABILIDADE DE REQUISITOS NA ENGENHARIA DE SOFTWARE

Este e-book mergulha no uso da Matriz de Rastreabilidade de Requisitos (MRR) como uma ferramenta essencial para conectar requisitos de software à sua implementação, explorando sua relação vital com a qualidade do produto final. Ao abordar tanto requisitos funcionais quanto não funcionais, a MRR é apresentada como uma estratégia gerencial que previne defeitos e facilita a gestão de mudanças, garantindo que os sistemas atendam às expectativas dos usuários. Com uma abordagem prática e estruturada, este guia é indispensável para profissionais que buscam aprimorar a eficiência e a clareza no desenvolvimento de software.