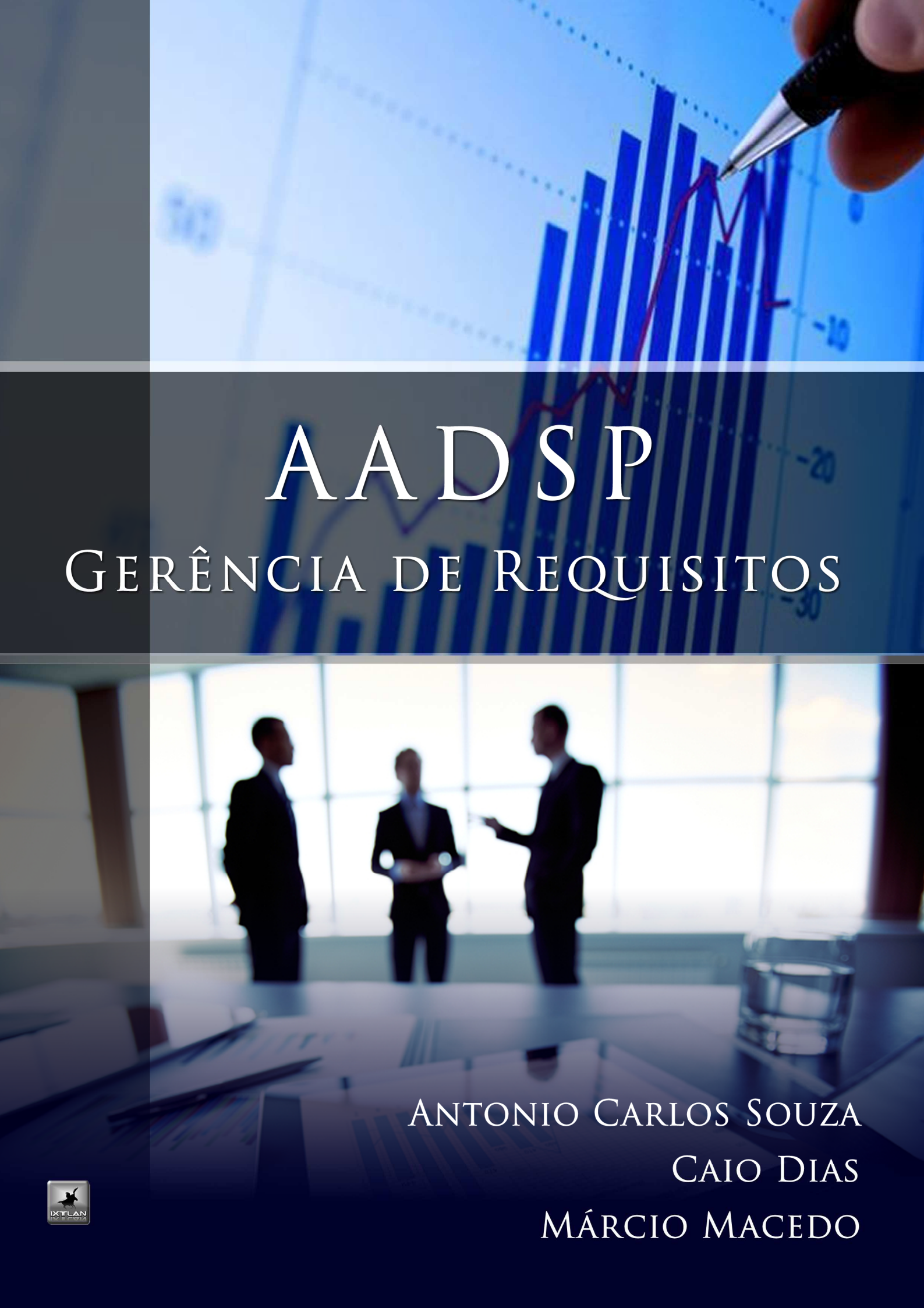

# AADSP

## Gerência de Requisitos

Antonio Carlos Souza
Caio Dias
Márcio Macedo

IXTLAN

# AADSP

## Gerencia de Requisitos

**Antonio Carlos Souza**

**Caio Dias**

**Márcio Macedo**

*São Paulo – 2018*

**Projeto gráfico:**

Editora Ixtlan

**Diagramação:**

Márcia Todeschini

**Capa:**

Gabriel Polizello

Antonio Carlos Souza; Caio Dias; Márcio Macedo

AADSP: Gerencia de Requisitos – São Paulo/SP - Ed. Ixtlan, Agosto/2018

ISBN: 978-85-8197-689-1

1. Ciência da computação 2. Desenvolvimento de softwares

CDD 000

Editora Ixtlan - CNPJ 11.042.574/0001-49 - I.E. 456166992117

# Conteúdo

**1 Introdução** **3**
1.1 Processo de Software . . . . . 3
1.2 Guia AADSP . . . . . 4
1.2.1 Valores dos Artefatos . . . . . 5
1.2.2 Gerências do AADSP . . . . . 6
1.3 Requisitos . . . . . 9

**I Referêncial Teórico** **11**

**2 Revisão Bibliográfica** **13**
2.1 Gerenciamento de Requisitos . . . . . 13
2.1.1 Requisitos . . . . . 13
2.1.2 A Engenharia de Requisitos . . . . . 15
2.1.3 O Gerenciamento dos Requisitos . . . . . 17
2.1.4 As Atividades do Gerenciamento dos Requisitos . . . . 18
2.2 Rastreabilidade . . . . . 19
2.2.1 Conceituando a Rastreabilidade . . . . . 19
2.2.2 Classificação da Rastreabilidade . . . . . 20
2.2.3 A Aplicação da Rastreabilidade de Requisitos . . . . . 22
2.2.4 Metamodelos para Rastreabilidade de Requisitos . . . 25
2.2.5 Metamodelo proposto para a ferramenta T-AADSP Requirements . . . . . 31
2.3 The Next Release Problem . . . . . 33
2.4 Modelos de Maturidade e Requisitos . . . . . 35
2.4.1 Guia MPS.BR . . . . . 35
2.4.2 Guia CMMI . . . . . 37

**II Artefatos 45**

**3 Artefatos e Documentação 47**
3.1 Artefatos . . . . . 47
3.1.1 Documento de Requisitos (Essencial) . . . . . 47
3.1.2 Documento de Mudanças dos Requisitos (Essencial) . . 48
3.1.3 Aprovação dos Clientes (Essencial) . . . . . 49
3.1.4 Comprometimento da Equipe (Importante) . . . . . 49
3.1.5 Rastreabilidade . . . . . 49
3.1.6 Checklist de Aceitação (Importante) . . . . . 50

**III Avaliação AADSP 53**

**4 Avaliação AADSP 55**
4.1 Avaliação do Documento de Requisitos . . . . . 56
4.1.1 Questões Avaliativas . . . . . 56
4.2 Avaliação do Documento de Mudanças dos Requisitos . . . . . 57
4.2.1 Questões Avaliativas . . . . . 57
4.3 Avaliação do Comprometimento da Equipe . . . . . 58
4.4 Avaliação da Rastreabilidade entre Requisitos . . . . . 58
4.4.1 Questões Avaliativas . . . . . 58
4.5 Avaliação da Rastreabilidade entre Requisitos e Artefatos . . . 59
4.5.1 Questões Avaliativas . . . . . 60

**IV Ferramentas 61**

**5 Ferramentas 63**

**V Conclusão 67**

**6 Conclusão 69**

**Índice Remissivo 73**

# Lista de Figuras


2.1 Processo Geral da Engenharia de Requisitos. Adaptado de [1] 15
2.2 Área da Rastreabilidade dentro da Engenharia de Software. Extraído de [1] . . . . . 19
2.3 Pré e Pós Rastreabilidade. Extraído de [1] . . . . . 21
2.4 Rastreabilidade Vertical e Horizontal. Extraído de [1] . . . . . 22
2.5 Artefatos conectados através de um elo. Extraído de [1] . . . . 25
2.6 Metamodelo proposto por [2] adaptado por [1]. . . . . 26
2.7 Metamodelo proposto por Toranzo. Extraído de [3] . . . . . 27
2.8 Metamodelo proposto por Genvigir. Extraído de [1] . . . . . 28
2.9 Metamodelo proposto pela ferramenta T-AADSP Requirements 31
2.10 Exemplo do NRP. Adaptado de [4] . . . . . 34

# Lista de Tabelas


4.1 Definição da meta para avaliação do Documento de Requisitos 56
4.2 Definição da meta para avaliação do Documento de Mudanças dos Requisitos . . . . . 57
4.3 Definição da meta para avaliação do Documento de Comprometimento da Equipe . . . . . 58
4.4 Definição da meta para avaliação do Documento de Rastreabilidade entre Requisitos . . . . . 59
4.5 Definição da meta para avaliação do Documento de Mudanças dos Requisitos . . . . . 59

# Prefácio

A abordagem AADSP - *Adaptive Approach for Deployment of Software Process* - Abordagem Adaptativa para Implantação de Processo de Software, proposta pelo Labrasoft - Laboratório de Desenvolvimento de Software - em 2014 e financiado pelo edital de fortalecimento de pesquisa do IFBA - Instituto Federal da Bahia - tem como alicerce práticas inovadoras em conformidade com o modelo MPS.BR da SOFTEX, práticas dos métodos ágeis, demandas dos profissionais da área de análise e desenvolvimento de sistemas, para acompanhamento dos projetos de *software* por interesse das empresas clientes e das produtoras de *software*.

O objetivo da abordagem AADSP é promover a qualidade de *software* por meio do equilíbrio na utilização de métodos ágeis e a melhoria contínua dos processos de *software*. O AADSP tem quatro pilares:

1. Bem-estar e realização das pessoas envolvidas;
2. Transparência e melhoria da comunicação com o cliente ou consumidor;
3. Pré-avaliação dos projetos da organização com base em artefatos;
4. Melhoria contínua do retorno por investimento;

Esse livro também apresenta outros guias além do AADSP tais como o guia MPS.BR - Melhoria de Processo de Software Brasileiro - que é bastante utilizado em grandes empresas brasileiras para desenvolvimento e avaliação de *softwares* e serviços, e o CMMI - *Capability Maturity Model Integration*, ou Modelo Integrado de Maturidade em Capacitação - aplicado em âmbito internacional bem como o método utilizado para a avaliação do CMMI, chamado SCAMPI - *Standard CMMI Appraisal Method for Process Improvement*.

Esse é o primeiro livro do guia AADSP e aborda o gerenciamento de requisitos para *softwares* com base nos artefatos estabelecidos pelo guia. O gerenciamento de requisitos deve acompanhar os requisitos durante todas as fases de desenvolvimento do *software*, desde o levantamento dos requisitos (elicitação) junto com os clientes e *stakeholders* até a fase de manutenção do

*software* que apresenta como principal característica a modificação em requisitos já existentes. Visando colaborar e enriquecer a área de Engenharia de Software, o guia AADSP também promove a geração de *Datasets* com dados sobre os requisitos, testes, projetos e colaboradores para serem utilizados na área de Engenharia de Software baseada em busca, do inglês *Search-Based Software Engineering*.

# 1

# Introdução

## 1.1 Processo de Software

A indústria do *software* mudou e mais importante do que desenvolver e entregar *softwares* funcionais, tornou-se obrigatória a entrega de *softwares* com qualidade e que se adaptem rapidamente às mudanças solicitadas pelos clientes [5, 6, 7]. *Softwares* projetados com qualidade não representam apenas satisfação para o(s) cliente(s) mas também um considerável aumento na produtividade, redução dos custos e cumprimento de cronogramas estipulados. Nesse sentido, existem diversas propostas de soluções para auxiliar na qualidade de *software*, tais como modelos, padrões e métodos para o processo de desenvolvimento de software [8, 9].

Visando contemplar e se adequar aos atuais paradigmas de desenvolvimentos de *softwares*, as organizações têm destinado parte de seus recursos para a especialização de seus funcionários e a adequação a processos de *software*. O desenvolvimento de *software* deve ser visto como um conjunto de processos predefinidos e que sejam facilmente adaptáveis, necessitando de um constante acompanhamento, análise, planejamento, execução e melhorias visando atender macro fatores como diferentes tipos de investimentos: financeiro, social, tecnológico, educacional e, principalmente, dos indivíduos participantes da confecção do *software*.

A Engenharia de *Software* busca contemplar as melhores práticas de desenvolvimento de *software* de forma que o produto a ser entregue apresente qualidade e eficiência. Tal área de conhecimento cita diversos modelos de ciclo de vida de *software*, tais como: cascata, espiral, incremental, iterativo, evolucional, prototipação e outros. Esses modelos, geralmente, possuem as atividades de Análise, Projeto, Codificação e Teste [7, 5], sendo que as atividades de Análise e Projeto estão fortemente ligadas a atividades da Enge-

nharia de Requisitos que aborda com maior importância e detalhamento os requisitos. As atividades mencionadas são detalhadas a seguir:

1. **Análise**: fase relacionada ao entendimento inicial do problema que o cliente deseja resolver com a criação do *software*. Essa fase é caracterizada pela obtenção dos requisitos. Segundo Lopes e Sommervile [5, 7] essa atividade visa especificar o problema a ser resolvido, declarando-o e entendendo-o com o auxílio de técnicas desenvolvidas no contexto da Engenharia de Requisitos;

2. **Projeto**: essa fase visa a descrição e avaliação do problema a ser solucionado com o *software*. Os esforços são voltados à transformação dos requisitos iniciais obtidos com os clientes e uma posterior análise de fatores como qualidade e viabilidade antes da codificação [5, 7];

3. **Codificação**: visa construir o *software* com base nos requisitos coletados e analisados nas fases anteriores com a utilização de ferramentas computacionais [5, 7];

4. **Teste**: almeja verificar se o programa construído está de acordo com o projeto e com os requisitos levantados e documentados [5, 7];

De acordo com Genvigir [1], todos os modelos de ciclo de vida de *software* citados anteriormente tem início com a Engenharia de Requisitos, pois ela trata da identificação dos envolvidos e suas necessidades de descoberta dos requisitos e de documentação. Em seu artigo, Lopes [5] traz diversas definições de autores renomados na área da Engenharia de Requisitos e destaca que as atividades ligadas à essa disciplina ocorrem durante todo ciclo de vida do *software* desde as atividades iniciais (elicitação e análise dos requisitos) e as que perduram durante toda a sua vida útil (codificação, testes e evolução).

## 1.2 Guia AADSP

Visando atender as atuais diretrizes do desenvolvimento de *software*, o LABRASOFT, Laboratório de Desenvolvimento de Software, situado no Instituto Federal de Ciência e Tecnologia da Bahia, propõe uma abordagem adaptativa para implantação de processos de desenvolvimento e concepção de *softwares*, o AADSP - *Adaptive Approach for Deployment of Software Process* - de forma que as empresas possam flexibilizar seus processos a depender do contexto socioeconômico e cultural na qual elas se encontrem. Essa abordagem tem como alicerce práticas inovadoras em conformidade com o modelo

MPS.BR, desenvolvido no Brasil pela Softex, práticas de métodos ágeis e o guia de conhecimento PMBOK da PMI - *Project Management Institute*.

O AADSP estima um conjunto de artefatos para que uma empresa possa apresentar, comprovar e estimar constantes melhorias e qualidade em seus processos e *softwares* sendo que esses artefatos são valorados com os seguintes graus de importância: **essencial**, **importante** e **desejável**. O primeiro grupo de artefatos servem como base de implementação do modelo AADSP, deste modo sua continuidade deverá ser garantida objetivando os resultados proposto pela abordagem. O segundo grupo, são consideráveis, todavia não são obrigatórios deste modo sua implementação resultará em resultados adicionais ao modelo. E o último grupo são os artefatos pouco consideráveis, estes não implicam necessariamente na melhoria do processo ou em resultados satisfatórios.

O objetivo da abordagem AADSP é promover a qualidade de *software* por meio do equilíbrio na utilização de métodos ágeis e de melhoria de processos de software. O AADSP tem quatro pilares:

1. Bem-estar e realização das pessoas envolvidas: O grande diferencial nos projetos bem-sucedidos é o comprometimento, colaboração e auto-organização dos profissionais envolvidos (desenvolvedores internos, líderes e gerentes de desenvolvimento e sistemas, *freelancers*, *stakeholders*, clientes ou financiadores do projeto de software);

2. Transparência e melhoria da comunicação com o cliente ou consumidor: O entendimento e participação do cliente no projeto precisa ser facilitado com linguagens comuns para que este entenda o andamento do projeto e o que está sendo feito;

3. A empresa desenvolvedora pode pré-avaliar seus projetos com base em artefatos e uso da ferramenta em conformidade com a AADSP, gerando *datasets* para pesquisa e medição;

4. Melhoria contínua do Retorno por Investimento (ROI) para todos os envolvidos através da reutilização de artefatos, da melhor utilização do tempo e redução de retrabalho;

### 1.2.1 Valores dos Artefatos

De acordo com o AADSP, artefatos são todas as formas de conhecimento produzido pelo homem registradas em algum meio físico ou lógico [10]. AADSP qualifica os artefatos documentais de acordo com seu grau de importância,

desse modo esta abordagem busca implementar estes artefatos de forma adaptativa nas MPEs. Os três graus de importância são:

1. **Essencial**: Artefatos base para implementação do modelo AADSP, de modo que sua continuidade deverá ser garantida no processo de implantação desta abordagem;

2. **Importante**: Artefatos que são consideráveis, todavia não são obrigatórios, assim sua implementação resultará em resultados adicionais ao modelo;

3. **Desejável**: Artefatos pouco consideráveis, estes não implicam necessariamente na melhoria do processo ou em resultados satisfatórios;

### 1.2.2 Gerências do AADSP

Com base nas gerências propostas pelo guia [8], o AADSP catalogou seis gerências que tem como principal objetivo definir um conjunto de artefatos que garantam a qualidade do *softwares*.

#### Gerência de Requisitos e Modelagem

Almeja gerenciar os requisitos e componentes do projeto, controlar a evolução e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto além de proporcionar uma visão comum entre a alta gerência e os *stakeholders* [10]. Tem como principais artefatos:

1. Documento de Requisitos;

2. Documento de Solicitação de Mudanças;

3. Análise de Impacto em modificações em Requisitos;

4. Histórico e Versionamento dos Requisitos;

5. Sinalizar Inconsistências e Correções;

6. Matriz de Rastreabilidade entre Requisitos;

7. Matriz de Rastreabilidade entre Requisitos e Artefatos;

### Gerência de Projetos

Visa estabelecer e manter planos de trabalhos que definam atividades, cronogramas, recursos e os responsáveis pelo projeto a partir dos seguintes artefatos [10]:

1. Termo de Abertura do Projeto: Documento de autorização formal de início do projeto e que possibilita ao gerente do projeto a autoridade para aplicar os recursos necessários para a execução do projeto;

2. Plano do Projeto: Documento que possui todas as ferramentas e metodologias a serem utilizadas para a execução do projeto;

3. Ata de Reuniões;

### Gerência de Configuração e Mudanças

O escopo do projeto e os seus requisitos podem ser modificados a qualquer momento durante o ciclo de vida do *software*. Assim, requisitos adicionais podem ser incorporados no projeto, os requisitos podem ser removidos do projeto e/ou as mudanças podem ser feitas para os requisitos existentes [10]. Para implantação dessa gerência, é necessário fazer uso dos artefatos:

1. Documentos comprobatórios de solicitações de mudanças;

2. Avaliação dos impactos positivos e negativos que uma modificação pode gerar;

3. Uso de repositórios distribuídos para gerenciamento de configuração e mudança;

4. Sistemas de Controle de Versão para melhor gerenciar os documentos código-fonte;

### Gerência de Colaboradores e Stakeholders

Tem como objetivo organizar, gerenciar, delegar responsabilidades para membros envolvidos na construção do *software* sejam eles desenvolvedores ou *stakeholders* que conhecem as regras do negócio [10]. Para tanto, exige-se:

1. Planejamento das necessidades estratégicas da organização e dos projetos para identificar recursos, conhecimentos e habilidades requeridos e, de acordo com a necessidade, planejar como desenvolvê-los ou contratá-los.

2. Para todo e qualquer projeto o conjunto de pré-requisitos para o desenvolvimento do mesmo deve ser elencado para que treinamentos efetivos e objetivos possam ser aplicados;

3. Os artefatos com regras de negócios devem ser amplamente difundidos para todos os indivíduos que possuírem a devida permissão de acesso de acordo com seu cargo e grau de interesse;

4. Dar um contínuo retorno sobre o progresso das atividades para a equipe de desenvolvimento bem como aos *stakeholders* do projeto;

**Gerência de Testes**

Visa desenvolver planos e estratégias para implementação de teste e inspeção de *software* objetivando a melhoria na qualidade e integridade dos dados apresentados nos produtos finais, de acordo com [10]:

1. Elaboração de Planos de Testes para o Projeto. Esses planos de teste podem estar vinculados ao conjunto de funcionalidades e requisitos de um *software*;

2. Utilização de testes automatizados;

3. Designação de um conjunto de colaboradores da empresa para as atividades de descrição de casos testes e implementação dos testes;

4. Utilização das diversas metodologias para aplicação em testes de *softwares* disponíveis no atual contexto tecnológico;

5. Aplicação de ferramentas para inspecionar qualidade do código-fonte desenvolvido de acordo com o padrão adotado pela empresa;

**Gerência de Reuso**

Tem como objetivo fornecer maiores artefatos de usabilidade produzidos pelos projetos por mapeamento e documentação de componentes e outros ativos reutilizáveis de um software, a partir da [10]:

1. Definição da padronização do código-fonte elaborado para cada projeto de *software*;

2. Disponibilização do conjunto de artefatos reutilizáveis em uma base de fácil acesso aos colaboradores, com *backup* incluso;

3. Definição clara de ativos reutilizáveis ou que possam ser reutilizados;

A abordagem AADSP é voltada para a entrega de artefatos, e deste modo, os processos, qualidade e controle de mudanças são comprovados a partir da existência em projetos de *software* que implementem esta metodologia. Os artefatos exigidos pela gerência de requisitos garantem conformidade entre o que foi implementado e as solicitações dos clientes, controle de mudanças, identificação dos requisitos e rastreabilidade dos mesmos. Diferente do guia MPS.BR[8], essa abordagem é mais flexível e pode proporcionar melhorias aos produtos entregues por micro e pequenas empresas e por *freelancers*.

Para que uma empresa possa implantar o gerenciamento de requisitos, independente do guia ou metodologia adotada, é necessário seguir um conjunto de atividades prescritos pela literatura. Essas atividades garantem controle, monitoramento, uniformidade e reuso dos requisitos.

## 1.3 Requisitos

A Engenharia de Requisitos é uma das disciplinas da Engenharia de *Software* que busca promover as melhores práticas e metodologias para a obtenção, análise, validação, gerenciamento e evolução dos requisitos dentro do ciclo de vida de um software [5, 6, 7]. O gerenciamento dos requisitos permite que os responsáveis pelo negócio consigam identificar os requisitos, quais são os *stakeholders*[1] do projeto, rastrear dos requisitos em todos seus níveis de derivação e controlar as mudanças dos requisitos com a justificativa da referida mudança [8].

Os requisitos são funcionalidades que os sistemas devem implementar para atender necessidades do mundo real fornecidos por pessoas que possuem algum problema e desejam resolvê-lo com uso de *softwares* e uma de suas principais características é sua volatilidade e não previsibilidade [5]. Eles podem mudar durante todas as fases de concepção do *software* devido a fatores internos ou externos como melhorias, incompatibilidade, análises de risco, mudanças políticas, mudanças na tecnologia e outros [5].

Atualmente tem-se a convicção que mudanças em requisitos ao longo do processo de desenvolvimento de *software* fazem parte do processo e devem ser bem vistas pois elas objetivam o melhor para os clientes [6]. Segundo Malcher [9], a gerência de requisitos envolve um conjunto de atividades que apoiam a identificação, o controle e rastreamento de requisitos, bem como o gerenciamento de mudanças de requisitos já que estes podem sofrer modificações em qualquer momento ao longo do ciclo de vida do *software*. Já o

[1]Conjunto de pessoas (clientes, gestores e desenvolvedores) interessadas no desenvolvimento do sistema e que muitas vezes são detentoras dos recursos e conhecimentos sobre regras de negócio.

MPS.BR, destaca que o propósito dessa gerência é o acompanhamento dos requisitos do produto e seus componentes e a identificação de inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto [8]. Um dos critérios da gerência de requisitos é a rastreabilidade dos requisitos de um projeto de *software*.

O rastreamento de requisitos é utilizado para prover relacionamentos entre requisitos, fornecedores dos requisitos, arquitetura e implementação final do sistema e possibilita uma adequada compreensão dos relacionamentos de dependência entre os requisitos e os artefatos gerados para atender requisitos tais como diagramas, planos de teste, casos de uso e outros [6, 2, 8, 3, 5]. Em uma pesquisa realizada por Ramesh [2], a rastreabilidade de requisitos não tem sido adequadamente implementada em algumas empresas e quando implementada é feita de forma amadora e superficial.

A ausência do gerenciamento de requisitos pode comprometer a qualidade do *software* que está sendo desenvolvido, pois os requisitos mudam durante todo o ciclo de vida do *software*. O gerenciamento de requisitos proporciona um maior controle dos requisitos com a identificação, rastreamento em artefatos do *software* e o controle de mudanças, contudo, a implantação dessa gerência pode ser custosa e os benefícios que a mesma proporciona podem ser visualizados apenas de médio a longo prazo [5].

Esse livro aborda o gerenciamento de requisitos proposto pelo guia AADSP. Para o gerenciamento de requisitos, esse guia propõe o auxilio aos *stakeholders* do projeto para tomada de decisões que possam trazer impactos negativos a qualidade, custo e orçamento gerados pelas mudanças nos requisitos de forma que não comprometam a entrega do *software* bem como o conjunto de boas práticas de gerenciamento de requisitos previstas pela literatura tais como identificação, controle de mudanças e rastreabilidade dos requisitos. As práticas propostas pelo guia AADSP podem ser aplicada em empresas que desejam melhorar o seu processo de *software* começando por melhorias em seus requisitos bem como atender a critérios do guia MPS.BR [8].

# Parte I

# Referêncial Teórico

# 2

# Revisão Bibliográfica

## 2.1 Gerenciamento de Requisitos

### 2.1.1 Requisitos

Os requisitos representam o conjunto de funcionalidades que um *software* deve atender e são descobertos juntamente aos solicitantes do *software* (*stakeholders*) durante todo o ciclo de vida do *software*, principalmente no início do projeto em entrevistas, *brainstorms*, questionários e outras atividades que buscam fomentar uma melhor compreensão dos problemas a serem resolvidos pelo *software* em questão [5, 7, 2]. Os requisitos de *software* devem possuir um conjunto de características para que sejam considerados bons requisitos de forma que os mesmos possam trazer benefícios ao projeto.

De acordo com o guia MPS.BR [8], os requisitos representam uma capacidade requerida pelos solicitantes do *software* e que deve ser encontrada no produto para resolver um problema, alcançar um objetivo ou atender contratos, padrões, normas e especificações. Apesar dessas definições transmitirem claramente o conceito de requisitos, para Lopes [5] elas ainda são insuficientes para esclarecer aspectos não funcionais tais como desempenho, integridade, disponibilidade e segurança. Segundo Lopes [5], os requisitos podem ser classificados em requisitos funcionais e requisitos não funcionais [7]. Os primeiros são responsáveis por funcionalidades do *software* e o segundo grupo representa como o *software* deve se comportar em questões de desempenho, segurança, integridade e outros aspectos.

Para o **IEEE** [11] e **MPS.BR** [8], bons requisitos de *software* apresentam um conjunto de características para que possam ser avaliados pela equipe técnica para uma posterior avaliação dos clientes, sendo elas:

1. **Identificação única**: um identificador único deve ser atribuído aos

requisitos para que eles possam ser identificados em qualquer projeto da empresa de forma que possam ser rastreados, promovam o reuso e desambiguem a identificação de diferentes requisitos [8];

2. **Clareza**: as especificações dos requisitos devem ser descritas em uma linguagem clara de forma que qualquer interessado no requisito consiga entender a funcionalidade que o requisito expressa [8];

3. **Indubitabilidade**: a descrição do requisito não deve apresentar margem para mais de uma interpretação pelos interessados no projeto. A ambiguidade pode provocar divergências entre o que foi acordado e o que foi implementado no *software* [11];

4. **Completude**: a funcionalidade que o requisito expressa quando o requisito está implementado no produto final e atende a todas as expectativas dos *stakeholders* do projeto [11];

5. **Consistência**: atributo associado ao requisito que foi submetido a análises, negociações, validações, está em conformidade com o interesse dos *stakeholders* [11] e não possui conflitos com outros requisitos [5];

6. **Classificação**: indicam o grau de importância do requisito para o projeto [11];

7. **Implementabilidade**: característica que verifica se as tecnologias para desenvolvimento do *software* oferecidas no contexto ao qual o *software* será desenvolvido atendem as expectativas dos *stakeholders* [5];

8. **Testabilidade**: indica se o requisito pode passar por práticas da Gerência de Teste de *Software* [8];

9. **Rastreabilidade**: é possível rastrear as fontes do requisito, requisitos vinculados a ele, solicitantes, *rationale* e artefatos que foram originados a partir do requisitos [11];

Espindola [12] ressalta que uma grande quantidade de projetos de *software* são cancelados ou falham por não atenderem completamente as necessidades dos *stakeholders*. De acordo com Genvigir [1], requisitos são de extrema importância para o sucesso do *software* e que nenhuma outra parte do desenvolvimento de *software* é tão difícil de corrigir quantos os requisitos. De forma a garantir o sucesso do projeto, o guia MPS.BR [8] sugere uma boa comunicação com os clientes para garantir um entendimento uniforme das

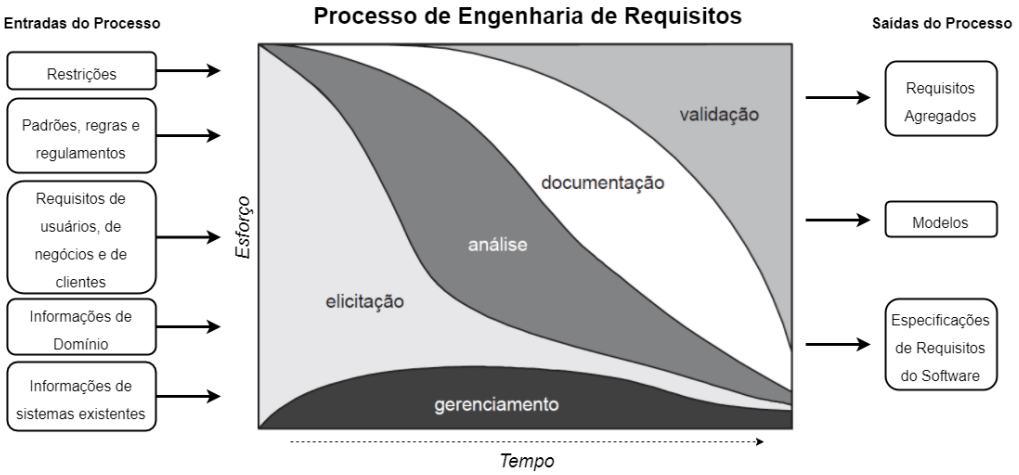


Figura 2.1: Processo Geral da Engenharia de Requisitos. Adaptado de [1]

necessidades que o *software* atenderá. Para Sayão, Lopes, Ramesh e Toranzo [6, 5, 2, 3], os requisitos de *software* são voláteis, instáveis e propícios a mudanças durante o processo de desenvolvimento de *software* e tais mudanças não devem ser vistas como impedimentos e devem ser atendidas. Contudo, essas mudanças devem ser devidamente controladas e registradas para se possa estimar custos, prazos e impactos no projeto.

Para auxiliar os projetos de *software* a garantir qualidade, não ultrapassar custos e prazos de entrega previamente estipulados juntamente aos clientes, a Engenharia de Requisitos, sub-área da Engenharia de *Software*, preocupa-se com a identificação dos requisitos, os envolvidos e suas necessidades, bem como com a análise, classificação, documentação, delimitação de comportamentos, validação, evolução e gerenciamento dos requisitos durante o processo de desenvolvimento de *software* [7, 12, 13, 1] de forma que os requisitos sejam previamente conhecidos e controlados durante todo o processo de desenvolvimento.

### 2.1.2 A Engenharia de Requisitos

O conhecimento correto dos requisitos evita que os *softwares* sejam desenvolvidos incorretamente, possuam funcionalidades inutilizáveis e/ou conflitantes e que atendam às reais necessidades e expectativas dos *stakeholders* [6]. A necessidade de desenvolvimento de um *software* tem como base a resolução de um dado problema do mundo real. Sendo assim, entender precisamente tal problema e como um *software* pode auxiliar é essencial para o sucesso do *software*. Os requisitos são elicitados inicialmente no projeto e conhecê-

los adequadamente é uma das condições básicas para as fases posteriores da construção do *software* [1].

Para Lopes [5], a Engenharia de Requisitos é uma disciplina relacionada ao estudo dos requisitos, sua especificação, documentação, validação e negociação entre todos os *stakeholders* de um sistema.

As atividades que contemplam a Engenharia de Requisitos são cinco: Elicitação, Análise e Negociação, Documentação, Validação e Gerenciamento [7, 5, 1]. Essas atividades fazem parte do processo da engenharia de requisitos (Figura 2.1) que compreende na aquisição de um conjunto de dados como entradas, o processamento desses dados com as cinco atividades listadas anteriormente e as saídas que são os requisitos analisados e agregados ao sistema. Essas atividades são detalhadas a seguir:

1. **Elicitação**: essa fase é caracterizada pela identificação dos requisitos do sistema com base em interações com os solicitantes do *software*. De acordo com Lopes [5], essa fase consiste na consulta dos *stakeholders*, análise da documentação, análise de informações do domínio e/ou de estudos de mercado;

2. **Análise e Negociação**: essa fase consiste na avaliação da consistência, conflitos e inconsistências entre requisitos com relação a necessidade dos clientes. Para Genvigir [1], essa atividade tem como objetivo analisar os detalhes dos requisitos e verificar se os requisitos possuem conflitos se atendem as regras de negócio do cliente;

3. **Documentação**: os requisitos aprovados na fase anterior e que serão codificados devem ser documentados de forma clara e que não traga ambiguidade. Genvigir [1] destaca que "...os requisitos devem ser documentados a fim de servir de base para o restante do processo de desenvolvimento. Essa documentação deve ser feita de forma consistente, seguindo-se um padrão que permita demonstrar, em vários níveis de detalhes, a especificação dos requisitos levantados";

4. **Validação**: após a fase de documentação, os requisitos são validados com critérios baseados em consistência, completude, corretude e viabilidade [7, 5, 1].

5. **Gerenciamento**: essa atividade tem como objetivo o controle dos requisitos já elicitados, analisados e validados com os usuários e esses novos requisitos elencados serão agregados ao sistema [1]. Essa atividade é uma das mais importantes dentro da Engenharia de Requisitos pois acompanha o estado do requisito durante todo ciclo de vida do *software* [6, 5, 1];

O gerenciamento dos requisitos acompanha os requisitos desde a sua concepção junto aos *stakeholders* à sua evolução. Esse gerenciamento é essencial para o sucesso do projeto já que mantém o controle dos requisitos durante todo ciclo de vida do *software*, auxiliando os *stakeholders* a prever impactos a custos, prazos e a qualidade do produto em questão.

### 2.1.3 O Gerenciamento dos Requisitos

O gerenciamento de requisitos objetiva o acompanhamento dos requisitos de um *software* em todas as fases de concepção do mesmo, desde a elicitação à evolução do *software* [7, 2]. Diversos autores e abordagens conceituam essa gerência e todos apontam a importância da mesma como fator determinante na qualidade do *software* entregue [2, 6, 5, 3, 1].

De acordo com Sommervile [7], o papel dessa gerência é o acompanhamento entre requisitos, relacionamentos entre requisitos, gerenciamento das dependências entre a documentação de requisitos e outros artefatos originados durante outros processos da engenharia de *software*.

Lengwell e Widrig, citados por [1], definem o gerenciamento de requisitos como um esforço sistemático para elicitar, organizar, e documentar um processo de engenharia de requisitos afim de estabelecer acordo entre clientes, usuários e o grupo de desenvolvimento no que tange às mudanças dos requisitos de um sistema.

A opinião dos autores convergem em um ponto em comum: o gerenciamento de requisitos pode se tornar uma tarefa difícil e árdua. Os requisitos de *software* mudam constantemente e a documentação e atualização dos mesmos se torna custosa [6, 5]. Atualmente, tem-se a convicção que mudanças em requisitos ao longo do processo de desenvolvimento de *software* fazem parte do processo [6]. Lopes [5] afirma que os requisitos possuem uma natureza volátil e são instáveis, fatores como mudanças na legislação, adaptações no requisito, atualizações tecnológicas, mudanças no mercado, mudanças na política da empresa e correção de requisitos entendidos incorretamente fazem com que os requisitos sejam modificados e possam afetar outros requisitos e artefatos.

Para que o gerenciamento de requisitos seja efetivo e traga benefícios ao projeto, ele deve ser implementado desde o início do projeto do *software*. Segundo Lopes [5], os benefícios dessa atividade não são visualizados imediatamente e são "percebidos no médio prazo, sendo que são necessários investimentos no curto prazo". Lopes [5] ainda destaca que a não implementação do gerenciamento de requisitos pode proporcionar economias de curto prazo mas que geram impactos em custos e prazos posteriores ao projeto. Visando garantir a qualidade e melhoria de processos de *software*, diversos

guias foram desenvolvidos dentre eles o guia MPS.BR [8] e o guia AADSP [10].

### 2.1.4 As Atividades do Gerenciamento dos Requisitos

O processo de Gerenciamento de Requisitos ocorre em paralelo com as fases da Engenharia de Requisitos elencadas anteriormente [1]. As atividades do Gerenciamento por Requisitos incluem identificação dos requisitos e suas ramificações, controle de mudanças dos requisitos e a rastreabilidade dos requisitos.

1. **Identificação**: Essa atividade tem como objetivo identificar e registrar os requisitos. É essencial que para cada projeto os requisitos tenham um identificador único. Essa identificação única permite que o requisito seja referenciado, versionado e gerenciável [7].

2. **Controle dos Requisitos**: consiste no acompanhamento de entrada de novos requisitos a serem integrados ao sistema bem como a alteração de requisitos já existentes [6, 7]. Os responsáveis pelo desenvolvimento do *software* conseguem analisar o impacto da inclusão ou alteração de requisitos no sistema e estimar possíveis aumentos ou redução em variáveis como custo, tempo e qualidade do *software* [1].

3. **Rastreabilidade**: Atividade mais importante do Gerenciamento de Requisitos, a rastreabilidade proporciona aos envolvidos responsáveis pela concepção do *software* um acompanhamento e mapeamento de todos os elos entre requisitos bem como dos produtos gerados a partir de um requisito ou um conjunto de requisitos [2, 6, 7].

A atividade de Rastreabilidade, destacada na Figura 2.2, é a atividade central do gerenciamento de requisitos [1] e tal atividade é essencial para a construção de *softwares* com qualidade, pois os *stakeholders* conseguem visualizar quais requisitos já estão implementados, se há conflitos entre requisitos, os graus de dependência entre requisitos e quais impactos podem ser gerados com a modificação de um ou mais requisitos.

Por conta da sua importância, a rastreabilidade de requisitos será melhor discutida nas próximas subseções. Serão apresentados os principais conceitos, classificações, impactos positivos e negativos da aplicação da rastreabilidade em um projeto de *software* bem como os metamodelos para rastreabilidade de requisitos propostos por alguns autores.

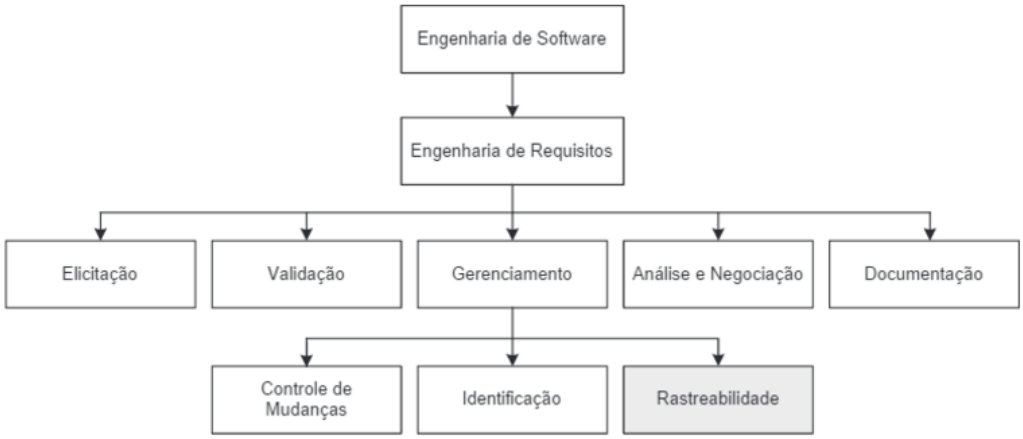


Figura 2.2: Área da Rastreabilidade dentro da Engenharia de Software. Extraído de [1]

## 2.2 Rastreabilidade

### 2.2.1 Conceituando a Rastreabilidade

A rastreabilidade de requisitos é utilizada para localizar todas as dependências e relacionamentos que deram origem ou foram originados a partir dos requisitos durante todo o ciclo de vida do *software* [2, 3, 6]. Os requisitos podem ser encontrados antes da sua implementação no *software* em atas de reuniões, entrevistas e outros métodos para elicitação de requisitos e posteriormente em artefatos documentais, componentes e código-fonte do *software*.

De acordo com o guia MPS.BR [8], a rastreabilidade é o relacionamento que pode ser estabelecido entre dois ou mais produtos do *software*, principalmente por produtos que tenham um vínculo do tipo mestre e subordinado. Além disso, o guia destaca que a rastreabilidade de requisitos é fundamental para apoiar o processo de mudanças de requisitos.

Segundo Sayão [6], a rastreabilidade de requisitos é utilizada para promover o relacionamento entre os requisitos, a arquitetura do *software* e implementação final do sistema, permitir uma melhor compreensão dos relacionamentos existentes entre os requisitos do *software* e negociar novos prazos e custos a partir de mudanças com cliente. Sua implementação pode ser feita através de elos (ligações) entre os requisitos, dos requisitos a suas fontes e dos requisitos a sua implementação [6].

Em sua dissertação, Genvigir[1] diz que a rastreabilidade está diretamente associada ao processo de construção do *software* com a capacidade de estabelecer vínculos entre requisitos e outros artefatos de *softwares* como modelos,

documentos, código-fonte e testes. Outras características da rastreabilidade ressaltadas por Genvigir [1] são: assistir o processo de verificação dos requisitos em um sistema, medir os impactos que a mudança em um requisito pode trazer ao projeto, compreender aspectos do projeto bem como a evolução de um componente do sistema e as motivações que levaram a modificações nos requisitos.

Para melhor compreender a rastreabilidade é necessário classifica-lá. A classificação da rastreabilidade dá-se com base nas informações em que desejase rastrear e o sentido/direção da rastreabilidade [6]. As informações que podem ser rastreadas são os requisitos, artefatos, vínculos entre os requisitos e entre requisitos e artefatos, origem dos requisitos (*stakeholders* ou documentos). Já a direção da rastreabilidade está diretamente ligada as fases do desenvolvimento de *software* em que um requisito está presente e as ligações entre requisitos e artefatos.

### 2.2.2 Classificação da Rastreabilidade

A rastreabilidade de requisitos pode ser classificada em dois tipos: quanto às fases que antecedem e sucedem a especificação dos requisitos representadas pela pré-rastreabilidade e pós-rastreabilidade [6] e quanto ao seu sentido/direção, representada pelas rastreabilidade horizontal (RH) e rastreabilidade vertical (RV) [8, 6].

#### Pré e Pós Rastreabilidade

A pré-rastreabilidade e pós-rastreabilidade, conforme ilustradas na Figura 2.3, estão relacionadas ao ciclo de vida do requisitos antes e depois da fase de especificação dos requisitos [1, 5]. A pré-rastreabilidade registra o contexto em que os requisitos surgiram (reuniões, atas, *workshops*, entrevistas, questionários e outros), já a pós-rastreabilidade relaciona os requisitos aos artefatos do sistema e código-fonte [6, 1].

De acordo com Genvigir [1], a principal diferença entre as duas são as informações que ambas são responsáveis. A pós rastreia um requisito a partir de um ponto de referência (especificação dos requisitos) aos artefatos oriundos dos requisitos contidos na especificação, enquanto a pré cria um rastreamento entre as fontes dos requisitos (cliente, usuários, normas e padrões) aos artefatos gerados pelas primeiras fases da Engenharia de Requisito como a Elicitação, Análise, Validação e Documentação dos Requisitos [5].

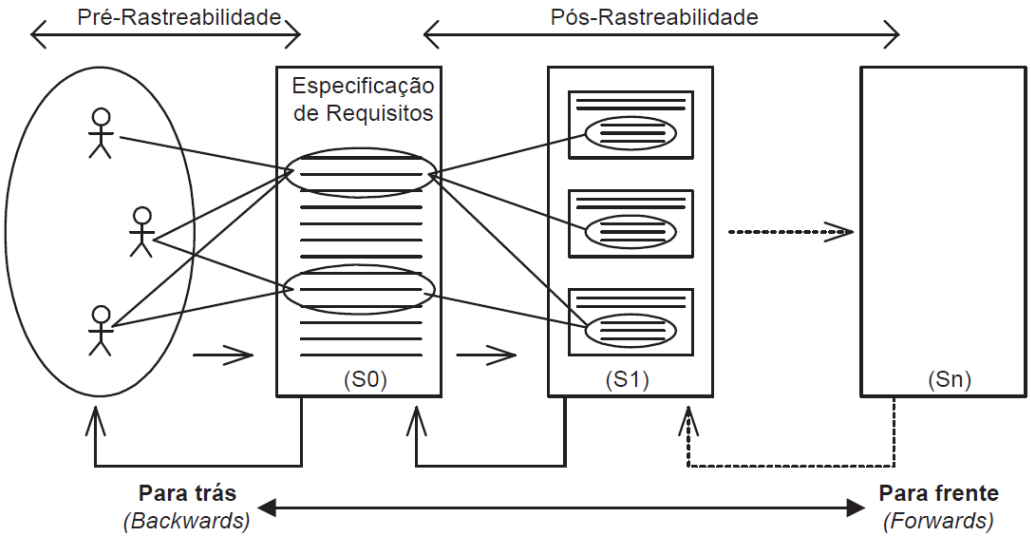


Figura 2.3: Pré e Pós Rastreabilidade. Extraído de [1]

### Rastreabilidade Horizontal e Vertical

A aplicação da Ratreabilidade Horizontal (RH) e da Rastreabilidade Vertical (RV) indexrastreabilidade horizontal e vertical, conforme apresentadas na Figura 2.4, auxilia a determinar se os requisitos foram implementados e se os mesmos possuem uma fonte válida, além de fornecer ao responsável pelo gerenciamento do projeto informações precisas para negociar com o cliente mudanças no plano do projeto para atender a mudanças no requisitos de forma que desvios no cronograma e em custo sejam previamente estimados [8, 6].

A RH, também conhecida como inter-rastreabilidade, é a forma de rastreabilidade entre diferentes versões do requisito e permite uma visualização de como os requisitos relacionam-se com outros requisitos. É também utilizada para obter conhecimento dos versionamentos de um requisito e verificar as dependências de um requisito em um mesmo nível ou fase do ciclo de vida do produto [1, 8]

Já a RV, também conhecida como extra-rastreabilidade, é realizada entre requisitos e artefatos produzidos pelo processo de desenvolvimento ao longo do ciclo de vida do projeto, ou seja, permite ver como os requisitos relacionam-se com os artefatos que são gerados no decorrer do projeto [1]. Essa forma de rastreabilidade proporciona o conhecimento de todos os refinamentos dos requisitos que foram produzidos ao longo do ciclo de vida do projeto [1]. De acordo com o guia MPS.BR [8], a RV estabelece um rastreio desde um requisito fonte aos níveis mais baixos de decomposição do produto

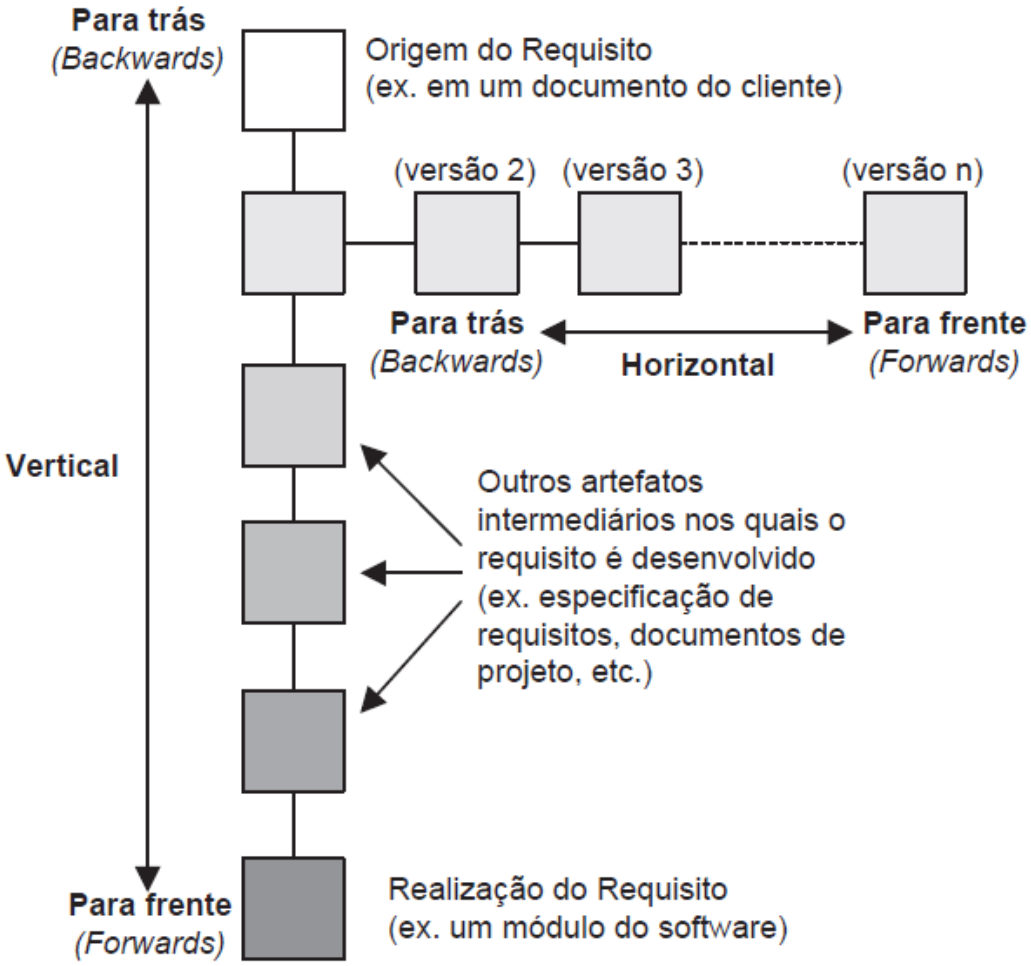


Figura 2.4: Rastreabilidade Vertical e Horizontal. Extraído de [1]

como código-fonte e testes unitários e vice-versa.

### 2.2.3 A Aplicação da Rastreabilidade de Requisitos

Como citado anteriormente, os requisitos de um *software* possuem uma natureza volátil e podem ser modificados durante o ciclo de vida de *software* por diversos fatores, além disso, os responsáveis pela criação do *software* tem interesse em saber o estado atual dos requisitos e encontrar artefatos, estruturas, documentos e componentes que estão vinculados aos requisitos [8, 5]. Nesse contexto, a rastreabilidade de requisitos pode auxiliar no processo da gerência de projetos bem como no processo de desenvolvimento de *software*, contudo, a manutenção da rastreabilidade e a atualização da documentação

vinculada a ela pode ser custosa em alguns cenários de desenvolvimento de *software*.

Conforme destacado pelo guia MPS.BR [8], a rastreabilidade tem o papel de determinar se todos os requisitos previstos para um *software* foram implementados e, além disso, permite que o gerente de projetos negocie com os clientes as mudanças nos requisitos de forma que os riscos no projeto, custo, cronograma e qualidade não sejam comprometidos.

Diversos autores destacam os benefícios da rastreabilidade tais como:

1. Estimar variações em cronogramas e alterações de custos no desenvolvimento [6];
2. Verificar a alocação de requisitos a componentes de *software* [6];
3. Compreender o relacionamento entre requisitos, requisitos e artefatos, projeto e implementação [1];
4. Suportar a verificação de requisitos de um sistema [1]
5. Validar o *software* junto ao cliente [6];
6. Resolver de conflitos entre requisitos [6];
7. Compreender a evolução de um artefato [1];
8. Compreender aspectos do projeto [1];
9. Entendimento das razões por trás das decisões de projeto (*Design Rationale*) [1];
10. Analisar o impacto na evolução do sistema [6];
11. Reusar componentes [6];
12. Realizar a cobertura de testes [6];
13. Corrigir defeitos [6];
14. Garantir uma contínua concordância entre os requisitos dos interessados no sistema [1];

Contudo, em sua dissertação, Sayão [6] destaca que a pouca flexibilidade de ferramentas disponíveis no mercado obriga os desenvolvedores a catalogar a rastreabilidade manualmente o que pode levar ao desuso da mesma.

Para Lopes [5], a rastreabilidade é um trabalho extenso, caro e vagaroso e que o principal problema da rastreabilidade está vinculado a grande quantidade informações que ela pode gerar. E, assim como Sayão [6], ele enfatiza a importância do uso ferramentas para catalogar e associar elementos relacionados ao requisitos. Além disso, Lopes [5] destaca que o comprometimento da equipe técnica com atividade de rastreabilidade é fundamental para manter a rastreabilidade dos requisitos do *software* e que jamais deve ser vista como um processo impeditivo pois resulta em benefícios a longo prazo.

Em suas pesquisas, Ramesh [2] relata que os problemas relacionados a rastreabilidade estão nas maneiras como ela é aplicada, o mau uso de ferramentas e na escolha dos requisitos a serem rastreados. Devido a grande quantidade de informação que a rastreabilidade pode gerar [5, 6], os requisitos a serem rastreados devem ser escolhidos de forma que não comprometam prazos e custos do projeto. Além disso, Sayão [6] afirma que a prática incorreta ou incompleta da rastreabilidade na pré-rastreabilidade pode causar impactos posteriores ao projeto.

Com relação ao uso da rastreabilidade por parte de usuários (desenvolvedores), Ramesh [2] conduziu uma pesquisa sobre as influências da organização na rastreabilidade e classificou os usuários de rastreabilidade em dois grupos, usuários avançados e mais sofisticados denominados de *high-end* e os usuários comuns e básicos denominados *low-end*.

O primeiro grupo, *high-end*, utiliza técnicas de rastreabilidade mais avançadas e veem a rastreabilidade como uma atividade fundamental para melhoria no processo de *software* e que trará benefício a longo prazo à todos envolvidos no projeto. Usuários desse primeiro grupo utilizam esquemas de rastreabilidade mais ricos, registram as motivações por trás das mudanças nos requisitos e se preocupam com a evolução do *software*. Já o segundo grupo, *low-end*, utiliza técnicas de rastreabilidade mais simples, encaram a rastreabilidade como uma imposição do gerente do projeto e registram de forma incompleta ou insuficiente a motivação por trás da criação ou mudança nos requisitos [2].

Apesar de apresentar uma implementação cara e vagarosa [5], a rastreabilidade pode trazer ótimos benefícios ao *software* produzido, à empresa e aos clientes, uma vez ela sendo bem aplicada. Escolher adequadamente quais requisitos devem ser rastreados e difundir a prática de rastreabilidade entre os envolvidos pode facilitar o gerenciamento dos requisitos [6]. Lopes [5] cita que a rastreabilidade é um fator cultural e que cada empresa deve incentivar que a equipe técnica empenhe-se na implementação da atividade vislumbrando benefícios futuros.

Tendo em vista os problemas da rastreabilidade, diversos autores desenvolveram metamodelos para auxiliar na atividade. Alguns metamodelos clas-

sificam as atividades processuais, informações a serem rastreadas e os tipos de vínculos entre requisitos e artefatos.

### 2.2.4 Metamodelos para Rastreabilidade de Requisitos

De acordo com Genvigir [1], vários autores fazem o uso de modelos com base nas informações de um determinado domínio que desejam representar de forma gráfica ou textual. Para a rastreabilidade dos requisitos, os modelos são utilizados para representar os diferentes tipos de elos que podem ocorrer entre requisitos e artefatos de *software* [2]. Para Sayão [6], esses elos possibilitam a identificação da origem de cada funcionalidade presente no sistema de acordo com a necessidade dos clientes, verificação e validação da alocação dos requisitos solicitados no *software* desenvolvido e conformidade dos testes com os requisitos.

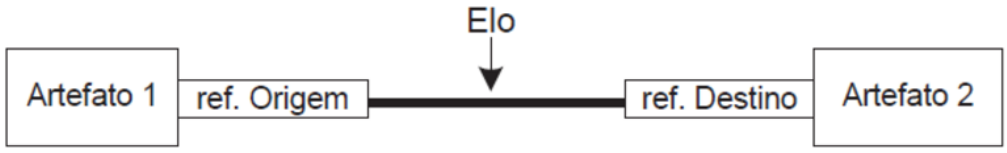


Figura 2.5: Artefatos conectados através de um elo. Extraído de [1]

Conforme ilustrado na Figura 2.5, o elo é o estabelecimento de um relacionamento direto entre um artefato de origem e um artefato de destino [1] e eles são os principais recursos para manter e representar os relacionamentos da rastreabilidade além de serem utilizados em diversas áreas da engenharia de *software*, tais como: a engenharia de requisitos nas atividades de validação, análise, evolução e referência cruzada entre requisitos e artefatos, na gerência de projetos com a análise de custos, restrições e impactos, na evolução do *software*, nos testes e na gestão de qualidade [6].

#### Metamodelo proposto por Ramesh

Ramesh [2] desenvolveu um metamodelo de rastreabilidade que pode ser visualizado na Figura 2.6 baseado em uma longa pesquisa de forma que esse modelo pudesse representar a rastreabilidade de forma generalista. De acordo com Sayão [6], o metamodelo proposto por Ramesh [2] permite a captura de informações relacionadas a fontes (*source*), interessados ou envolvidos (*stakeholders*), objetos ou artefatos (*objects*) [2, 1].

As fontes podem ser classificadas como os documentos que remetem a origem dos requisitos tais como normas, padrões, editais, atas de reunião,

regras institucionais, leis e medidas governamentais. Já os interessados podem ser os solicitantes do *software* que propõem melhorias e novas mudanças bem como os engenheiros e desenvolvedores interessados na rastreabilidade dos requisitos. Por fim, os objetos são todos os artefatos gerados durante o processo de desenvolvimento, tais como: código-fonte, arquitetura, casos de testes, documentos de requisitos e outros [2, 1, 6].

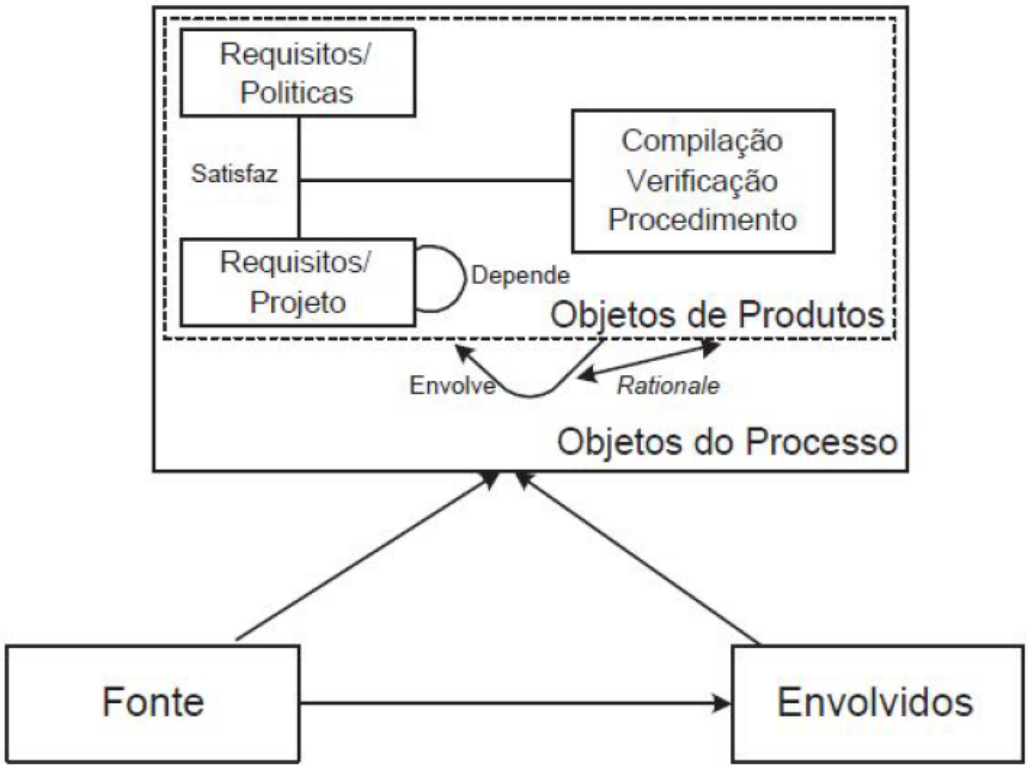


Figura 2.6: Metamodelo proposto por [2] adaptado por [1].

Em sua pesquisa, Ramesh [2] destacou que mesmo havendo uma grande quantidade de elos de rastreabilidade, eles podem ser agrupados em dois grupos básicos: produtos e processo. Os elos do primeiro grupo são divididos em satisfação e dependência e descrevem as propriedades e os relacionamentos entre os objetos [1]. Já o segundo grupo que é dividido em elos de evolução e *rationale* tem o objetivo de registrar o histórico de ações e mudanças no processo [6].

### Metamodelo proposto por Toranzo

Em sua proposta para melhoria da rastreabilidade entre requisitos, Toranzo [3] propõe a classificação das informações a serem rastreadas em um meta-

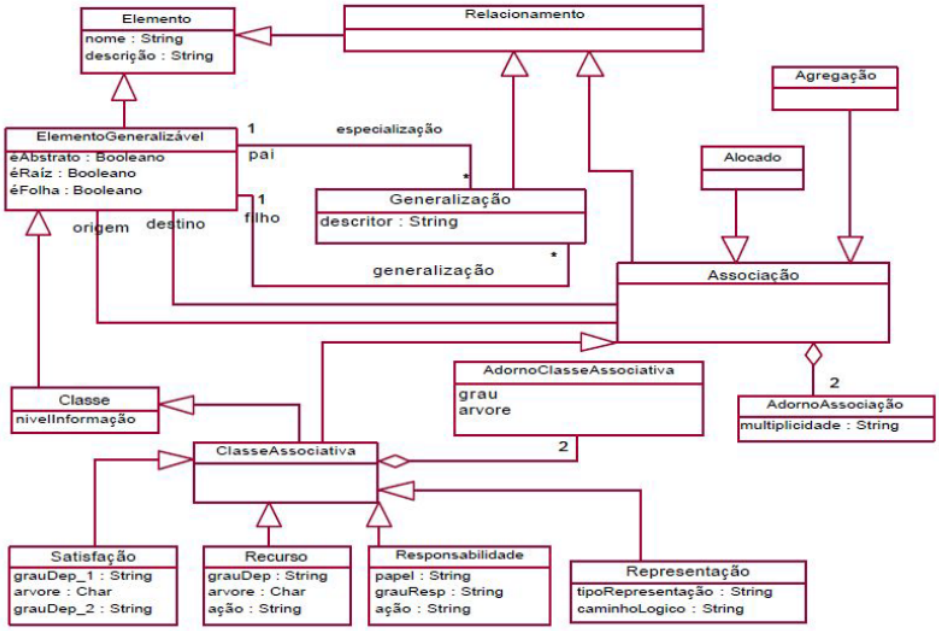


Figura 2.7: Metamodelo proposto por Toranzo. Extraído de [3]

modelo, que pode ser visualizado na Figura 2.7 para auxiliar no rastreamento do requisito, além de um metamodelo intermediário e um processo de rastreabilidade que não serão abordados neste livro.

Através deste metamodelo, Toranzo [3] propõe seis tipos de elos: satisfação, recurso, responsabilidade, representação, alocação e agregação.

1. **Satisfação**: indica que a classe de origem tem dependência de satisfação com a classe de destino [3];

2. **Recurso**: indica que uma classe de origem têm dependência de recurso com a classe destino [6, 1]. Para Toranzo [3], recurso é um elemento que representa uma informação tais como motivações por trás de mudanças em requisitos;

3. **Responsabilidade**: representa a participação, os responsáveis e os conjuntos de ações das pessoas sobre os artefatos de *software* gerados [6, 1, 3];

4. **Representação**: captura a representação dos requisitos em linguagens sejam elas de programação ou modelagem de dados;

5. **Alocação**: similar ao elo de satisfação, esse elo representa a dependência de uma classe de origem com uma classe de destino que representa

um subsistema [6];

6. **Agregação**: visualiza se um elemento é composto por outros elementos [3];

Outra contribuição importante na proposta de Toranzo [3] é a classificação do conjunto de informações que devem ser rastreadas para melhor entender a atividade de rastreamento dos requisitos. Ele propôs a classificação dessas informações em quatro níveis: ambiental, organizacional, gerencial e de desenvolvimento.

As informações do nível ambiental podem contemplar as informações de onde o *software* será desenvolvido, o contexto organizacional da empresa [6], os contextos econômicos, políticos e os padrões (externos) [3]. O nível organizacional agrega informações como missão, objetivos, metas, técnicas internas e padrões (internos) [6]. As informações do nível gerencial relacionam os requisitos com as tarefas realizadas para atender os mesmos e um acompanhamento e controle dos requisitos [3]. E o último nível, de desenvolvimento, representa os artefatos tais como documentos de requisitos, arquitetura, diagramas, código-fonte, casos de teste e outros que são produzidos durante a construção do *software*.

### Metamodelo proposto por Genvigir

Para solucionar os problemas acima citados, Genvigir [1] propõe um metamodelo que pode ser visualizado na Figura 2.8 baseado na generalização de todos os tipos de artefatos e elos que possam participar do processo de rastreabilidade. Seu modelo busca possibilitar a definição dos diferentes tipos de artefatos que podem ser gerados durante o desenvolvimento do *software* com a inclusão de atributos aos elos.

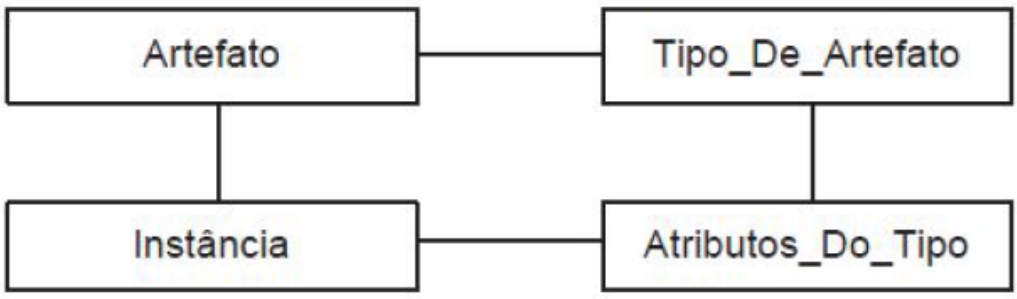


Figura 2.8: Metamodelo proposto por Genvigir. Extraído de [1]

1. **Tipo de Artefato**: Classificação de um grupo de artefatos que possuem mesmas características tais como casos de uso, arquitetura, classes, módulos e outros;

2. **Atributos do Tipo**: Para cada tipo de artefato, podem ser criados diversos atributos que tem o objetivo de detalhar seu comportamento ou suas propriedades;

3. **Artefato**: Reflete o artefato criado para o *software* e que será rastreado;

4. **Instâncias**: São os valores reais atribuídos para um artefato;

Um aspecto similar entre o modelo proposto por Genvigir e Ramesh [1, 2] é a simplicidade e o fácil entendimento do modelo. Contudo, o modelo de Gengivir [1] é generalista e não define os elos, permitindo que os usuários da rastreabilidade possam adaptar o modelo ao seu processo de desenvolvimento, podendo criar artefatos e elos durante todas as fases do projeto. Além disso,, tal modelo possibilita a adição de novos atributos a um tipo de artefato durante qualquer fase do projeto.

Para a criação de elos entre os requisitos e artefatos do sistema, Genvigir [1] propõe a criação de um tipo de artefato denominado elo que possui três atributos: código da origem, identificador e código de destino. Os códigos de origem e destino armazenam os identificadores de requisitos e/ou artefatos que deseja-se criar um rastro. E o atributo identificador serve para classificar o tipo do elo criado.

### Analise dos Metamodelos propostos por Ramesh, Toranzo e Genvigir

Sayão e Genvigir [6, 1] realizaram uma breve análise dos modelos de Ramesh e Toranzo [2, 3] citados anteriormente e concluíram que em certos momentos os metamodelos apresentam aspectos similares e contribuições diferentes para a literatura.

De acordo com Sayão [6], os elos de satisfação criados por Ramesh [2] podem ser equiparados aos elos de alocação criados por Toranzo [3]. Ambos tem o propósito de indicar que o requisito está alocado em um sistema. Outra similaridade entre os trabalhos encontrada por Sayão [6] está nos elos de dependência. Os elos de dependência criados por Ramesh [2] podem ser equiparados aos elos de recurso, satisfação ou os de agregação criados por Toranzo [3]. Todos esses elos tem como objetivo mostrar um relacionamento

direto ou indireto entre recursos, classes/componentes e elementos do *software*.

Além disso, Sayão [6] destaca que a maior contribuição da proposta de Ramesh [2] é a simplicidade do modelo e a criação de elos voltados para a fase de evolução dos requisitos de forma que possam ser registradas as alterações do *software* e as motivações para tais mudanças (*Design Rationale*). Outra contribuição do trabalho de Toranzo [3] é a atenção empenhada no aspectos gerenciais do projeto com a criação de elos de responsabilidade e classificação do ambiente, já que o gerenciamento dos requisitos é uma atividade interligada a gerência do projeto e que pode auxiliar na redução dos impactos negativos aos custos, cronogramas e otimização das atividades.

No ponto de vista de Genvigir e Lopes [1, 5], Ramesh [2] realizou uma importante classificação dos usuários de rastreabilidade em *high-end* e *low-end users*, citados anteriormente neste artigo, e concordam com [6] no tocante a importância do aspecto gerencial abordado no trabalho de [3] com a criação do elo de responsabilidade para rastrear os *stakeholders* do projeto. Porém Genvigir [1] cita alguns pontos negativos em ambos trabalhos:

1. A predefinição dos elos que devem ser rastreados através de padrões predefinidos em grupos de elos;

2. Representação dos elos em matrizes de rastreabilidade torna-se incompleto uma vez que deve ser criada uma matriz para cada tipo de elo;

3. Impossibilidade de adicionar atributos aos elos para enriquecimento da rastreabilidade;

O modelo proposto por Genvigir [1] apresenta uma alta flexibilidade para a criação de novos tipos elos e de artefatos de acordo com a necessidade da empresa diferente dos modelos de Ramesh [2] e Toranzo [3] no qual os elos são predefinidos. O autor justifica que essa generalização dos elos e artefatos se dá pela agregação de novos campos para enriquecer os dados de um elo ou de um artefato e a alocação de novos tipos de artefatos ao projeto.

Um dos problemas encontrados no modelo proposto por Genvigir [1] está relacionado a falta de flexibilidade para criação de atributos apenas para os artefatos, sendo possível apenas a inclusão de atributos aos tipos dos artefatos. Consequentemente, todo artefato criado poderá possuir todos os atributos vinculados ao tipo. Como, por exemplo, um artefato é criado e a ele é atribuído o Tipo “Requisito”. Caso esse Tipo possua um atributo chamado Versionamento, todos os artefatos criados a partir desse tipo poderão ter instâncias do atributo Versionamento sendo que nem todos os Requisitos de um *software* são versionáveis.

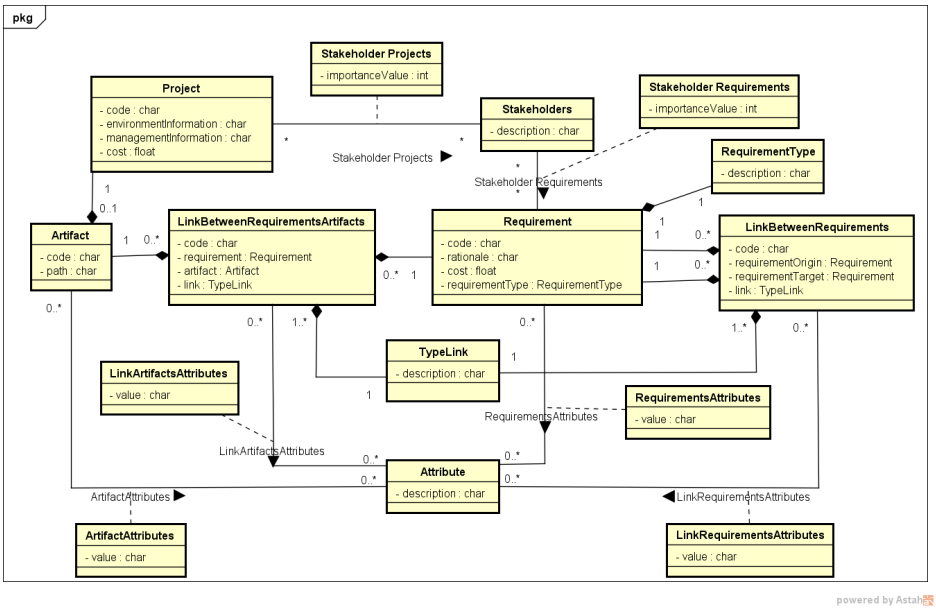


Figura 2.9: Metamodelo proposto pela ferramenta T-AADSP Requirements

Outro problema está na possibilidade de geração de inconsistências em bases de dados caso o modelo proposto por Genvigir [1] seja aplicado. De acordo com o modelo proposto pelo autor e por exemplos em sua dissertação, o elo entre requisitos e artefatos pode ser mantido através de um Tipo de Artefato denominado "Elo" que possui dois Atributos do Tipo denominados "Código de Origem" e "Código de Destino". A Instância desses Atributos do Tipo recebem no campo "Valor" a chave primária da tabela Artefatos (que nesse contexto está representando um requisito e um artefato) criando assim o elo de rastreabilidade. Contudo, o campo Valor não é uma *Foreign Key* da tabela Artefato, possibilitando assim que um artefato possa ser excluído e o banco não aplique regras básicas de integridade de dados providas por Softwares de Gerenciamento de Base de Dados (SGBDs).

### 2.2.5 Metamodelo proposto para a ferramenta T-AADSP Requirements

O metamodelo proposto, conforme Figura 2.9, baseia-se nas principais contribuições dos metamodelos acima citados e em conceitos do Gerenciamento de Requisitos apresentados nas seções anteriores.

1. **Project**: Representa um projeto de *software* que possui as informações ambientais e organizacionais para auxiliar no desenvolvimento do

projeto [3]. Os projetos devem possuir um custo que será particionado entre os requisitos do *software*;

2. **Requirement**: Representa um requisito do *software* que será passível de gerenciamento. Cada requisito deve possuir uma descrição do *rationale* por trás da criação do mesmo [6, 7, 2]. O custo de cada requisito deve estar dentro do valor estipulado pelo projeto;

3. **LinkBetweenRequirements**: Representa um elo entre dois requisitos. Esse elo deve possuir um tipo de elo representado pela classe ***TypeLink***.

4. **TypeLink**: Representa o conjunto de elos disponíveis e que foram elencados no trabalho de Lopes [5];

5. **RequirementType**: Classifica os tipos de requisitos (funcionais, não funcionais e outros);

6. **Stakeholders**: Classifica os *stakeholders* do projeto que podem estar vinculados a projetos e a requisitos;

7. **StakeholderProjects**: Adiciona um vínculo entre um *Stakeholder* e um Projeto. Cada *stakeholder* tem um valor de importância para o projeto que pode variar de 1 a 9;

8. **StakeholderRequirements**: Adiciona um vínculo entre um *Stakeholder* e um Requisito. Cada *stakeholder* tem um valor de importância para o requisito que pode variar de 0 a 9;

9. **Artifact**: Representa um artefato do *software* que possui um identificador único. O diretório lógico do artefato deve ser informado para que o mesmo possa ser rastreado.

10. **LinkBetweenRequirementArtifacts**: Representa um elo entre um requisito e um artefato do *software*. Assim como o elo entre requisitos, este deve possuir um tipo de elo representado pela classe ***TypeLink***;

11. **Attribute**: Representa um atributo que poderá ser utilizado pelas classes que a agregarem. Essa classe foi criada baseada no conceito de atributos para entidades proposto por [1];

As classes *Project, Requirement, LinkBetweenRequirements, Artifacts* e *LinkBetweenRequirementArtifacts* devem possuem um identificador único representados através do atributo de classe *code* para que aspectos como rastreabilidade, reuso de código, não ambiguidade e outros possam ser alcançados.

*Requirement, LinkBetweenRequirements, Artifacts* e *LinkBetweenRequirementArtifacts* podem possuir um conjunto de atributos específicos que ajudam no enriquecimento da rastreabilidade de informações [1]. Esses atributos são representados pelo campo *value* pertencente as classes-associativas que são geradas a partir da relação entre as quatro classes acima citadas com a classe *Attribute*.

## 2.3 The Next Release Problem

Um projeto de *software* pode ter a participação de diversos *stakeholders* de diferentes setores e/ou áreas de conhecimento com diferentes níveis de interesse sobre o *software* em questão [6, 7]. Cada um desses *stakeholders* possui maior ou menor grau de importância para o bom desempenho e sucesso do projeto, critério esse determinado pela organização solicitante do *software* [5, 4]. Tais *stakeholders* podem solicitar funcionalidades à serem desenvolvidas para atender suas expectativas e necessidades do mundo real, logo, os requisitos de um projeto de *software* possuem diferentes níveis de interesse para cada *stakeholder* do projeto. Além disso, os requisitos solicitados pelos *stakeholder* do projeto devem possuir um custo financeiro que esteja dentro do orçamento total disponível para o projeto [7, 2].

Um dos problemas computacionais vinculados aos requisitos de *software* está atrelado a seleção de requisitos que devem ser implementados em um período de desenvolvimento de forma que a escolha desses atenda as necessidades dos clientes e ao atual custo do projeto [4]. Objetivando resolver esse impasse, o *The Next Release Problem* (NRP), busca estabelecer um balanceamento na seleção dos requisitos que serão contemplados em uma *release*[1] levando em consideração diversos fatores como custo, prazo, esforços, riscos, satisfação dos clientes e interdependência entre os requisitos através da aplicação de algoritmos genéticos multiobjetivos [14].

O **NRP** é um dos problemas abordados pela área de Otimização em Engenharia de Software, também conhecida como *Search-Based Software Engineering*, que tem como objetivo aplicar um conjunto de técnicas de seleção e otimização para problemas recorrentes da Engenharia de *Software* [4].

Para a aplicação dos algoritmos que objetivam a resolução do problema *Next Release Problem*, um conjunto de entradas parametrizadas e preestabelecidos deve ser disponibilizadas. Essas entradas seguem os critérios abaixo:

1. **Requisitos**: representam o conjunto de funcionalidades que devem

[1]Termo em inglês que se refere a uma liberação de *software*, para uso tal como uma nova versão oficial do mesmo

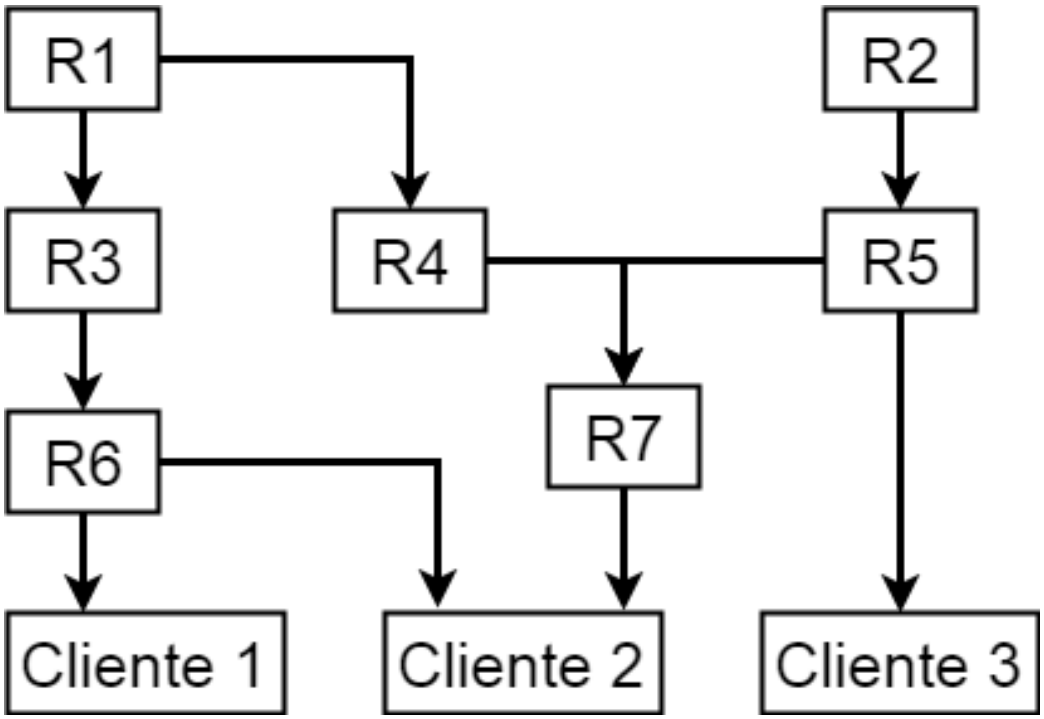


Figura 2.10: Exemplo do NRP. Adaptado de [4]

ser atendidas pelo sistema em questão e que podem ser alocadas em uma nova *release*. O desenvolvimento do requisito exige a estimativa de um custo operacional. Para atender essa condição, esse trabalho considerou a criação de um atributo na classe *Requirement* denominado *cost*, conforme ilustrado na Figura 2.9 que representa o valor financeiro do requisito [4, 14];

2. **Clientes**: representam o conjunto de pessoas que estão interessadas no sucesso do projeto. Cada cliente possui um **grau de importância** para a organização ou para o projeto que deve variar de 1 (menor importância) à 10 (maior importância). Para atender essa condição, esse projeto representa os **clientes** como ***stakeholders*** e o grau de importância deles é definido na classe ***StakeholderProject*** com o atributo ***importanceValue*** conforme ilustrado na Figura 2.9 [4, 14];

3. **Clientes** *versus* **Requisitos**: Os clientes de uma organização ou projeto possuem diferentes interesses para cada requisito. Cada cliente pode atribuir uma importância para um requisito no qual ele possua interesse, sendo que, esse grau de importância, deve variar entre 0 (o cliente não tem interesse no requisito) à 9 (alta importância). Para

atender essa condição, esse trabalho modelou a classe ***Stakeholder-Requirement*** com o atributo ***importanceValue*** conforme ilustrado na figura 2.9 [4, 14];

4. **Dependência e/ou Precedência entre Requisitos**: A criação dos elos entre requisitos permite que os clientes possam estabelecer quais requisitos são prioritários para o desenvolvimento bem como a dependência entre requisitos. Ao estabelecer quais requisitos devem ter prioridade em uma possível entrega, a satisfação do cliente é garantida. Para atender essa condição o trabalho propõe a criação de uma classe chamada *LinkBetweenRequirements* que possui como atributos o código do requisito de origem, o código de requisito de destino e o tipo do elo entre eles que pode ser de dependência e/ou precedência [4, 14];

A **SBSE** é uma das áreas de pesquisa da Engenharia de *Software* que está em crescimento e que apresenta algumas carências tais como a falta de dados parametrizados extraídos de fontes reais para aplicação de algoritmos para resolução de problemas, como por exemplo o já citado **NRP**, que é dos problemas do gerenciamento de requisitos. O ideal seria que as ferramentas de gerenciamento de requisitos disponibilizassem *datasets* para auxiliar no crescimento dessa área.

## 2.4 Modelos de Maturidade e Requisitos

### 2.4.1 Guia MPS.BR

A Associação para Promoção da Excelência do *Software* Brasileiro também conhecida como SOFTEX, com o apoio do Ministério da Ciência, Tecnologia e Inovação (MCTI), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) e Banco Interamericano de Desenvolvimento (BID/FUMIN), coordena o Programa MPS.BR. Criado em 2003, é um programa mobilizador e de longo prazo que tem como principal objetivo o aumento da competitividade das organizações através da melhoria nos processos de *softwares* e que seus modelos MPS sejam adequados a diferentes empresas e sejam compatíveis com padrões de qualidade aceitos internacionalmente. Para isso é necessário a implantação e contínuo aprimoramento dos Guias que contém uma descrição geral do modelo de referência com todas as definições necessárias para entendimento e aplicação [8].

O MPS.BR possui três modelos de referência: MPS-SW voltado para Melhoria de Processos de Software, MPS-SV voltado para Melhoria de Processo

de Serviços e MPS-RH voltado para Melhoria de Processo de Recursos Humanos. Segundo MPS.BR [8], o modelo MPS para *software* (MR-MPS-SW) tem como base os requisitos de processos definidos nos modelos de melhoria de processo e atende a necessidade de implantar os princípios de engenharia de *software* de forma adequada ao contexto das empresas, estando em conformidade com as principais abordagens internacionais para definição, avaliação e melhoria de processos de *software* [8].

O guia geral do MPS.BR divide os processos de melhoria em sete níveis de maturidade, sendo eles:

1. Nível G - Parcialmente Gerenciado;
2. Nível F - Gerenciado;
3. Nível E - Parcialmente definido;
4. Nível D - Largamente definido;
5. Nível C - Definido;
6. Nível B - Gerenciado quantitativamente;
7. Nível A - Em otimização;

Cada nível possui um guia específico para sua implementação e um conjunto de gerências que deve ser atendido para que a empresa possa alcançar o nível de maturidade desejado. O nível que será abordado nesse trabalho é o Nível G, Parcialmente Gerenciado, que possui as Gerências de Projeto (GPR) e de Requisitos (GRE) sendo que dessas gerências apenas a segunda será discutida.

Para o guia MPS.BR [8], a GRE deve gerenciar os requisitos do produto e identificar inconsistências entre os requisitos, planos de projeto e os produtos de trabalho do projeto. Essa gerência também deve controlar a evolução dos requisitos, controle de mudanças, revisão continua, documentar mudanças e suas devidas justificativas e manter a rastreabilidade.

Os seguintes resultados são esperados para a Gerência de Requisitos do MPS.BR:

1. **GRE1** - O entendimento dos requisitos é obtido junto aos fornecedores de requisitos;
2. **GRE2** - Os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com estes requisitos é obtido;

3. **GRE3** - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida;

4. **GRE4** - Revisões em planos e produtos de trabalho do projeto são realizadas visando a identificação e correção de inconsistências em relação aos requisitos;

5. **GRE5** - Mudanças nos requisitos são gerenciadas ao longo do projeto;

### 2.4.2 Guia CMMI

No âmbito internacional, o CMMI (*Capability Maturity Model Integration*), Modelo Integrado de Maturidade em Capacitação, é um modelo de referência que contém a especificação das melhores práticas para melhoria dos processos de uma empresa de *software*. Uma das premissas do CMMI é a integração entre pessoas com habilidades, treinamento e motivação, ferramentas e procedimentos e métodos para que os produtos e serviços desenvolvidos possuam qualidade final através de processos utilizados pela organização.

O CMMI é composto por diversos CMMs (*Capability Maturity Model*), Modelos de Maturidade em Capacitação, que são elementos essenciais para auxiliar nos processos de desenvolvimento, manutenção e aquisição de produtos ou serviços.

A atual versão do CMMI, versão 1.3, apresenta três modelos de referência:

1. CMMI para Desenvolvimento (CMMI-DEV): voltado ao processo de desenvolvimento de produtos e serviços;

2. CMMI para Aquisição (CMMI-ACQ): voltado aos processos de aquisição e terceirização de bens e serviços;

3. CMMI para Serviços (CMMI-SVC): voltado aos processos de empresas prestadoras de serviços;

O CMMI-DEV consiste no conjunto de práticas voltadas para organização e o conjunto de atividades para o desenvolvimento de produtos e serviços de forma que esses sejam acompanhados desde a sua concepção até a entrega final e manutenção, sendo elas focadas em desenvolvimento e manutenção dos requisitos, soluções técnicas, integração de produtos, verificação e validação de processos.

O CMMI-DEV utiliza a definição de níveis para que as empresas que desejam melhorar seus processos possam ter conhecimento do atual nível dos seus processos bem como melhorar processos já existentes. Existem duas formas de representação de níveis: representação contínua (níveis de capacidade) e a representação por estágios (níveis de maturidade).

### Níveis de Capacidade

São aplicados para a melhoria de processos de forma separada, assim, processos de uma mesma área podem ser avaliados separadamente e estar em diferentes níveis. Os níveis de capacidade são classificados de 0 a 3, sendo eles:

1. Nível 0 - Incompleto: refere-se a um processo que não é executado ou é parcialmente executado;

2. Nível 1 - Executado: refere-se a um processo que atende a todos os objetivos estabelecidos pela área de processo;

3. Nível 2 - Gerenciado: refere-se a um processo que é executado de acordo com os interesses da organização, aloca as pessoas adequadas e com capacidade e avalia a adesão do processo com a área de processo em questão;

4. Nível 3 - Definido: refere-se a um processo que possui uma descrição mantida e segue os padrões da organização;

### Níveis de Maturidade

São aplicados para avaliação de um conjunto de processos de uma mesma área. Para alcançar um nível superior, a organização deve atender a todas as áreas de processos de um nível inferior. Os níveis de maturidade são classificados de 1 a 5, sendo eles:

1. Nível 1 - Inicial: as organizações que se encontram nesse nível não possuem processos ou os implementou de forma imatura/caótica. Tais organizações conseguem executar a entrega de produtos e serviços contudo, excedem os prazos e orçamentos acordados;

2. Nível 2 - Gerenciado: os processos são planejados e executados de acordo com os interesses da organização. Possuem pessoas e recursos adequados para executar, monitorar, controlar e revisar o produto ou serviço a ser entregue;

3. Nível 3 - Definido: os processos são descritos e caracterizados em normas, procedimentos, ferramentas e métodos. Cada novo projeto da organização deve adaptar seus processos ao processo estabelecido pela organização;

4. Nível 4 - Quantitativamente gerenciado: a organização estabelece objetivos quantitativos visando mensurar a qualidade e o desempenho do processo e os utilizam como base para gerenciamento do projeto. Esses objetivos podem ser as necessidades dos usuários finais ou da própria organização;

5. Nível 5 - Em otimização: nesse último nível, a organização utiliza métricas quantitativas para auxiliar na compreensão dos seus processos e buscam propor melhorias incrementais e inovadoras aos seus processos;

O CMMI-DEV possui 22 áreas de processos, sendo que nesse livro abordaremos apenas 2 dessas áreas, sendo elas: Gerenciamento de Requisitos e Desenvolvimento de Requisitos. Seguindo a representação por estágios (níveis de maturidade), o gerenciamento de requisito encontra-se no nível 2 (gerenciado) já o desenvolvimento de requisitos se encontra no nível 3 (definido).

### Gerenciamento de Requisitos

Responsável por manter os requisitos, essa área de processo descreve o conjunto de atividades para obter e controlar as mudanças nos requisitos. Esse processo também visa proporcionar a rastreabilidade dos requisitos do cliente ao produto final bem como dos componentes do produto gerado.

O gerenciamento de requisitos é uma área dinâmica na qual as mudanças nos requisitos devem refletir nos planos do projeto, atividades e produtos do trabalho. Além disso, as mudanças nos requisitos podem afetar outras áreas de processos.

#### Objetivos Específicos

1. **Gerenciamento de Requisitos**: Os requisitos são gerenciados e as inconsistências com planos do projeto são identificadas e tratadas. Todos os requisitos selecionados para o projeto devem estar submetidos a uma analise das mudanças nos mesmos, manter o relacionamento entre os requisitos e planos de projeto e conter ações corretivas para erros os inconsistências.

   (a) Prática Específica 1 - Entendimento dos Requisitos: os requisitos devem ser entendidos juntamente com os fornecedores dos requisitos (clientes interessados no desenvolvimento do projeto e que possuem conhecimento das regras de negócio);

(b) Prática Específica 2 - Comprometimento com os Requisitos: conjunto de práticas e acordos entre aqueles que são responsáveis por realizar atividades necessárias para implementar os requisitos;

(c) Prática Específica 3 - Gerenciar alterações nos Requisitos: Os requisitos de *software* mudam a medida que eles evoluem no projeto ou a perspectiva dos interessados muda. É essencial gerenciar essas mudanças de forma eficiente e eficaz. Para isso, é necessário conhecer origem de cada requisito bem como a lógica da implementação;

(d) Prática Específica 4 - Manter a Rastreabilidade Bidirecional dos Requisitos: quando os requisitos são bem gerenciados é possível manter uma ligação entre os requisitos e ter conhecimento entre as origens dos requisitos para os requisitos e quais requisitos e planos de trabalhos foram originados a partir de um requisito;

(e) Prática Específica 5 - Garantir o alinhamento entre Produtos do Trabalho e os Requisitos: utilizada para identificar as inconsistências entre os requisitos e entre os requisitos e produtos de trabalho e a partir dessas inconsistências, ações corretivas são estabelecidas para resolução;

### Desenvolvimento de Requisitos

Responsável pela elicitação, análise e estabelecimento das necessidades dos clientes. Os requisitos elicitados podem ser de três tipos: requisitos do cliente, requisitos do produto e requisitos dos componentes do produto. Esse conjunto de requisitos descrevem as características que o produto deve atender e auxilia nas decisões de *design* do projeto, arquitetura do projeto, tecnologias a serem utilizadas.

#### Objetivos Específicos

1. **Desenvolver Requisitos do Cliente**: As necessidades dos interessados na concepção do *software*, sejam eles, usuários finais, testadores, gerentes, desenvolvedores e outros, devem ser analisadas, refinadas e elaboradas em conjunto de requisitos dos clientes. Geralmente a identificação dos requisitos são pouco conhecidas em sua totalidade no início do projeto ou são conflitantes, logo, um processo interativo com o cliente deve ser mantido para obter um melhor detalhamento dos requisitos e resolver e/ou evitar conflitos.

(a) Prática Específica 1 - Elicitar os Requisitos: consiste na prática de descobrir os requisitos do produto a ser desenvolvido mediante interação com o cliente. Além disso, novos requisitos podem ser descobertos mediante uma análise mais crítica e pontual que os responsáveis pelo desenvolvimento do produto podem obter e que os clientes não conseguiram transmitir;

(b) Prática Específica 2 - Transformar as necessidades dos clientes em requisitos de clientes: o conhecimento transmitido através dos *stakeholders* deve ser consolidado, informações incompletas devem ser preenchidas e conflitos entre as informações devem ser resolvidos. Os requisitos dos clientes podem representar necessidades, expectativas, regras de negócio e regras de verificação e validação;

2. **Desenvolver Requisitos do Produto**: Os requisitos dos clientes devem ser analisados e refinados buscando um melhor detalhamento para que possam ser obtidos os requisitos do produto. Os requisitos podem surgir de restrições, questões implícitas que não foram encontradas nos requisitos dos clientes (descoberta pós-análise), *rationale* da arquitetura. A rastreabilidade dos requisitos para funções, objetos, casos de testes e outros deve ser documentada.

   (a) Prática Específica 1 - Estabelecer Requisitos de Produto e Componente de Produto: com base nos requisitos dos clientes citados anteriormente, um conjunto de requisitos do produto devem ser estabelecidos e mantidos. Tais requisitos são a demonstração dos requisitos dos clientes sob forma técnica e que auxilie engenheiros de *software* em decisões técnicas;

   (b) Prática Específica 2 - Alocar Requisitos dos Componentes do Produto: a arquitetura deve ser construída com base nos requisitos nos produtos de forma que atenda a premissas estabelecida pelos requisitos tais como segurança, confiabilidade, tempo de resposta e outras;

   (c) Prática Específica 3 - Identificar requisitos de interface: uma interface entre requisitos ou entre componentes do produto devem ser identificadas. Eles são controlados como parte de integração de componentes de produtos e são parte integrante da definição de arquitetura.

3. **Analisar e Validar os Requisitos**: A análise e validação devem servir como apoio para o desenvolvimento de requisitos dos clientes e dos requisitos dos produtos. A análise e validação devem ser feitas sempre

com o pensamento no ambiente final pretendido pelo cliente solicitante do produto. As análises são realizadas para estimar o impacto que uma modificação em um requisito ou nova solicitação por parte dos clientes pode ocasionar ao ambiente operacional. E as validações são utilizadas para aumentar a probabilidade do produto funcionar conforme desejado pelos clientes.

(a) Prática Especifica 1 - Estabelecer Conceitos Operacionais e Cenários: buscar o entendimento das funcionalidades, desempenho, manutenção e suporte. O ambiente em que o produto irá operar, incluindo limites e restrições;

(b) Prática Especifica 2 - Estabelecer uma Definição de Funcionalidades Exigidas: as funcionalidades exigidas pelos usuários finais devem ser analisadas e quantificadas;

(c) Prática Especifica 3 - Analisar os Requisitos: busca garantir que todas as necessidades dos clientes são necessárias e suficientes de forma que o produto gerado esteja atendendo as expectativas. A análise dos requisitos também visa identificar requisitos cruciais que possuem forte influência em outros requisitos, custo, funcionalidade e performance bem como verificar se o requisito esta completo, testável e verificável;

(d) Prática Especifica 4 - Analisar os Requisitos para Alcançar o Equilíbrio: buscar um acordo com o cliente sobre o desenvolvimento dos requisitos para o produto final de forma que critérios como custo, cronograma, desempenho do produto, manutenção e riscos não sejam fatores negativos ao produto;

(e) Prática Especifica 5 - Validar Requisitos: garantir que o produto final funcionará de acordo com as expectativas do usuário final;

Para avaliar se uma organização está em conformidade com os critérios e requisitos estabelecidos pelo CMMI ou se a mesma deseja conquistar o certificado em um determinado nível ou área de processo, um processo de avaliação deve ser feito para verificar se a empresa atende os critérios de um determino nível. Para prover um mecanismo de avaliação, a equipe do CMMI publicou o *Appraisal Requirements for CMMI* na versão 1.1 (ARC v1.1) que define os requisitos e critérios considerados como essenciais para o processo de avaliação do CMMI em uma organização. O **ARC** possui três classes para avaliação, sendo elas: A, B e C. Diferentes método de avaliação foram criados para atender as três classes citadas anteriormente contudo, somente a classe A possibilita uma graduação para o próximo nível.

### SCAMPI

Para avaliar o **CMMI** que esteja em processo de implantação ou até mesmo que já esteja implantado em uma empresa, o SCAMPI (*Standard CMMI Appraisal Method for Process Improvement*), método avaliativo para a **classe A**, é o método utilizado para identificar os pontos fracos e fortes dos atuais processos da organização, estimar riscos para auxiliar no desenvolvimento ou aquisição de produtos e serviços e determinar os níveis de capacidade e maturidade citados anteriormente. Essa avaliação de crucial importância para que a empresa possa conhecer seus processos, planejar melhorias e obter/prover melhores produtos e serviços.

Esse método define o conjunto de processos, atividades, documentos e agentes necessários para a avaliação além disso, descreve práticas para preparação do local para avaliação, descobertas, índices, classificações, relatórios e atividades corretivas.

Sua abordagem consiste em coletar dados e evidências através de uso de técnicas tais como *brainstorms*, formulários, questionários, entrevistas individuais e coletivas, avaliação de documentos e outros. Esses dados coletados são analisados e utilizados para verificar se a organização atende a critérios do CMMI.

**Fases do SCAMPI**

1. **Planejamento e Preparação da Avaliação**
   (a) Análise dos Requisitos;
   (b) Desenvolvimento do plano de avaliação;
   (c) Seleção e preparação da equipe;
   (d) Obtenção e análise de evidência de objetivo inicial;
   (e) Preparação para a coleta de evidência de objetivo;
2. **Condução da Avaliação**
   (a) Exame de evidência de objetivo;
   (b) Verificação e validação de evidência de objetivo;
   (c) Documentação de evidência de objetivo;
   (d) Geração de resultados de avaliação;
3. **Relatos dos Resultados**
   (a) Entrega de resultados de avaliação;
   (b) Arquivamento das informações de avaliação;

# Parte II

# Artefatos

# 3

# Artefatos e Documentação

Nesse capítulo serão apresentados os artefatos previstos pelo guia AADSP para o gerenciamento de requisitos. Conforme estabelecido pelo guia, cada artefato possui um grau de importância e relevância. Assim como os artefatos, os itens que os compõem também devem possuir um grau de importância pois nem todas as organizações podem conseguir implementá-los completamente por ainda estarem em processo de amadurecimento.

## 3.1 Artefatos

### 3.1.1 Documento de Requisitos (Essencial)

O documento de requisitos de um *software* deve contemplar todos os requisitos que serão atendidos pelo *software* que será desenvolvido. Tal documento representa todas as necessidades que o *software* deve atender e deve estar em conformidade com as necessidades dos clientes solicitantes.

Todos os requisitos contidos nesse documento devem possuir um **identificador único** de forma que este possa ser identificado em qualquer fase do projeto e seja facilmente rastreável quando relacionado a outros requisitos e a artefatos tais como diagramas arquiteturais, casos de testes e outros.

**Especificação do Projeto**

1. **Descrição do Projeto** (Essencial);
2. **Data de Início** (Essencial);
3. **Data do Término** (Essencial);
4. **Informações Ambientais** (Desejável);

**Especificação dos Requisitos do Projeto**

1. **Identificador Único** (Essencial);
2. **Versão** (Essencial);
3. **Título** (Essencial);
4. **Status** (Importante);
5. **Importância** (Importante);
6. **Tipo/Subtipo** (Essencial);
7. **Rationale** (Essencial);
8. **Descrição** (Essencial);
9. **Responsável** (Essencial);
10. **Solicitante** (Essencial);
11. **Custo** (Desejável);
12. **Ponto de Função** (Desejável);

### 3.1.2 Documento de Mudanças dos Requisitos (Essencial)

Os requisitos de *software* são altamente voláteis e podem ser modificados a qualquer momento dentro do ciclo de vida de um *software* seja por fatores tecnológicos, políticos, sociais, econômicos e novas perspectivas por parte dos interessados no projeto. Essas mudanças não devem ser vistas como aspectos negativos pois as mesmas visam atender novos desafios e melhor atender os clientes. Contudo, essas mudanças devem ser bem documentadas e versionadas. Cada solicitação de mudança deve ser justificada por parte dos clientes informando assim os propositivos e motivos que levaram à solicitação da mudança (*rationale*) e caso essa modificação seja aceita, a equipe de desenvolvimento de *software* deve versionar o requisito e estimar o impacto daquela modificação.

1. **Identificador do Requisito** (Essencial);
2. **Soliciantes** (Essencial);
3. **Data da Solicitação** (Essencial);

4. **Status da Modificação**[1] (Essencial);

5. ***Rationale* (Motivação)** (Importante);

### 3.1.3 Aprovação dos Clientes (Essencial)

### 3.1.4 Comprometimento da Equipe (Importante)

Todos os indivíduos que estejam vinculados a um projeto e de maneira específica a um requisito, devem comprometer-se com o cumprimento do conjunto de tarefas para o sucesso do projeto e atendimento dos requisitos.

**Comprometimento da Equipe para com o Projeto**

1. **Identificador do Projeto** (Essencial);

2. **Descrição do Projeto** (Essencial);

3. **Listagem de todos os Stakeholders com o descritivo de suas atividades e o seu grau de importância para o projeto** (Essencial);

**Comprometimento da Equipe para com o Requisito**

1. **Identificador do Requisito** (Essencial);

2. **Descrição do Requisito** (Essencial);

3. **Listagem de todos os Stakeholders com o seu grau de importância para o requisito** (Essencial);

### 3.1.5 Rastreabilidade

A rastreabilidade, atividade mais importante do gerenciamento de requisitos, tem como proposito mapear o ciclo de vida de um requisito desde a sua origem à todos os artefatos ou outros requisito com os quais ele possui algum vinculo.

[1]Solicitada: os Stakeholders solicitaram a modificação contudo, a mesma não foi avaliada pela gerência; Em Analise: a gerência está avaliando a viabilidade da mudança; Rejeitada: a gerência recusou a solicitação de mudança;

**Matriz de Rastreabilidade entre Requisitos (Essencial)**

A matriz de rastreabilidade entre requisitos objetiva demonstrar quais são os relacionamentos entre os requisitos de forma que possa ser estimado o impacto que a modificação em um requisito pode afetar outros requisitos bem como visualizar as dependências e precedências entre os requisitos para auxiliar no processo de seleção e desenvolvimento.

1. **Requisitos de Origem** (Essencial): Preferivelmente dispostos na primeira coluna da matriz, esses requisitos representam os requisitos que deram origem ao elo (ligação) para com o requisito de destino.

2. **Requisitos de Destino** (Essencial): Preferivelmente dispostos na primeira linha da matriz, esses requisitos representam os requisitos que foram oriundos a partir de um requisito disposto na primeira coluna da matriz.

3. **Sinalização do Tipo de Ligação** (Essencial): Tradicionalmente a sinalização visual da matriz pode ser feita com o uso de um "X"contudo, a deve ser armazenado o tipo de ligação entre os requisitos. Por exemplo, o requisito **A** disposto na primeira coluna da matriz possui uma ligação de **dependência** com o requisito **B** disposto na primeira linha da matriz;

**Matriz de Rastreabilidade entre Requisito e Artefatos (Importante)**

A matriz de rastreabilidade entre requisito e artefatos objetiva demonstrar quais artefatos estão vinculados a um de forma que a equipe de desenvolvimento consiga rastrear e verificar se um requisito possui algum vinculo com a arquitetura do sistema, casos de uso, casos de testes ou qualquer outro artefatos que traga melhores especificações e resultados positivos ao requisito.

1. **Requisitos** (Essencial);

2. **Artefatos** (Essencial);

3. **Sinalização do Tipo de Ligação** (Essencial);

### 3.1.6 Checklist de Aceitação (Importante)

Para o **IEEE** [11] e **MPS.BR** [8], bons requisitos de *software* apresentam um conjunto de características para que possam ser avaliados pela equipe técnica

para uma posterior avaliação dos clientes sendo elas: identificação única, clareza, indutibilidade, relevância, completude, consistência, classificação, implementabilidade, testabilidade e rastreabilidade, assim como apresentado na Seção 2.1.1.

# Parte III

# Avaliação AADSP

# 4

# Avaliação AADSP

Nesse capítulo serão elencados o conjunto de critérios para a avaliação dos artefatos requeridos pelo guia AADSP para a gerência de requisitos sendo eles: documento de requisitos, documento de mudanças dos requisitos, comprometimento dos envolvidos, rastreabilidade entre requisitos e entre requisitos e artefatos. Para o processo de avaliação e auditoria dos artefatos o guia AADSP propõe o uso do paradigma **GQM** (*Goal/Questions/Metrics*) em conjunto com a escala Likert. Amplamente utilizada na Engenharia de *Software* o **GQM** baseia-se na definição de uma uma Meta (*Goal*) e um conjunto de Questões (*Questions*) são estabelecidas para auxiliar na compreensão da Meta, além disso, um conjunto de Métricas (*Metrics*) são estabelecidas para avaliar essas questões. Para auxiliar no processo de auditoria e validação dos artefatos, o guia AADSP sugere a substituição das Métricas do **GQM** pela avaliação com a Escala Likert [15] que proporciona uma avaliação quantitativa para as questões. Todas as questões avaliativas devem ser respondidas seguindo o formato tradicional da Escala Likert:

1. Discordo totalmente (1);
2. Discordo parcialmente (2);
3. Indiferente (3);
4. Concordo parcialmente (4);
5. Concordo totalmente (5);

Cada artefato possui um conjunto de **Questões Avaliativas** (QA) que serão respondidas conforme a Escala Likert acima representada. A pontuação que o artefato receberá será composto pelo somatório de todas as QAs divido pela quantidade de QAs do artefato em questão.

## 4.1 Avaliação do Documento de Requisitos

Tabela 4.1: Definição da meta para avaliação do Documento de Requisitos

| | |
|---|---|
| Objeto de Estudo | Documento de Requisitos |
| Propósito | Validar o documento de requisitos gerado frente ao conjunto de critérios estabelecidos pelo guia AADSP |
| Fator de Qualidade | O documento gerado auxilia no processo de desenvolvimento e promove uma visão uniforme entre os clientes e desenvolvedores |
| Perspectiva | Sob o ponto de vista de clientes e desenvolvedores envolvidos na confecção do *software* |
| Contexto | Avaliação feita juntamente com uma empresa que deseja obter melhorias em seu processo de *software* e a certificação AADSP. |

### 4.1.1 Questões Avaliativas

1. O documento de requisitos possui informações sobre o projeto;
2. Os requisitos estão vinculados a um projeto;
3. O cadastro dos requisitos é feito mediante a utilização de ferramentas;
4. A empresa utiliza *templates* pré-definidos para o cadastro dos requisitos;
5. Os requisitos são versionados a cada modificação;
6. Os requisitos dispostos no documento possuem identificador único;
7. Os requisitos dispostos no documento estão claros e bem redigidos;
8. Os requisitos dispostos no documento não apresentam ambiguidade;
9. Os requisitos dispostos no documento são completos;
10. Os requisitos dispostos no documento são consistentes;
11. Os requisitos dispostos no documento são classificáveis;
12. Os requisitos dispostos no documento são/foram implementados;

13. Os requisitos dispostos no documento são testáveis;

14. Os requisitos dispostos no documento são rastreáveis;

## 4.2 Avaliação do Documento de Mudanças dos Requisitos

Tabela 4.2: Definição da meta para avaliação do Documento de Mudanças dos Requisitos

| | |
|---|---|
| Objeto de Estudo | Documento de Mudanças dos Requisitos |
| Propósito | Validar o documento de mudanças disponibilizado para empresa frente ao conjunto de critérios estabelecidos pelo guia AADSP |
| Fator de Qualidade | O documento de mudanças demonstra todas as solicitações de mudanças feitas para cada requisito do projeto |
| Perspectiva | Sob o ponto de vista de clientes e desenvolvedores que enfrentaram mudanças em requisitos já implementados |
| Contexto | Onde os requisitos do *software* desenvolvido passaram por solicitações de mudanças |

### 4.2.1 Questões Avaliativas

1. Cada requisito do documento de mudanças está referenciado com seu identificador único;

2. Os requisitos apresentados no documento de mudanças possuem os solicitantes das mudanças;

3. A data da solicitação de mudança consta em cada requisito;

4. O status da solicitação de mudança está presente em cada requisito;

5. A motivação por trás da solicitação de mudança consta em cada requisito;

Tabela 4.3: Definição da meta para avaliação do Documento de Comprometimento da Equipe

| | |
|---|---|
| Objeto de Estudo | Documento de Comprometimento da Equipe |
| Propósito | Validar o comprometimento da equipe para com o projeto e para os requisitos |
| Fator de Qualidade | O comprometimento técnico é crucial para o sucesso do projeto pois todos os indivíduos sabem suas responsabilidades para com o projeto e os requisitos |
| Perspectiva | Sob o ponto de vista da equipe responsável pela criação do *software* |
| Contexto | Avaliação feita juntamente com uma empresa que deseja obter melhorias em seu processo de *software* e a certificação AADSP. |

## 4.3 Avaliação do Comprometimento da Equipe

1. O(s) documento(s) de comprometimento da equipe para com o projeto e para com os requisitos possuem um identificador único, tanto para os projetos quanto para os requisitos;
2. A descrição dos objetos (projeto e requisitos) é mantida no documento;
3. O conjunto de atividades de cada *stakeholder* é especificado;
4. O grau de importância de cada *stakeholder* para o projeto é especificado;
5. O grau de importância de cada *stakeholder* para o requisito é especificado;

## 4.4 Avaliação da Rastreabilidade entre Requisitos

### 4.4.1 Questões Avaliativas

1. A empresa utilizou *softwares* para geração da matriz de rastreabilidade;
2. Os requisitos apresentados na matriz possuem identificador único;
3. Os requisitos de origem são facilmente distinguidos dos de origem;

Tabela 4.4: Definição da meta para avaliação do Documento de Rastreabilidade entre Requisitos

| | |
|---|---|
| Objeto de Estudo | Documento de Rastreabilidade entre Requisitos |
| Propósito | Validar o documento de rastreabilidade entre requisitos |
| Fator de Qualidade | Difundir o conhecimento entre requisitos que possuem alguma conexão (dependência, precedência e outras) para uma melhor avaliação de impacto em casos de mudanças nos requisitos |
| Perspectiva | Sob o ponto de vista de *stakeholders* que desejam efetuar ações estratégicas sobre os requisitos |
| Contexto | Avaliação feita juntamente com uma empresa que deseja obter melhorias em seu processo de *software* e a certificação AADSP. |

4. O tipo de ligação entre os requisitos (precedência, dependência, satisfação e outros) é disponibilizado;

## 4.5 Avaliação da Rastreabilidade entre Requisitos e Artefatos

Tabela 4.5: Definição da meta para avaliação do Documento de Mudanças dos Requisitos

| | |
|---|---|
| Objeto de Estudo | Documento de Rastreabilidade entre Requisitos e Artefatos |
| Propósito | Validar o documento de rastreabilidade entre requisitos e artefatos |
| Fator de Qualidade | Tal documento demonstra de maneira claro e objetiva a quais artefatos os requisitos estão vinculados |
| Perspectiva | Sob o ponto de vista de *stakeholders* que desejam efetuar ações estratégicas sobre os requisitos |
| Contexto | Avaliação feita juntamente com uma empresa que deseja obter melhorias em seu processo de *software* e a certificação AADSP. |

### 4.5.1 Questões Avaliativas

1. A empresa utilizou *softwares* para geração da matriz de rastreabilidade;
2. Os requisitos apresentados na matriz possuem identificador único;
3. Os artefatos apresentados na matriz possuem identificador único;
4. É possível visualizar em quais artefatos um requisito está presente e em quais não está presente;
5. O tipo de ligação entre os requisitos e os artefatos (precedência, dependência, satisfação e outros) é disponibilizado;

# Parte IV

# Ferramentas

# 5

# Ferramentas

Nesta seção são apresentadas e analisadas as funcionalidades de ferramentas confeccionadas para a área de Gerenciamento de Requisitos.

1. **T-AADSP-Requirements**[1]: Desenvolvida pelo grupo de pesquisa LABRASOFT, essa ferramenta busca auxiliar no gerenciamento de requisito com base nos guia MPS.BR e os principais critérios do gerenciamento de requisitos estabelecidos pela Engenharia de Software. Um dos principais diferenciais dessa ferramenta é a geração de *Dataset* para de Otimização em Engenharia de Software também conhecida como *Search-Based Software Engineering*;

2. **Enterprise Architect**[2]: Criada pela Sparx Systems, essa ferramenta auxilia na visualização e gerenciamento dos requisitos e na integração dos mesmos com o ambiente de desenvolvimento. A ferramenta também possui a modelagem de Casos de Uso da UML, importação de requisitos com base no formato CSV, a rastreabilidade de requisitos na implementação do *software*, análise de impacto em modificações, na comunicação entre os *stakeholders* e a criação de um dicionário de comunicação para ajudar na comunicação.

3. **Controla**[3]: Ferramenta de apoio ao gerenciamento de requisitos resultante de estudos comparativos entre as metodologias de gerenciamento de projetos. Essa ferramenta possibilita a elicitação dos requisitos junto aos *stakeholders*, detalhamento, gerenciamento de mudanças, controle de versões, matriz de rastreabilidade, avaliação de impacto e outras.

[1] http://www.labrasoft.ifba.edu.br/

[2] http://www.sparxsystems.com.au/products/ea/requirements.html

[3] http://www.periodicosibepes.org.br/index.php/reinfo/article/viewFile/156/48

4. **AvenqoPEP**[4]: Voltada para o gerenciamento de requisitos, essa ferramenta permite o cadastro de requisitos com classificação de riscos, *status* do requisito, o anexo de arquivos ao requisito e a criação de relacionamento entre requisitos (elos). Um dos diferenciais dessa ferramenta é a sinalização de impactos quando um requisito ligado a outro é alterado e a associação de tarefas aos requisitos.

5. **Caliber**[5]: Criado pela Micro Focus[6], esse *software* fornece um gerenciamento dos requisitos com o uso de uma abordagem iterativa e colaborativa para definir e gerenciar requisitos em todas as fases da Engenharia de Requisitos de forma que os requisitos estejam em conformidade com as necessidades dos clientes. Assim como algumas outras ferramentas, o Caliber fornece visualização dos requisitos, rastreabilidade e análise de impacto em tempo real.

6. **IBM Rational DOORS**[7]: Solução para gerenciamento de requisitos que ajuda a empresa a gerenciar o escopo e custo do projeto. Permite a captura, rastreio, gerenciamento de mudanças e a criação de casos de testes para os requisitos cadastrados na ferramenta.

| **CARACTERÍSTICA** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| Estabelece relacionamentos entre cada requisito e suas fontes (Fornecedores de Requisitos associados) | S | NA | P | P | NA | S |
| Estabelece dependências entre Requisitos Funcionais (incluindo Regras de Negócio), Casos de Uso, Requisitos Não-Funcionais, Requisitos de Dados e Telas | S | S | P | S | S | S |
| Estabelece dependências entre os requisitos e os artefatos de design | S | S | N | S | S | S |
| Estabelece dependências entre os requisitos e os artefatos de implementação | S | S | S | S | NA | S |

[4]http://www.avenqo.com/
[5]https://www.microfocus.com/pt-br/products/requirements-management/caliber/
[6]https://www.microfocus.com/pt-br/
[7]https://www-03.ibm.com/software/products/pt/ratidoor

| | | | | | | |
|---|---|---|---|---|---|---|
| Dado um requisito, ou qualquer outro artefato, mostra as dependências do artefato em questão com o restante dos artefatos | S | S | N | S | NA | S |
| Permite definir requisitos e casos de uso na própria ferramenta (verificar se há *template* para definição destes artefatos) | S | S | S | S | S | S |
| Estabelece e indica dependências entre produtos gerados (dado um sistema, mostra as suas dependências com os demais sistemas) | S | NA | N | S | NA | P |
| É fácil identificar na Matriz de Rastreabilidade o requisito em questão | S | NA | S | S | NA | S |
| Divulga a Matriz de Rastreabilidade para todos os interessados | S | S | N | S | NA | S |
| Evidencia requisitos que não estão sendo implementados em qualquer artefato de projeto ou de implementação | S | NA | N | S | NA | N |
| Evidencia artefatos que não estão associados a requisitos | S | NA | N | S | NA | S |
| Verifica a existência de integração entre ferramentas de design e de implementação | S | S | N | S | NA | N |

**Ferramentas** – (1) T-AADSP-Requirements; (2) Caliber; (3) Controla; (4) Enterprise Architect; (5) Doors; (6) Avenqo;

**AVALIAÇÃO**

1. (S) Satisfaz o requisito em questão;
2. (N) Não atende o requisito;
3. (P) Atende Parcialmente o requisito;
4. (NA) O requisito em questão não foi avaliado;
5. (NE) A informação para avaliação do requisito Não foi Encontrada;

# Parte V

# Conclusão

# 6

# Conclusão

Esse livro apresentou o gerenciamento de requisitos sob o ponto de vista do guia AADSP. Foram discutidos os principais aspectos da Engenharia de Requisitos e as principais atividades que a compõem, sendo a principal o gerenciamento de requisitos. Além disso, o gerenciamento de requisitos promove a identificação, controle de mudanças e rastreabilidade dos mesmos como atividades centrais sendo a rastreabilidade a atividade mais importante por conseguir divulgar todas as origens, destinos e dependências entre requisitos e artefatos. Essa gerência difunde uma visão uniforme e linear a todos os interessados nos requisitos do projeto, desde os desenvolvedores aos clientes. Contudo, a aplicação do gerenciamento de requisito deve variar de empresa para empresa e de projeto para projeto de acordo com o contexto socio-econômico-cultural na qual será aplicada.

Para comprovar o atendimento das premissas estabelecidas pelo gerenciamento de requisitos, o guia AADSP baseia-se na entrega de artefatos como forma de validação e comprovação frente ao *software* que fora desenvolvido. Para o gerenciamento de requisitos foram elencados oito artefatos sendo que para cada um deles foi apresentado os campos (atributos) que o compõem, bem como o grau de importância. Além disso, para que os aderentes possam saber em qual nível o artefato encontra-se e como podem melhorar o mesmo, uma nota deve ser aplicada pelo auditor ao final do processo de auditoria. Além disso, o guia AADSP acredita que a geração de dados e *Datasets* com base nos requisitos de uma organização pode auxiliar no processo de melhoria dos processos e contribuição para a área de *Search-Based Software Engeneering* com a aplicação de algoritmos genéticos tais como o *The Next Release Problem* para selecionar os requisitos mais adequados e prioritários para a organização e seus *stakeholders*.

O processo de auditoria dos artefatos tem como objetivo checar se a empresa cumpriu todos os critérios estabelecidos pelos artefatos da gerência.

Para a avaliação de cada artefato previsto pelo guia AADSP uma meta foi criada com base no método **GQM** e um conjunto de questões avaliativas deve ser respondido baseado na Escala Likert com as notas de 1 a 5 para saber se o artefato atingiu ou não a meta e em quais questões especificas o artefato pode ser melhorado. Com todas as questões respondidas, o auditor pode inferir uma nota ao artefato baseado no somatório das questões avaliativas divido pela quantidade de questões.

Espera-se que com a entrega dos artefatos e o atendimento as questões avaliativas estabelecidas pelo mesmo as empresas aderentes ao guia AADSP consigam melhorar o seu processo de *software*, proporcionar uma melhor qualidade aos seus produtos e satisfação ao seu cliente final. Para o gerenciamento de requisitos os artefatos foram criados visando fomentar uma base sólida que fornecesse apoio as demais gerências estabelecidas pelo guia AADSP. Haverão outros livros publicados para cada uma das gerências previstas pelo guia AADSP sendo elas: projetos, testes, colaboradores, reuso e configuração e mudança.

# Bibliografia


[1] E. C. Genvigir, "Um modelo para rastreabilidade de Requisitos de Software baseado em Generalização de Elos e Atributos.," *Instituto Nacional de Pesquisas Espaciais - INPE*, pp. 27–70, 2009.

[2] B. Ramesh, "Factory influencing requirements traceability practice," *ACM*, vol. 41, no. 12, pp. 37–44, 1998.

[3] M. Toranzo and J. Castro, "Uma proposta para melhorar o rastreamento de requisitos," *WER02 - Workshop em Engenharia de Requisitos, Valencia, Espanha*, pp. 194–209, 2002.

[4] M. M. A. Brasil, F. G. de Freitas, T. G. N. da Silva, J. T. de Souza, and M. I. Cortés, "Uma nova abordagem de otimização multiobjetiva para o planejamento de releases em desenvolvimento iterativo e incremental de software," *I Workshop Brasileiro de Otimização em Engenharia de Software*, vol. 1, pp. 40–47, 2012.

[5] P. S. N. D. Lopes, "Uma taxonomia da pesquisa na Área de engenharia de requisitos," *Dissertação de Mestrado, IME/USP*, pp. 14–44, 2002.

[6] M. Sayão and J. C. S. P. Leite, "Rastreabilidade de requisitos," *Revista de Informática Teórica e Aplicada - RITA*, vol. XII, no. 1, pp. 1–22, 2005.

[7] I. Sommerville and G. Kotonya, *Requeriments Engineering: Process and Techniques*, ch. 5. Requeriments Engineering: Process and Technique, 1998.

[8] SOFTEX, "MPS.BR: melhoria de processo do software brasileiro," SOFTEX, 2012.

[9] P. R. C. Malcher, D. A. L. Ferreira, S. R. B. Oliveira, and A. M. L. de Vasconcelos, "Um mapeamento sistemático sobre abordagens de

apoio à rastreabilidade de requisitos no contexto de projetos de software," *Revista de Sistema de Informação da FSMA*, vol. 1, no. 16, pp. 3–15, 2015.

[10] AADSP, "Adaptive approach for deployment of software process." `http://www.labrasoft.ifba.edu.br/wp-content/uploads/2016/10/Guia-AADSP.pdf`, 2016 (acessado Dezembro 14, 2017).

[11] IEEE, "Ieee std 830-1998 - ieee recommended practice for software requirements specifications." `http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf`, 1998 (acessado Janeiro 21, 2018).

[12] R. S. de Espindola, A. Majdenbaum, and J. L. N. Audy, "Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software," *Anais do WER04 - Workshop em Engenharia de Requisitos*, pp. 226–238, 2004.

[13] S. de Rezende Alves and A. L. Alves, "Engenharia de requisitos em metodologias Ágeis," Pontifícia Universidade Católica do Rio Grande Sul, 2009.

[14] Y. Zhang, A. Finkelstein, and M. Harmn, "The multi-objective next release problem," *ACM. Proceedings of the 9th annual conference on Genetic and evolutionary computation*, vol. 1, pp. 1129–1137, 2007.

[15] T. C. Kinnear and J. R. Taylor, "Marketing research: an applied approach," McGraw Hill, 1991.

# Índice Remissivo


## A

AADSP 5, 47
análise 16
artefatos 14, 17, 19, 20, 47
avaliação 56

## C

CMMI 37
CMMI-DEV 37
comprometimento da equipe 49
controle de mudanças 18

## D

documentação 16
  documento de mudanças dos requisitos 48
documento de requisitos 47

## E

elicitação 16
elos 25
engenharia de requisitos 9, 16
engenharia de software 3, 9

## G

gerência de requisitos 6
gerenciamento 16
gerenciamento de requisitos 17, 39
GQM 55, 70
guias 18

## H

high-end users 24

## I

IEEE 13, 50

## L

Likert 55
likert 55, 70

## M

matriz de rastreabilidade 50
  matriz de rastreabilidade entre requisitos 50
Metamodelo 25, 27, 28, 31
modelos 25
modelos de maturidade 35
MPS.BR 13, 19, 23, 35
MPS.Br 1
mudanças nos requisitos 17

## N

NRP 33
nrp 69

## Q

qualidade 3

## R

rastreabilidade 18, 49
rastreabilidade de requisitos 19
rastreamento de requisitos 10
rationale 14, 41, 48
release 33
requisitos 13, 18

## S

**SBSE** 35, 69
**Scampi** 43
**scampi** 1
**software** 13
**stakeholders** 13, 17

## T

**The Next Release Problem** 33
**the next release problem** 69

## V

**validação** 16
**versionamento** 21