Olá, Professor!
Peço que realize a **avaliação do conteúdo** com o intuito de manter o nosso material sempre atualizado.

Avalie este conteúdo! 🚀 ✨

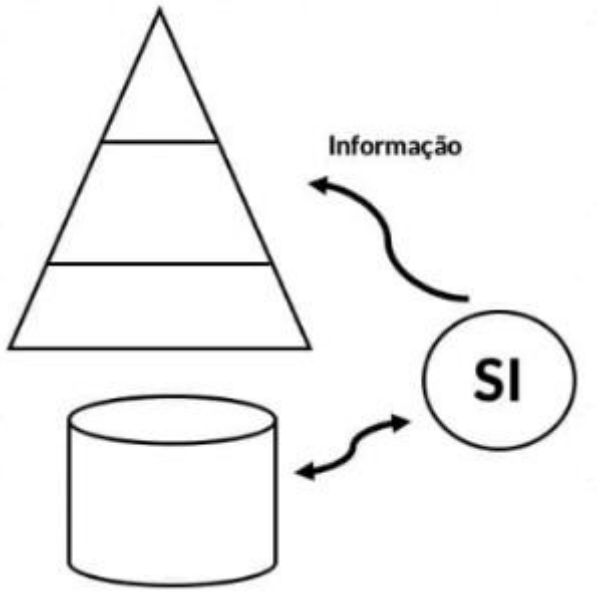
https://bit.ly/451BDqS

Engenharia de Software

Fundamentos de Software e Gerenciamento de Projetos

Software

- Avanço tecnológico do hardware impulsiona softwares mais complexos.
- Transição para o paradigma orientado a objetos permite reutilização e eficiência na manutenção.
- Software é crucial para geração de informações em Sistemas de Informação, essenciais para decisões e controle empresarial.



Sistema de Informação x Software.

Software

Desafios atuais de um engenheiro de Software.

Software de Sistema

Software de Aplicação

Software de Engenharia/ Científico

Software Embarcado

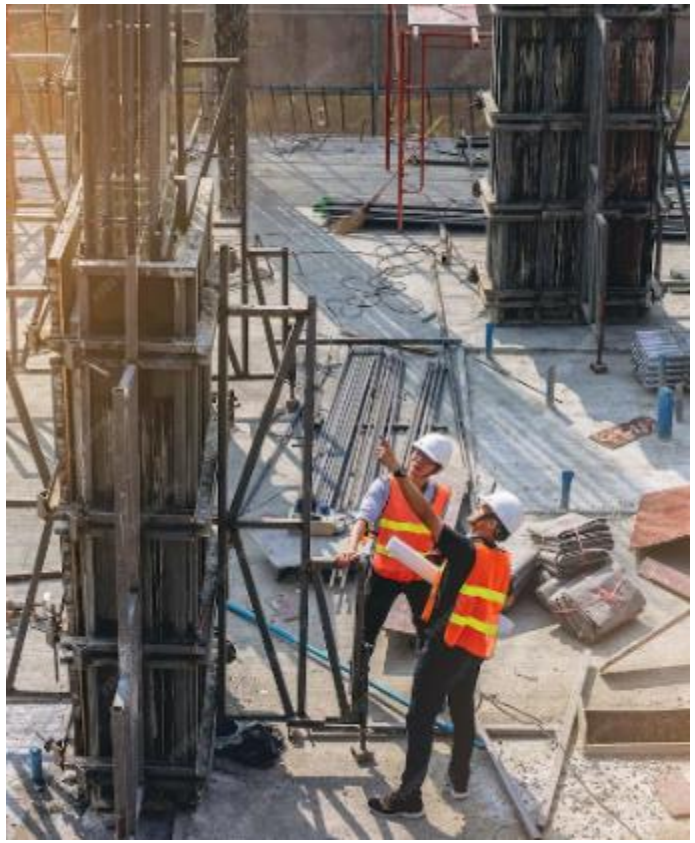
Software para Linha de Produtos

Aplicações Web/ Aplicativos Móveis

Software de Inteligência Artificial

Engenharia de Software

- A metodologia de decomposição sistemática é essencial para lidar com problemas complexos.
- A engenharia é crucial para resolver problemas, especialmente em projetos complexos.
- Projetos complexos, como edifícios inteligentes, requerem a colaboração de diversos profissionais e boas práticas de engenharia.



Engenharia de Software

> **O desenvolvimento de software deve submeter-se aos mesmos princípios aplicados nas engenharias tradicionais, sendo um produto distinto em função da sua intangibilidade que o associa, muitas vezes, somente aos códigos dos programas de computador. Entretanto, diferentemente de outros produtos de outras engenharias em produção, possui alta volatilidade em função de constantes evoluções na tecnologia e nos requisitos, agregando ao software uma complexidade adicional.**

Tecnologia em Camadas

Camada de Qualidade

Camada de Processo

Camada de Métodos

Camada de Ferramentas

Camadas da Engenharia de Software.

Processo de Software

- Um processo é uma sequência de etapas para criar um produto, como o software, muitas vezes envolvendo uma equipe multidisciplinar.

- A Engenharia de Software oferece vários modelos de desenvolvimento, adaptáveis à complexidade do sistema. A qualidade é essencial para sustentar o processo.

Metodologia de Processo.

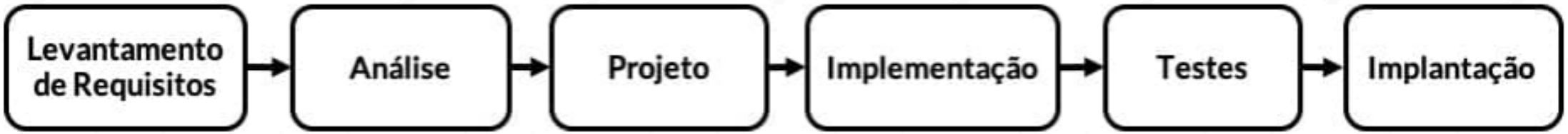
Processo de Software

Atividades de apoio:

Controle e acompanhamento de projeto	**Administração de riscos**	**Garantia de qualidade de software**	**Revisões técnicas**
Medição	**Gerenciamento da configuração de software**	**Gerenciamento da capacidade de reutilização**	**Preparo e produção de artefatos de software**

Processo de Desenvolvimento de Software

- O engenheiro de software deve escolher o processo de desenvolvimento em um projeto de software, como requisito. Primeiro, identifica as atividades necessárias e define seu sequenciamento.



Atividades típicas de um processo de desenvolvimento de software.

Fluxo de Processo

A especificação do processo de desenvolvimento de software envolve definir atividades e seu encadeamento no fluxo de processo ou ciclo de vida.

Fluxo de Processo Linear

Fluxo de Processo Iterativo

Fluxo de Processo Evolucionário

Fluxo de Processo Paralelo

Modelo de Ciclo de Vida Iterativo e Incremental

Gerenciamento de Projeto

> **“**
>
> **Gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto a fim de cumprir os seus requisitos. O gerenciamento de projetos é realizado através da aplicação e integração apropriadas dos processos de gerenciamento de projetos identificados para o projeto. O gerenciamento de projetos permite que as organizações executem projetos de forma eficaz e eficiente.**
>
> **(PMI, 2017)**

Ciclo de Vida do Projeto x Grupos de Processos

- O ciclo de vida do projeto abrange fases genéricas comuns a todos os projetos.
- A escolha de um Fluxo de Processo, como Evolucionário ou Iterativo, determina as etapas do projeto.
- A gestão do ciclo de vida do projeto envolve a execução de atividades de gerenciamento de projetos.

Ciclo de vida do projeto.

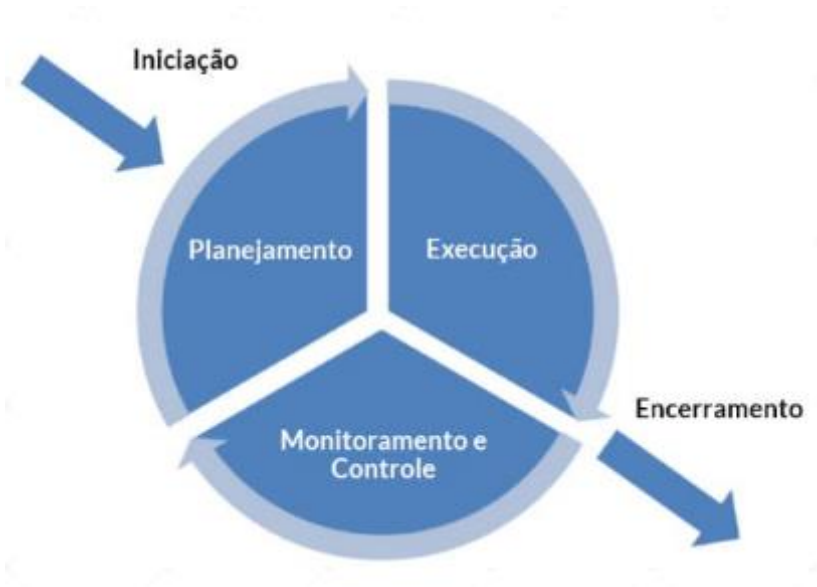
Ciclo de Vida do Projeto x Grupos de Processos

- Cada processo de gerenciamento converte entradas em saídas por meio de técnicas e ferramentas apropriadas.

Processo de Gerenciamento de Projeto.

Ciclo de Vida do Projeto x Grupos de Processos

- A “gestão” de um projeto: Iniciação; Planejamento; Execução; Monitoramento e Controle; e Encerramento.



Grupos de Processos.

Ciclo de Vida do Projeto x Grupos de Processos

- De uma forma simplificada, o gerente de projetos tem que considerar as etapas de gestão, por exemplo, para a etapa INICIAÇÃO:

1. Seleciona os processos que serão utilizados, considerando a complexidade do projeto.
2. Aloca responsáveis por cada processo de acordo com a área de conhecimento.
3. Os responsáveis realizam os processos de acordo com o que preconiza o PMBOK.
4. Os responsáveis geram seus respectivos resultados para cada processo.

- Em seguida, aplica os passos de 1 a 4 anteriores para cada etapa do ciclo de vida do projeto: Planejamento, Execução, Monitoramento e Controle e Encerramento. Em projetos complexos, as etapas podem ocorrer de forma iterativa.

Fluxo de Processo

Os cinco grupos de processos:

Grupo de Processos de Iniciação

Grupo de Processos de Planejamento

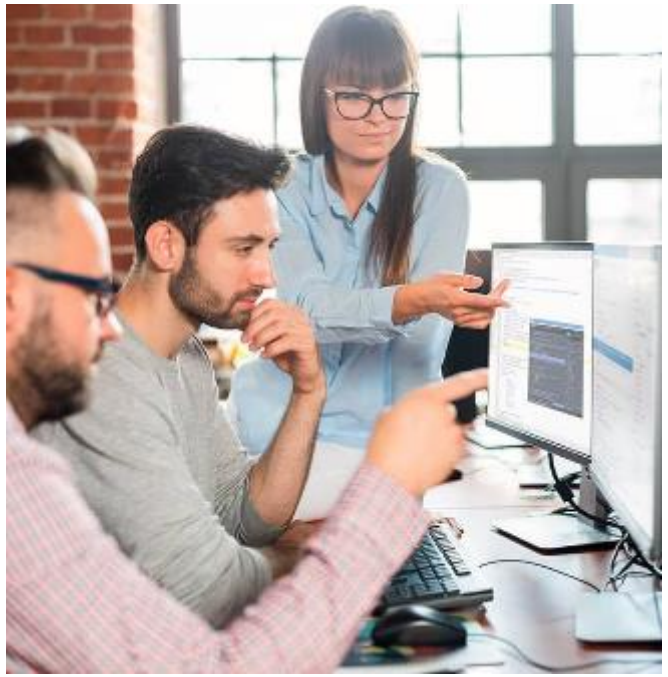
Grupo de Processos de Execução

Grupo de Processos de Monitoramento

Grupo de Processos de Encerramento

Áreas de Conhecimento em Gerenciamento de Projetos

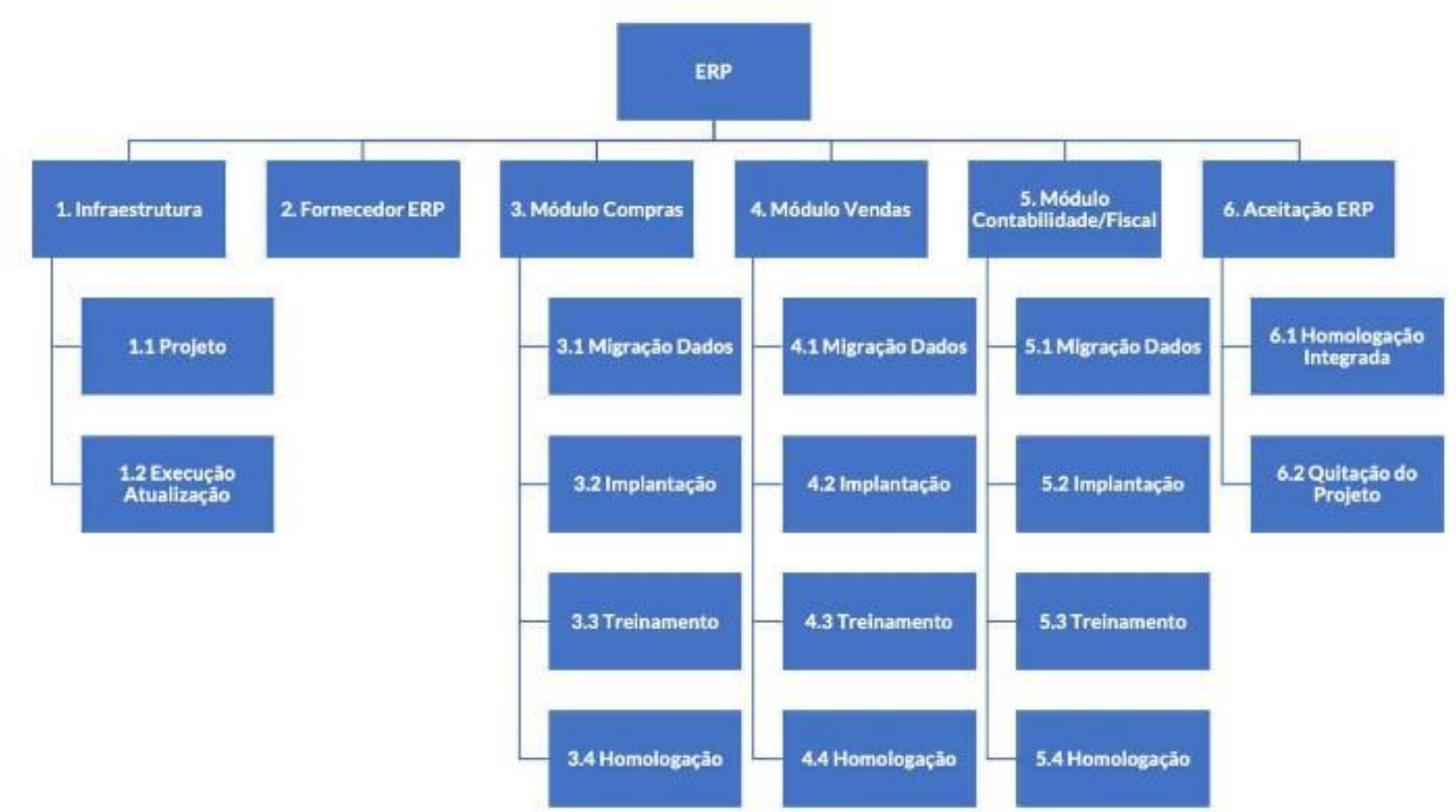
- A equipe de Gerenciamento de Projeto deve incluir especialistas em custos, aquisições, recursos humanos, entre outras áreas necessárias.
- Essa equipe técnica pode lidar com parte da complexidade na gestão de projetos de software.
- Gerenciamento de projetos envolve 10 áreas de conhecimento com processos específicos, exigindo alinhamento da equipe.



Gerenciamento da Integração do Projeto

- A área que integra diferentes áreas de conhecimento é controlada pelo gerente de projetos.
- O gerente age como um maestro, combinando resultados em todas as áreas de conhecimento e mantendo a visão geral do projeto, sendo o único responsável por ele.
- O Termo de Abertura do Projeto autoriza a alocação de recursos.

Gerenciamento do Escopo do Projeto



Exemplo de EAP.

Gerenciamento do Cronograma do Projeto

O cronograma é criado na etapa de planejamento, onde identificamos atividades com base nos pacotes de trabalho da EAP. Para cada atividade, especificamos a duração e os insumos.

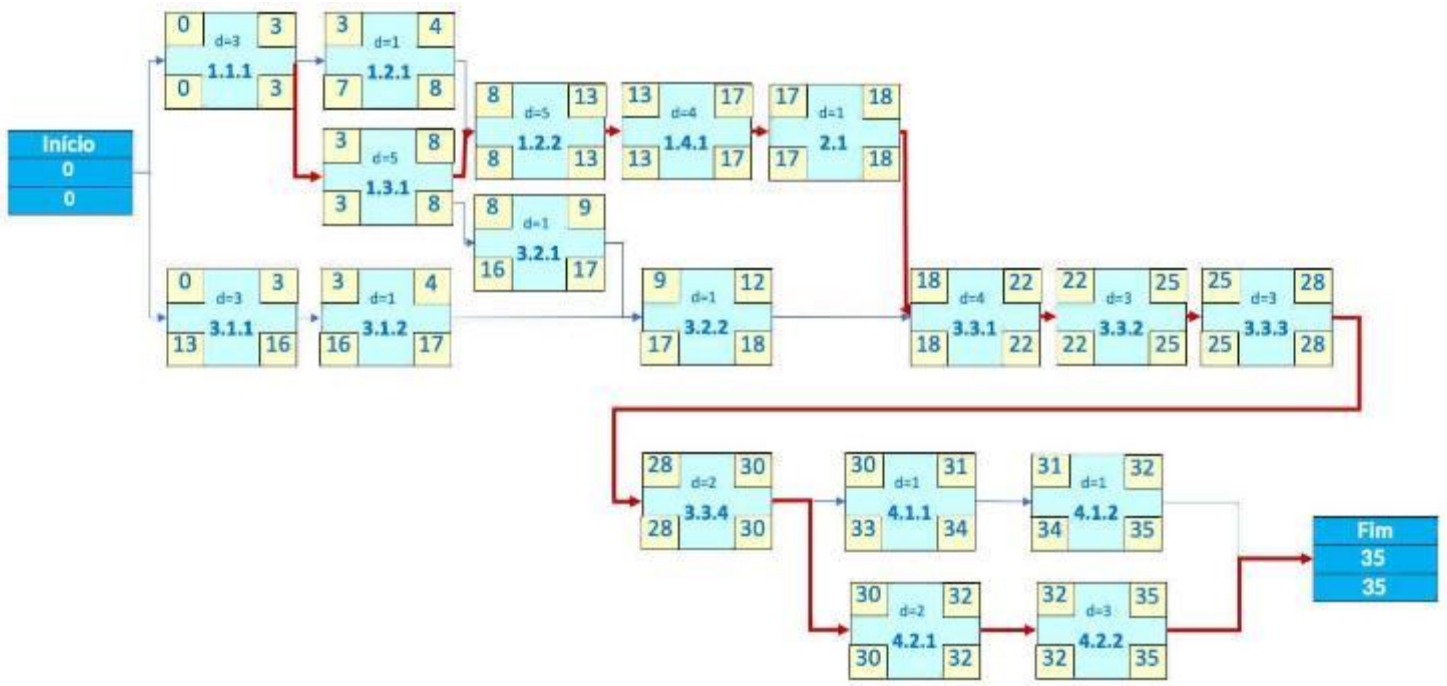
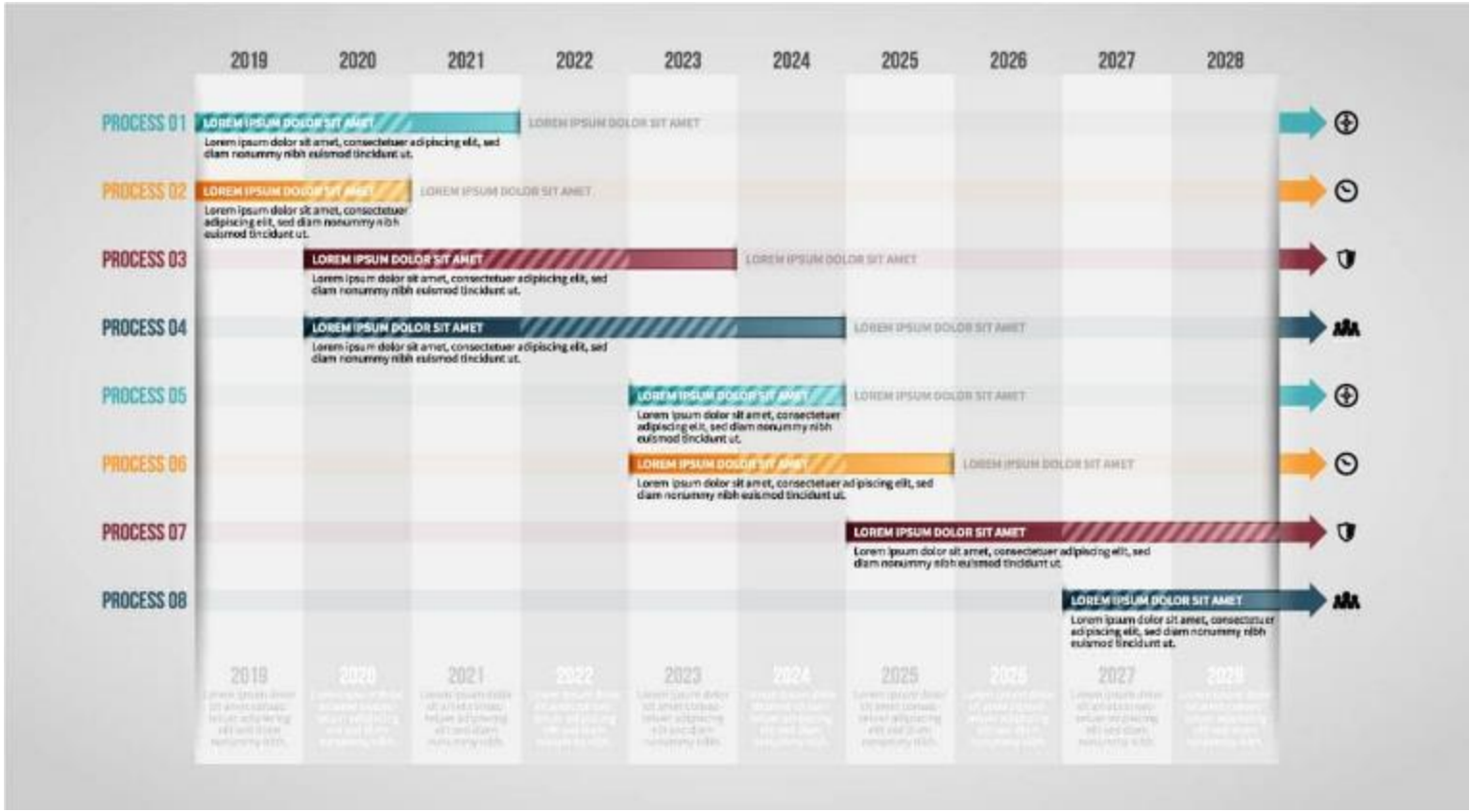


Diagrama de Rede do Cronograma do Projeto.

Gerenciamento do Cronograma do Projeto



Exemplo de Cronograma Detalhado do Projeto.

Gerenciamento do Cronograma do Projeto

Gerenciamento dos Custos do Projeto

Gerenciamento da Qualidade do Projeto

Gerenciamento dos Recursos do Projeto

Gerenciamento das Comunicações do Projeto

Gerenciamento dos Riscos do Projeto

Gerenciamento das Aquisições do Projeto

Gerenciamento das Partes Interessadas do Projeto

Risco

Eventos negativos para entendermos os conceitos:

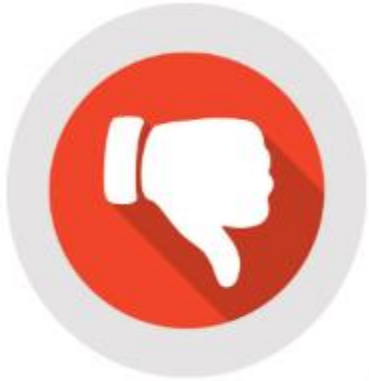
- Ocorrência de um problema de saúde, por causa de um acidente ou ter contraído alguma doença.
- Cancelamento da passagem pela companhia aérea.
- A quantidade de dólar adquirida não ser suficiente.

> **“**
>
> **Um risco é um evento ou condição incerta que, se ocorrer, provocará um efeito positivo ou negativo em um ou mais objetivos do projeto.**
>
> **(PMI, 2017)**

Risco

- Risco não envolve apenas situações negativas; pode ser positivo.

- O risco permite planejar o futuro e evitar que empresas tenham “sustos” nos seus projetos em função de eventos inesperados e, a partir destes, iniciar as devidas tratativas. Em síntese, a área de Gerenciamento de Risco permite a sistematização dessas tratativas em forma de plano.



Cenário Negativo.

Cenário Positivo.

Identificação de Riscos

- Um risco consiste em três elementos: um evento, a probabilidade de ocorrência desse evento (com incerteza) e o impacto associado à ocorrência (com perda).
- No Plano de Gerenciamento de Riscos, a Estrutura Analítica do Risco (EAR) ajuda a definir fontes potenciais de risco no projeto, facilitando a categorização dos riscos.
- A Identificação de Riscos necessita de uma compreensão clara da missão, escopo e objetivos do projeto.

Extrato de um exemplo de EAR.

Identificação de Riscos

Segundo Pressman (2016), uma proposta para identificação de categorias genéricas de risco relacionadas com o software inclui:

Tamanho do Produto

Impacto no Negócio

Características do Envolvimento

Definição do Processo

Ambiente de Desenvolvimento

Tecnologia a ser Criada

Quantidade de Pessoas e Experiência

Identificação de Riscos

- A identificação de riscos é um processo iterativo, pois novos riscos podem surgir durante o projeto.

- Na identificação de riscos algumas técnicas podem utilizadas, assim como:
    - Brainstorming;
    - Lista de Verificação;
    - Entrevista.

Identificação de Riscos

Exemplo da Técnica de Entrevista:

1. A alta gerência e o cliente estão formalmente comprometidos em apoiar o projeto?
2. Os usuários estão bastante comprometidos com o projeto e o sistema/produto a ser criado?
3. Os requisitos são amplamente entendidos pela equipe de engenharia de software e clientes?
4. Os clientes foram totalmente envolvidos na definição dos requisitos?
5. Os usuários têm expectativas realistas?
6. O escopo do projeto é estável?
7. A equipe de Engenharia de Software tem a combinação de aptidões adequadas?
8. Os requisitos de projetos são estáveis?
9. A equipe de projeto tem experiência com a tecnologia a ser implementada?
10. O número de pessoas na equipe de projeto é adequado para o trabalho?
11. Todos os clientes e usuários concordam com a importância do projeto e com os requisitos do sistema/produto a ser criado?

Análise Qualitativa dos Riscos

- Tratar riscos pode alocar recursos financeiros, afetando o orçamento do projeto. Portanto, não podemos tratar todos os riscos da mesma forma, devido às limitações de tempo e recursos.
- A Análise Qualitativa dos Riscos prioriza riscos combinando probabilidade e impacto, permitindo avaliar o potencial de influência de cada risco nos resultados do projeto.

Probabilidades

Graus de Impacto

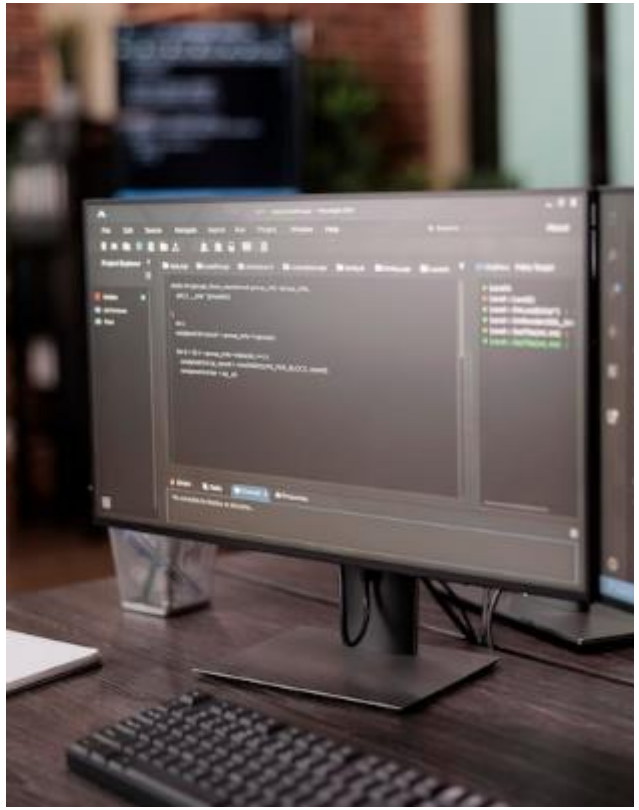
Aplicação

Análise Qualitativa dos Riscos

- Gerenciamento de Risco do Projeto exige especialistas com experiência em lidar com incertezas com base em projetos anteriores.
- O gerente de projetos é um integrador de conhecimentos que precisa entender bem dos processos aplicados nas diferentes áreas.

Análise Quantitativa dos Riscos

- Análise Quantitativa dos Riscos avalia efeitos numéricos de riscos priorizados na Análise Qualitativa.
- Riscos podem impactar resultados do projeto.
- Técnicas usadas: simulações, árvore de decisão, etc.
- Na Engenharia de Software, avaliamos contratação de fábrica de software ou desenvolvimento interno através de árvore de decisão.



Análise Quantitativa dos Riscos

Prioridade	Evento	Custo (R$)
1	Ocorrência de um problema de saúde, por causa de um acidente ou ter contraído alguma doença	10.000,00
2	A quantidade de dólar adquirida não ser suficiente	1.000,00
3	Cancelamento da passagem pela companhia aérea	3.000,00

Resultado da análise quantitativa.

Planejamento de Respostas aos Riscos

Planejar Respostas aos Riscos envolve ações para aumentar oportunidades e reduzir ameaças, priorizando respostas como mitigação ou eliminação do impacto.

Prioridade	Evento	Tratamento	Resposta
1	Ocorrência de um problema de saúde, por causa de um acidente ou ter contraído alguma doença	Eliminação	Contratar o seguro viagem
2	A quantidade de dólar adquirida não ser suficiente	Eliminação	Atualizar o cartão de crédito para uso internacional
3	Cancelamento da passagem pela companhia aérea	Mitigação	Confirmar a reserva em companhia aérea com boas alternativas de voos

Proposta de respostas aos riscos identificados.

Fases do Desenvolvimento de Software

Engenharia de Software

- Abordagem sistemática de decomposição na Engenharia de Software para tratar a complexidade.
- Software é volátil devido a mudanças tecnológicas e requisitos, aumentando sua complexidade.

Engenharia de Software

- A Engenharia de Software é uma tecnologia em camadas.
- Importante destacar que a base da Engenharia de Software é a camada de processo que trata das etapas de desenvolvimento.

Camada	Descrição
Camada qualidade	Garante que os requisitos atendam às expectativas dos usuários.
Camada de processo	Define as etapas de desenvolvimento do software.
Camada de métodos	Determina as técnicas e os artefatos de software.
Camada ferramentas	Estimula a utilização de ferramentas CASE.

Processo de Desenvolvimento de Software

Porque esse processo é iniciado com especificações e modelos com alto nível de abstração e, à medida que o desenvolvimento de software se aproxima da codificação, o nível de abstração diminui, de modo que o código representa o nível mais baixo da abstração ou de maior detalhamento na especificação do software.

1	2	3	4	5	6
Levantamento de Requisitos	Análise	Projeto	Implementação	Testes	Implantação

Engenharia de Requisitos

As etapas de levantamento de requisitos e análise, no processo de desenvolvimento de software, compõem a Engenharia de Requisitos, de modo que essa engenharia está no contexto da Engenharia de Software.

> **“**
>
> **Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Esses requisitos refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema.**
>
> **(Sommerville, 2007)**

Engenharia de Requisitos

A Engenharia de Requisitos inclui as atividades de descobrir, analisar, documentar e verificar os serviços fornecidos pelo sistema e suas restrições operacionais, possuindo um processo próprio

Engenharia de Requisitos

Concepção

Na etapa, o engenheiro de software:

- Estabelece o entendimento do problema.
- Identifica partes interessadas, a natureza da solução desejada.
- Facilita a comunicação entre clientes, usuários e a equipe de projeto.

Levantamento

Nesta etapa, os clientes devem entender os requisitos e dar feedback para evitar erros. A especificação classifica requisitos em três categorias:

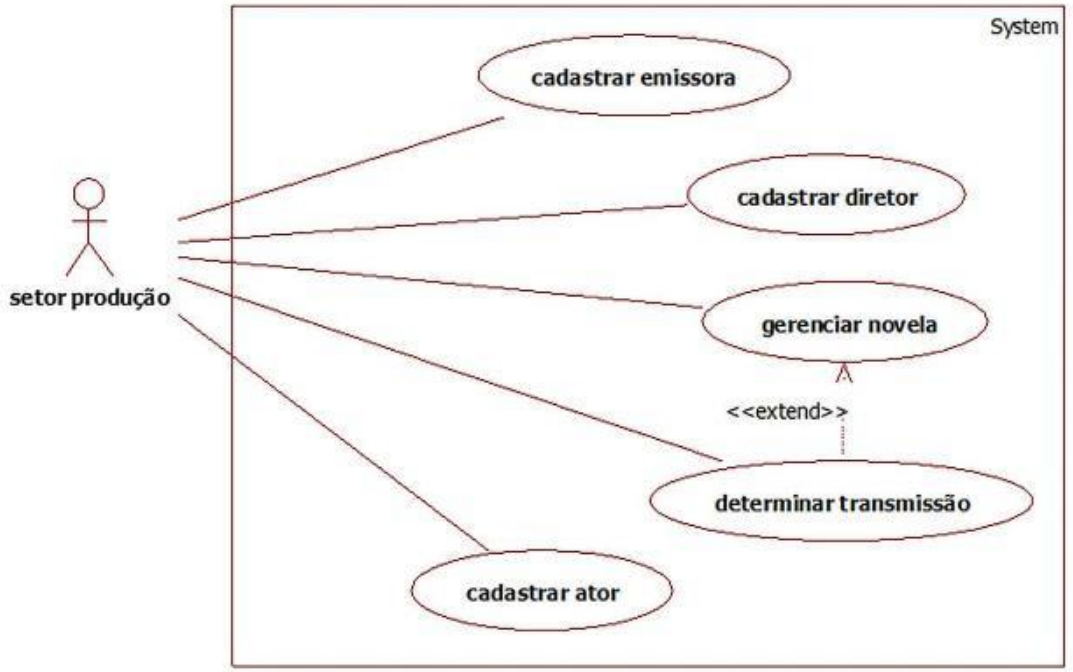
- Funcionais.
- Não funcionais.
- De domínio.

Engenharia de Requisitos

- Engenheiro observa o ambiente de trabalho e tarefas das partes interessadas.
- Entrevista: Diálogo com questões definidas.
- Pesquisa: Aplicação de questionário e análise das respostas.
- JAD: Reuniões de grupo com representantes de usuários e desenvolvedores.
- Brainstorming: Ideias e necessidades discutidas por todos os envolvidos.

Engenharia de Requisitos

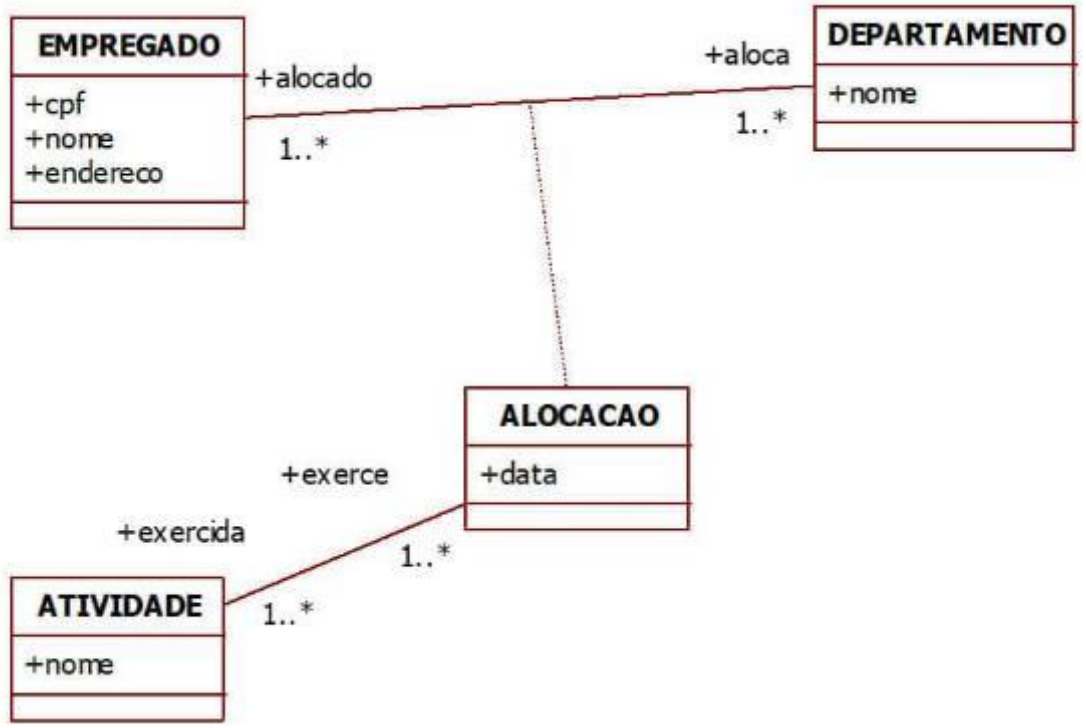
- Na etapa da elaboração, os engenheiros de software realizam um estudo detalhado dos requisitos levantados e, a partir desse estudo, são construídos modelos para representar o sistema a ser construído.
- Elaboração envolve modelagem com cenários a partir de requisitos funcionais.
- Modelagem de casos de uso na UML descreve cenários com diagramas, artefatos gráficos e descrições textuais.
- Criação e refinamento de cenários ajudam a entender como os usuários interagem com o sistema.



Exemplo de diagrama de casos de uso.

Engenharia de Requisitos

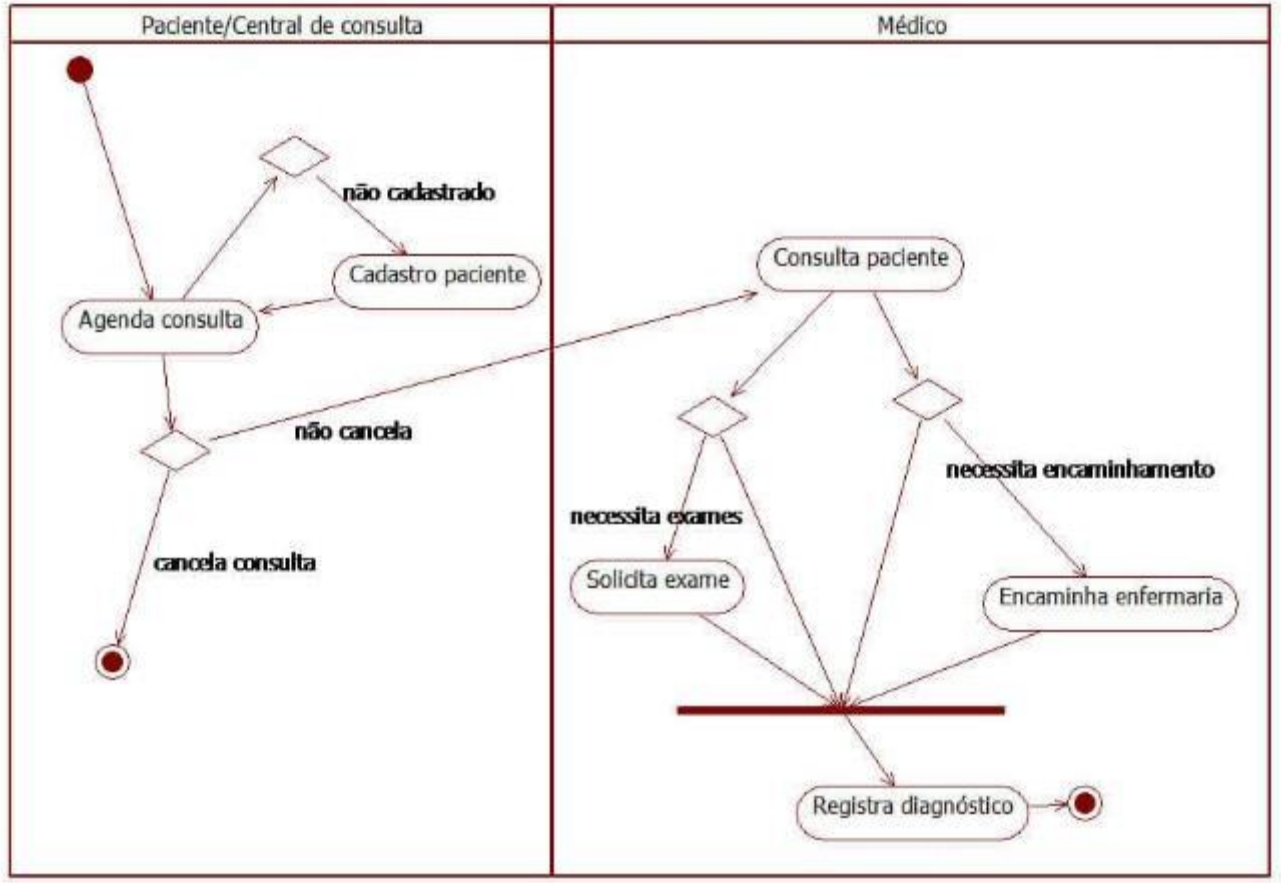
A partir dos casos de uso, podemos identificar as classes de análise que representam os objetos do negócio ou do domínio do problema.



Exemplo de diagrama de classes.

Engenharia de Requisitos

- O modelo de atividades, possui diagramas que podem ser utilizados no mapeamento de processos de negócios, na descrição gráfica de um caso de uso ou na definição do algoritmo de um método.
- O modelo de estados permite representar mudanças de estados significativas e respectivos eventos que causam a referida mudança.



Exemplo de diagrama de atividades.

Engenharia de Requisitos

Negociação

Nesta fase, os envolvidos priorizam e resolvem conflitos nos requisitos, avaliando custos, riscos e prioridades.

Especificação

O engenheiro de software deverá gerar um documento de especificação incluindo todos os requisitos e modelos gerados nas etapas anteriores.

Validação

A validação garante que os modelos atendam às necessidades dos usuários para evitar sistemas insatisfatórios.

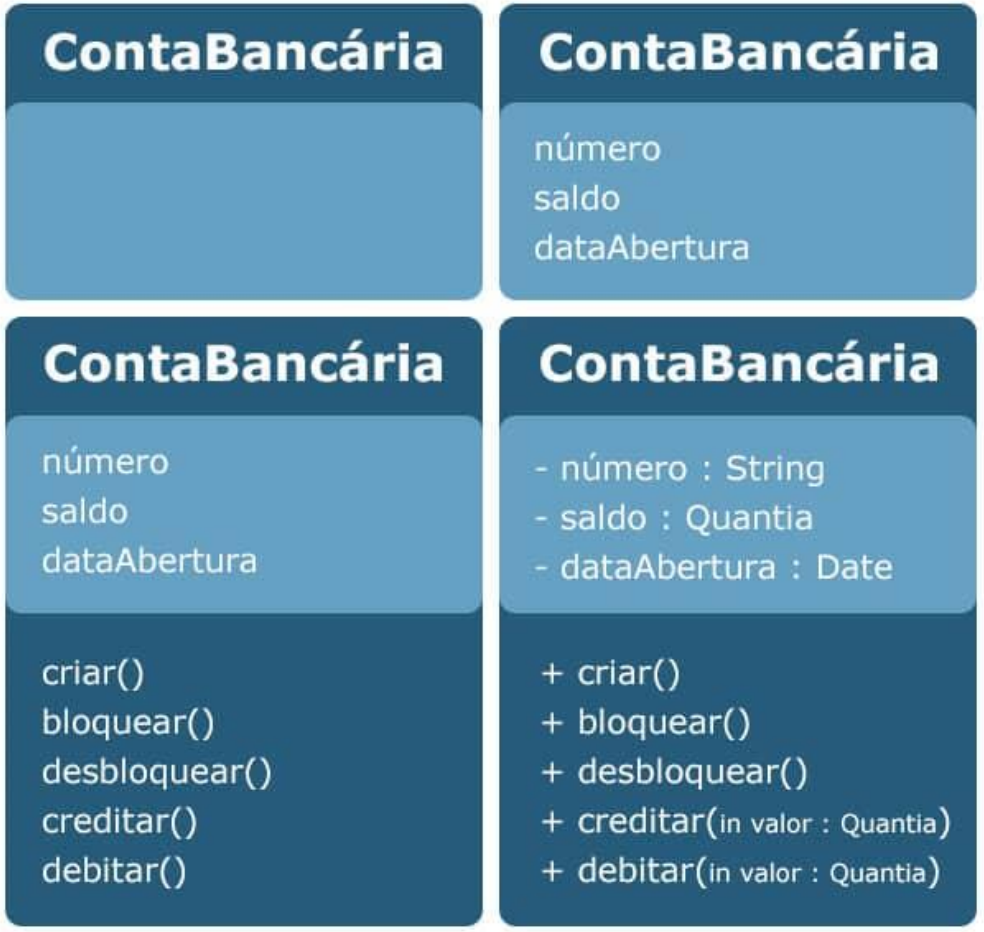
Gestão

Finalizando o processo da engenharia de requisitos, a etapa de gestão permite controlar as mudanças dos requisitos à medida que o projeto evolui.

Projeto de Software

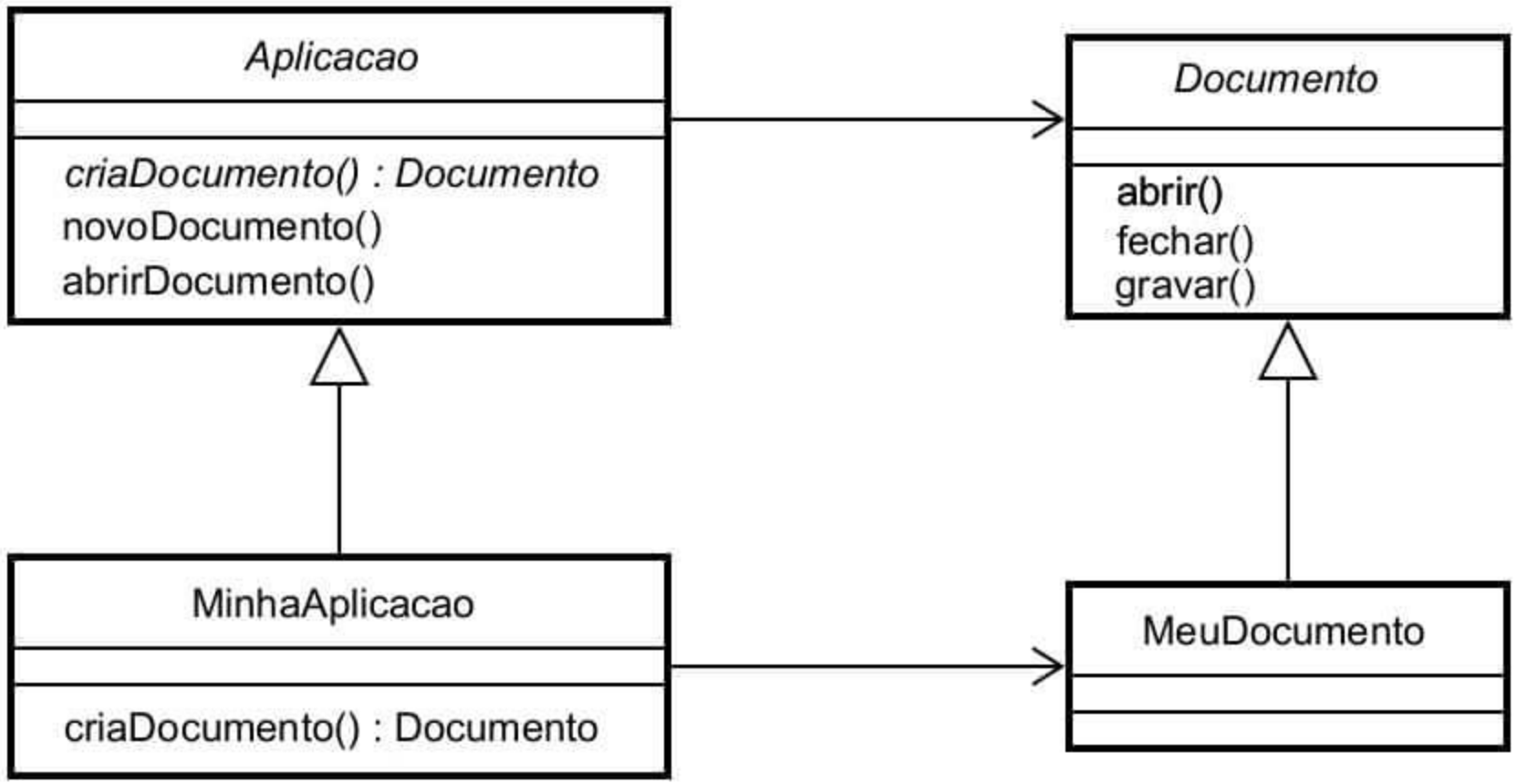
- Modelos-chave na análise incluem casos de uso, classes de análise, atividades e estados.
- O desenvolvimento de software começa com abstração em especificações e modelos de alto nível.
- Na etapa de projeto, ocorre refinamento dos diagramas de análise e criação de novos modelos com menor nível de abstração.

Refinamento dos Aspectos Estáticos e Estruturais do Sistema



Exemplo de refinamentos em uma classe.

Refinamento dos Aspectos Estáticos e Estruturais do Sistema



Exemplo do padrao *Factory Method.*

Detalhamento dos Aspectos Dinâmicos do Sistema

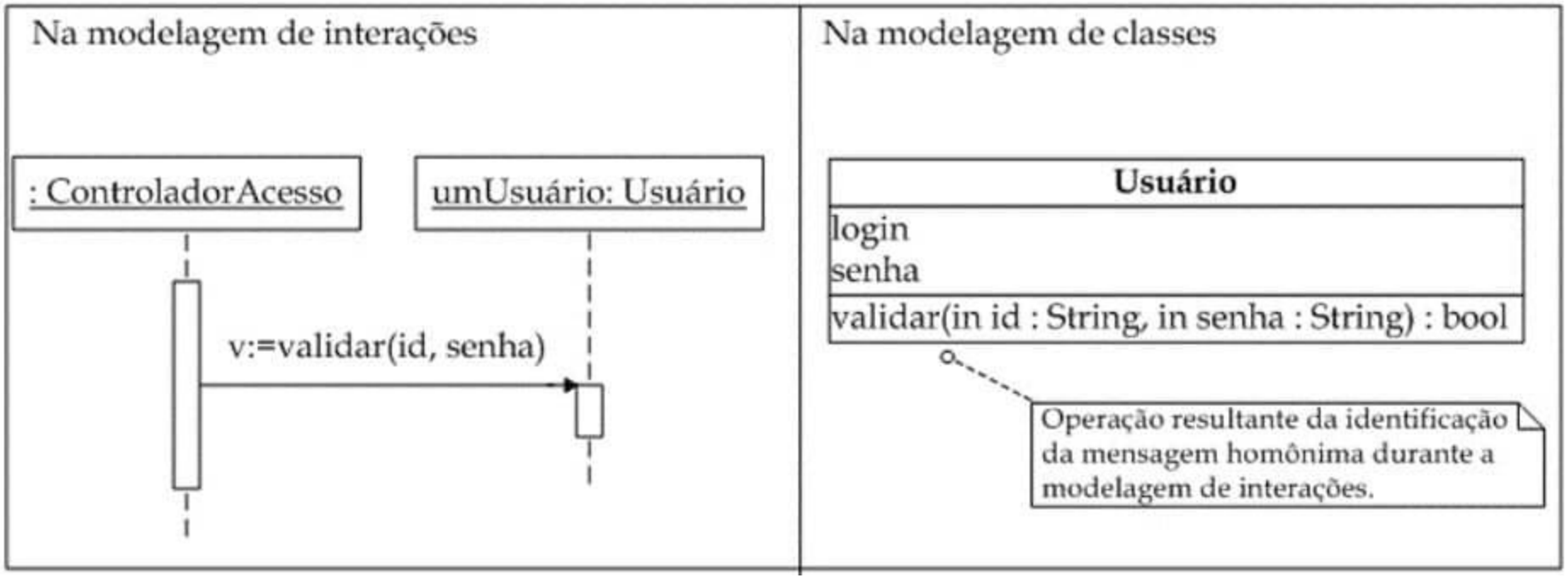
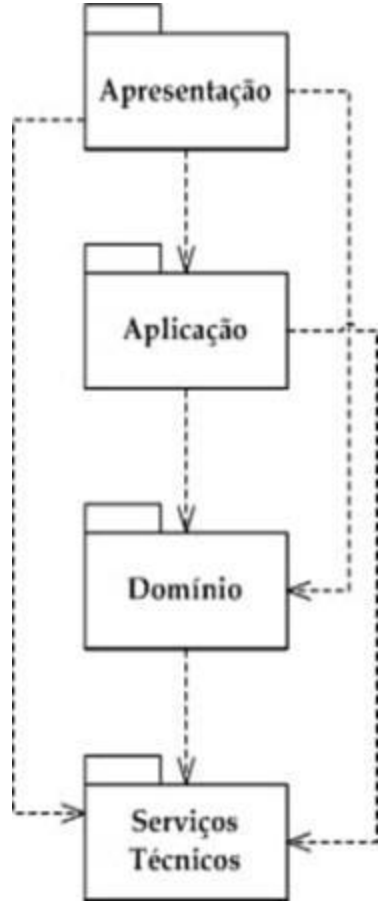


Diagrama sequência *versus* classes de projeto.

Detalhamento da Arquitetura do Sistema

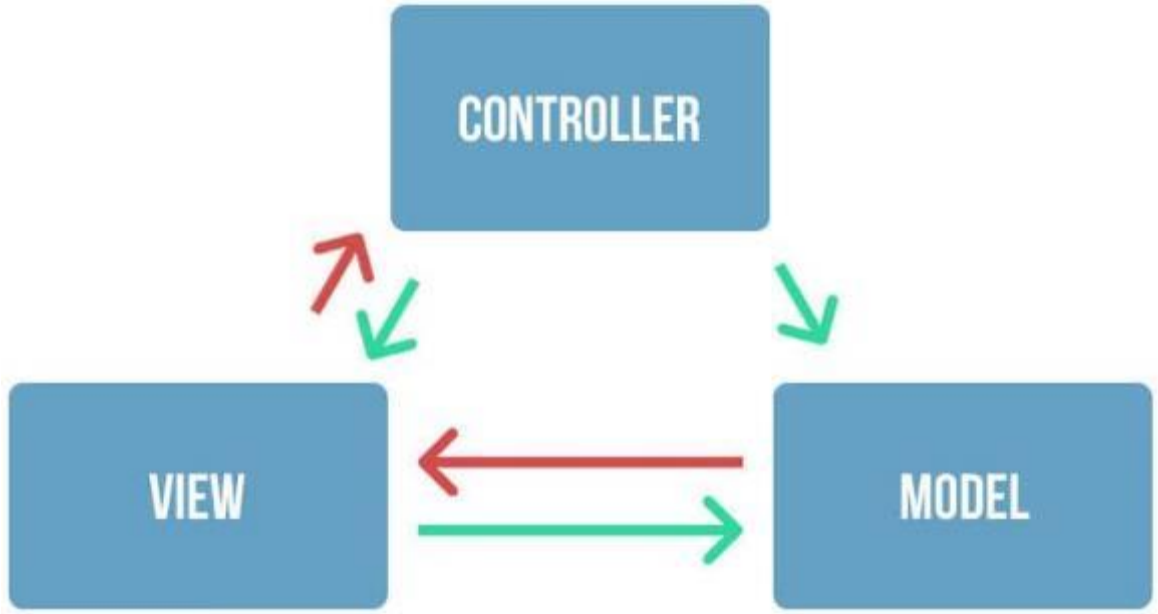
- Fatoração: Decomposição da solução do problema em partes menores.
- Projeto de arquitetura: Identificação de subsistemas com controle e comunicação entre eles.
- Arquitetura em camadas: Camadas de apresentação, aplicação, domínio e serviços técnicos.
- Vantagens da fatoração: Trato da complexidade, reuso, baixo acoplamento e alta coesão.
- Camadas específicas: Apresentação, aplicação e domínio com funções definidas.



Camadas de Software.

Detalhamento da Arquitetura do Sistema

- Padrões de arquitetura: Como o MVC (Model-View-Controller) para reúso e separação de conceitos em três camadas.
- MVC: Controlador atualiza o modelo e a visão exibe dados.
- Modelo: Mantém dados e notifica mudanças.
- Visão *(view)*: Exibe dados.
- Definição lógica e física da arquitetura: Incluindo a infraestrutura de hardware.



Arquitetura MVC.

Detalhamento da Arquitetura do Sistema

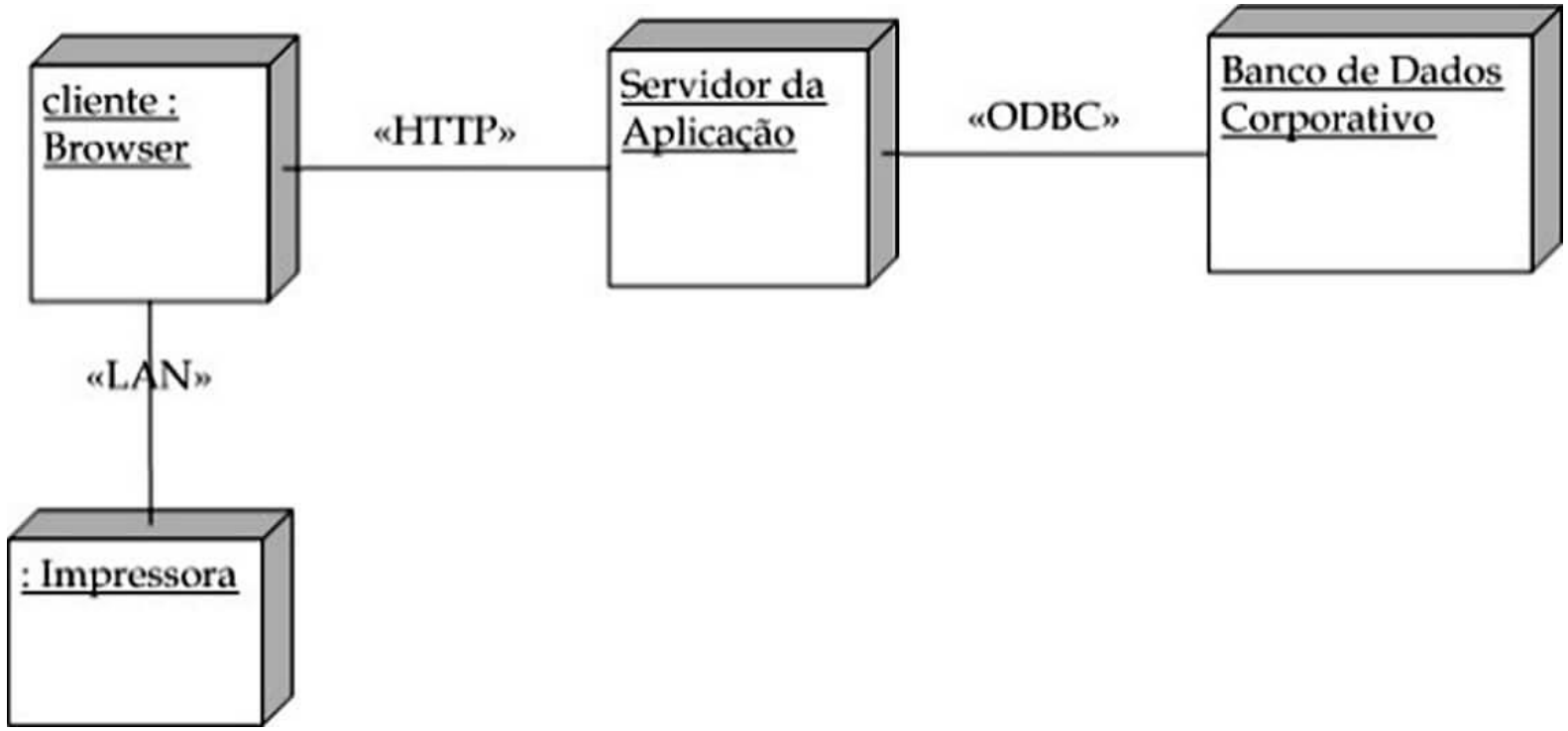


Diagrama de implantação.

Detalhamento da Arquitetura do Sistema

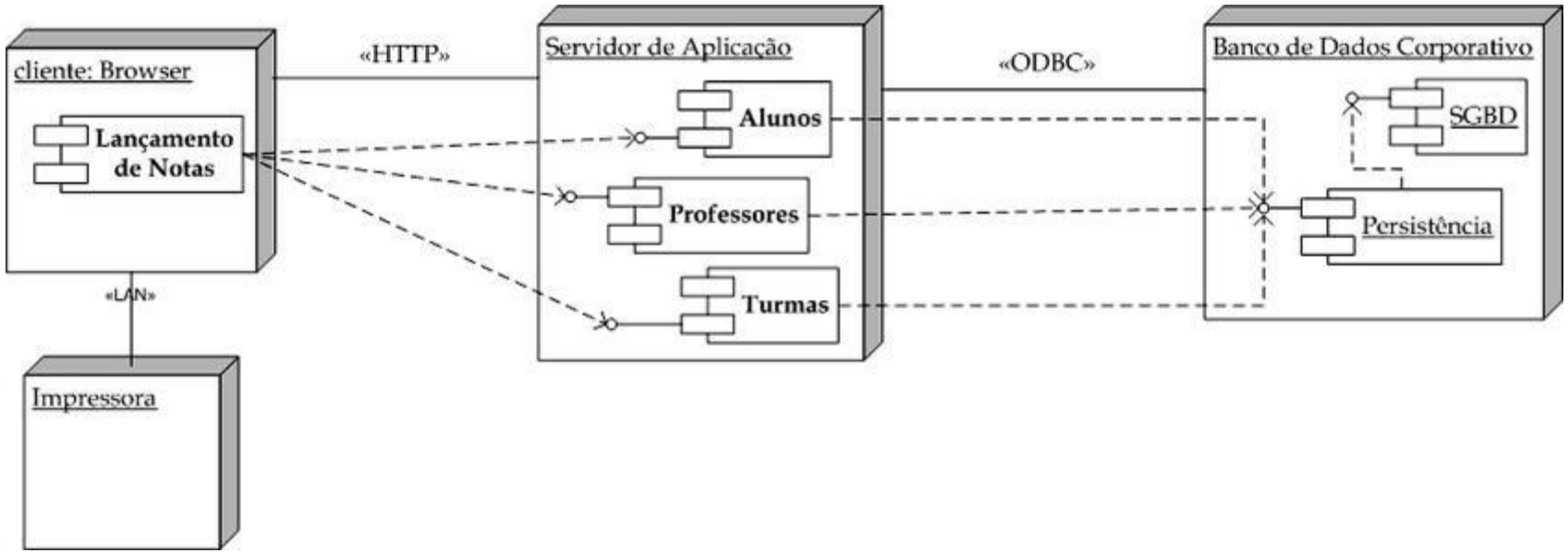
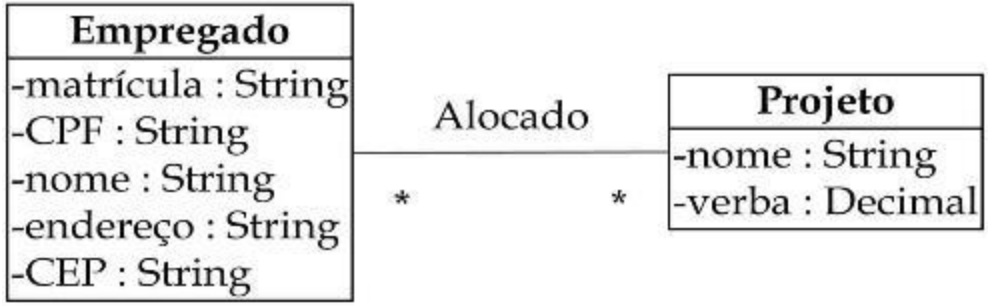


Diagrama de componentes embutidos.

Mapeamento Objeto-Relacional

O padrão da indústria é orientado a objetos, enquanto o padrão de banco de dados é relacional. Para integrá-los, usamos o mapeamento objeto-relacional, transformando o diagrama de classes em tabelas no modelo lógico de banco de dados.



Exemplo de associação muitos-para-muitos.

```
1  Empregado(id,matrícula,CPF,nome,endereço,CEP,idDepartamento)
2  Alocação(id, idProjeto, idEmpregado)
3  Projeto(id, nome, verba)
```

Resultado do mapeamento muitos-para-muitos.

Mapeamento Objeto-Relacional

Realização do projeto da interface gráfica com o usuário

- Diagrama de casos de uso define funcionalidades e interações com atores externos.
- Usabilidade é crucial, envolvendo o usuário na definição de interfaces com foco em tempo de resposta, recursos de ajuda, erros e acessibilidade.

Implementação

A etapa de "projeto" produz modelos para a implementação do software, assim como plantas em Engenharia Civil. A etapa de "implementação" traduz esses modelos em código executável usando linguagens de programação. Programadores lidam com desafios, incluindo requisitos não funcionais como linguagens, frameworks de persistência e qualidade.

> **“**
>
> **A maior causa da crise do software é que as máquinas tornaram-se várias ordens de magnitude mais potentes! Em termos diretos, enquanto não havia máquinas, programar não era um problema; quando tivemos computadores fracos, isso se tornou um problema pequeno, e, agora que temos computadores gigantescos, programar tornou-se um problema gigantesco.**
>
> **(Pressman, 2016)**

Qualidade de Software

- A complexidade e o tamanho das especificações afetam projetos de software.
- A qualidade está alinhada com requisitos e visa a satisfação do cliente.
- Qualidade abrange as dimensões do processo e do produto.
- Foco na qualidade do produto em relação aos testes de software.

Qualidade de Software

A norma ISO 9126 identifica seis atributos fundamentais de qualidade para o software:

Funcionalidade	**Confiabilidade**	**Usabilidade**
Eficiência	**Facilidade de Manutenção**	**Portabilidade**

Teste de Software

- Testes sistemáticos durante o desenvolvimento garantem a qualidade do software.
- Teste é um processo para identificar erros, especialmente em softwares complexos.
- Softwares mal testados podem causar prejuízos, erros financeiros e más decisões.
- Testes de software, dinâmicos e automatizados, simulam vários cenários de uso.

Atenção!

Quanto mais cenários simulados, maior o nível de validação que obtemos do produto, caracterizando maior nível de qualidade do software desenvolvido.

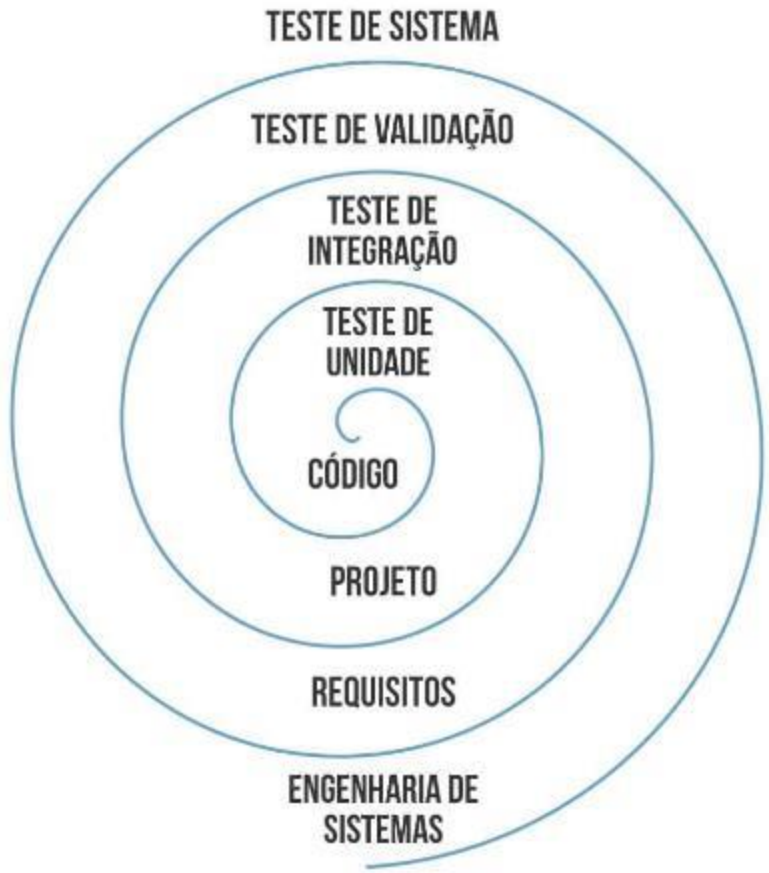
Teste de Software

- Testes avaliam a qualidade do software em várias dimensões.
- Estratégia de codificação progressiva representada em espiral.

Teste de Unidade

Teste de Integração

Teste de Validação



Estratégia de Teste.

Teste de Software

- Planejamento de testes começa com cenários de casos de uso.
- Analista de teste detalha cenários em casos de testes.
- Procedimentos de aceite ocorrem em cada ciclo de implementação.
- O aceite reduz riscos ao detectar falhas antes da implantação.
- Aceite dividido em Teste Alpha, Teste Beta e Aceite Formal.

Teste de Software

Aceite Teste Alpha

Os usuários finais são convidados a operar o software dentro de um ambiente controlado, sendo realizado na instalação do desenvolvedor.

Aceite Teste Beta

Os usuários finais são convidados a operar o produto utilizando suas próprias instalações físicas, ou seja, o software é testado em um ambiente não controlado pelo desenvolvedor.

Aceite Formal

Trata-se de uma variação do Teste Beta, cabendo aos próprios clientes e usuários determinarem o que deverá ser testado e validar se os requisitos foram adequadamente implementados.

Teste de Sistema

Após ser validado, o software deve ser combinado com outros elementos do sistema, tais como hardware, base de dados etc.

Teste de Recuperação

Teste de Segurança

Teste por Esforço

Teste de Desempenho

Teste de Disponibilização

Implantação

A implantação abrange a transferência do sistema da equipe de desenvolvimento para os usuários finais e inclui diversas atividades.

A produção de releases externos do software

A embalagem do software

Distribuição do software

Instalação do software

Prestação de ajuda e assistência aos usuários

Gestão de Configurações *versus* Implantação

Cenário 1

Terminar a versão 2 com o defeito corrigido e liberá-la para produção.

Cenário 2

Realizar a manutenção, eliminando o defeito na versão 1, liberá-la para produção e realizar a devida alteração na versão 2 em desenvolvimento.

- Cenário 2 é a melhor opção, apesar do impacto temporário.
- Estudo de caso destaca controle de alterações e versões.
- Gestão de configuração gerencia mudanças no desenvolvimento.

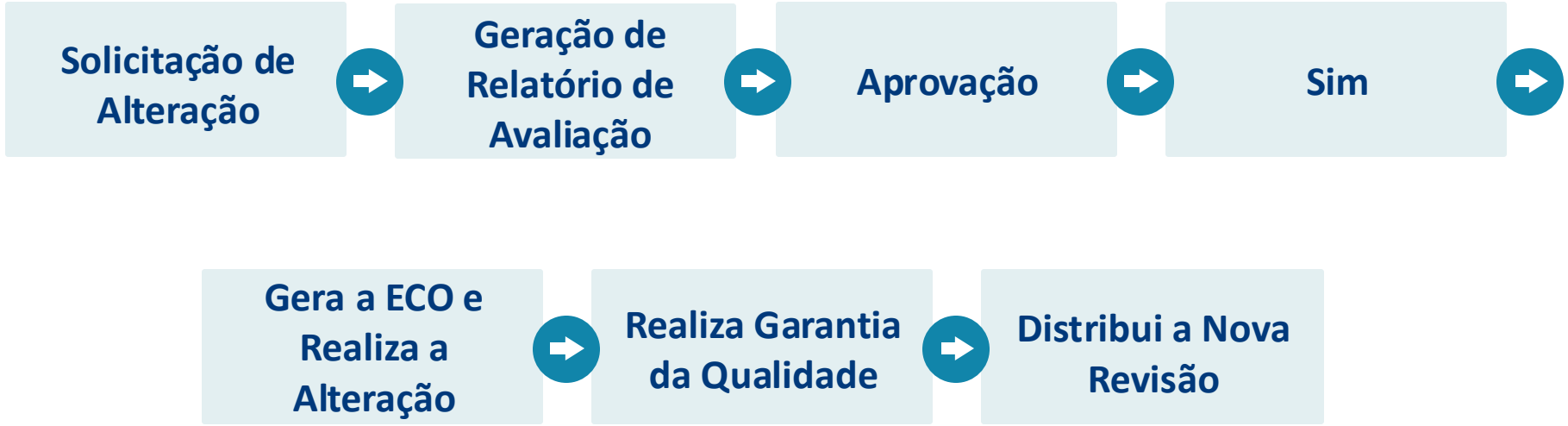
Gestão de Configurações *versus* Implantação

- O processo deve estar ajustado à complexidade do software para evitar erros na produção.
- A falta de controle de alterações em projetos complexos pode causar problemas.
- Controle de versões, incluindo gerenciamento de releases, é importante no gerenciamento de configuração.

1	2	3	4	5
Arquivos de Configuração	Arquivos de Dados	Um programa de Instalação	Documentação eletrônica ou em papel	Empacotamento e publicidade associada

Gestão de Configurações versus Implantação

O processo de controle de alterações:



Manutenção

- A manutenção é a etapa mais longa do ciclo de vida do software.
- Envolve correção de defeitos não identificados antes, aprimoramento de subsistemas e adição de novas funcionalidades.
- Problemas com erros e solicitações de mudanças surgem na fase de manutenção.
- Mobilidade da equipe de desenvolvimento é um desafio, pois membros podem ser substituídos.
- Cumprimento adequado das etapas do processo de desenvolvimento é crucial para garantir a manutenibilidade do software.

Manutenção versus Reengenharia de Software

- A reengenharia é um processo de reconstrução de software para melhorias.
- Dois níveis de reengenharia: de processos de negócio e de software.
- O software visa agregar valor à organização através da automação de processos.
- No segundo cenário, aplicamos a reengenharia de software, o que extrapola os objetivos da etapa manutenção

Cenário 1

Que exige a construção de um novo software.

Cenário 2

Que permite a modificação de um software existente.

Modelos de Processos de Desenvolvimento de Software

Engenharia de Software

A Engenharia de Software é uma tecnologia em camadas

Camada Qualidade

Garante o cumprimento dos requisitos que atendem às expectativas dos usuários.

Camada de Processo

Determina as etapas de desenvolvimento do software.

Camada de Métodos

Define técnicas de levantamento de requisitos e os artefatos gerados, como modelos de casos de uso ou de classes.

Camada Ferramentas

Promove o uso de ferramentas CASE para criar artefatos e gerar código automaticamente, entre outras aplicações.

Processo de Desenvolvimento de Software

- O desenvolvimento de software começa com alta abstração e diminui à medida que se aproxima da codificação.
- Atividades típicas do processo de desenvolvimento de software incluem especificações e modelos.
- As atividades genéricas são comuns a todos os processos, com ênfase variando em diferentes modelos.

Atividades Típicas de um Processo de Desenvolvimento de Software.

Modelos de Processos Prescritivos

- A Engenharia de Software usa modelos de processo para sistematizar o desenvolvimento de software.
- Modelos de processos incluem atividades, fluxos de processos e artefatos.
- A diferença entre os modelos está na ênfase dada a cada atividade e no encadeamento delas.
- Modelos de processos prescritivos fornecem metodologias, tarefas e mecanismos de controle.

Modelos de Processos Prescritivos

Modelo em Cascata

- O modelo em cascata é uma abordagem sequencial, historicamente usado na era do desenvolvimento estruturado.
- Projetos raramente seguem um fluxo sequencial devido à volatilidade dos requisitos.
- A abordagem assume que o cliente pode especificar todos os requisitos antecipadamente, o que pode causar erros.
- A divisão do projeto em fases distintas pode ser inflexível, com o software disponível apenas no final.

Modelos de Processos Prescritivos

Modelos de processos incremental e Modelos de processos evolucionário

- Ambos os modelos dividem o desenvolvimento em ciclos iterativos.
- Cada ciclo entrega um conjunto limitado de funcionalidades, agregando valor imediatamente em meio a restrições de prazo e requisitos voláteis.

Modelos de Processos Prescritivos

Modelos de processos incremental e Modelos de processos evolucionário

Modelo Incremental

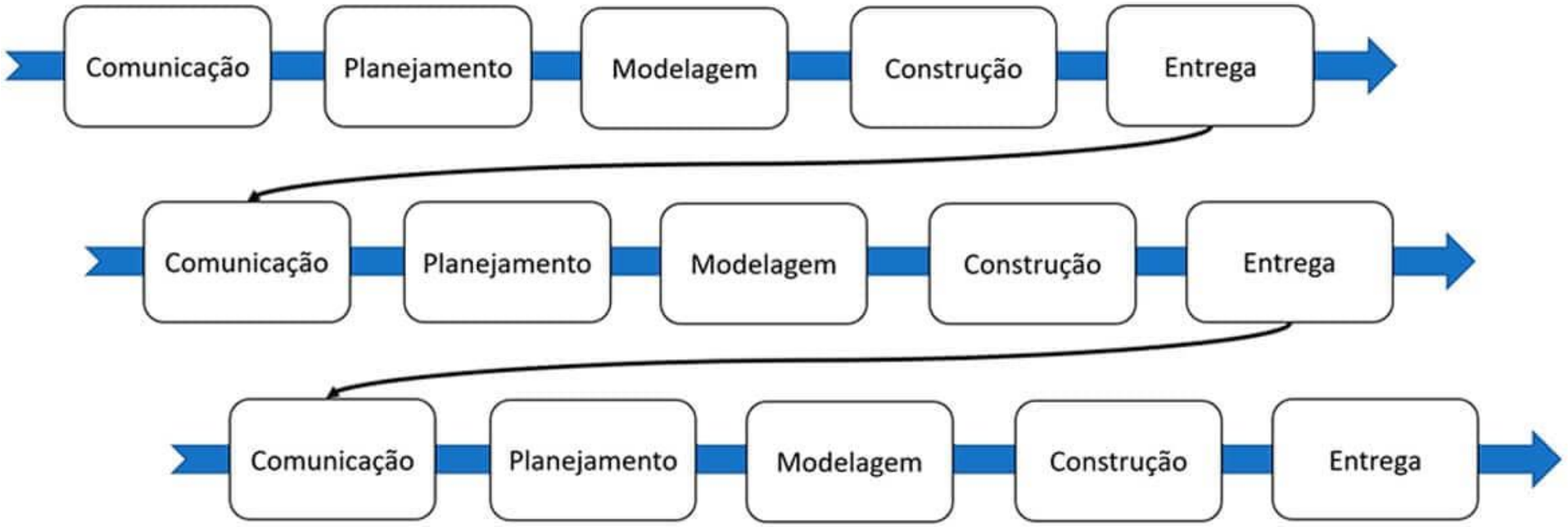
Neste modelo, ocorre a entrega de um produto essencial, estando as demais iterações bem definidas.

Modelo Evolucionário

Neste modelo, as versões são iterativamente desenvolvidas à medida que novos requisitos são compreendidos e definidos em cada iteração.

Modelos de Processos Prescritivos

Modelos de processos incremental e Modelos de processos evolucionário

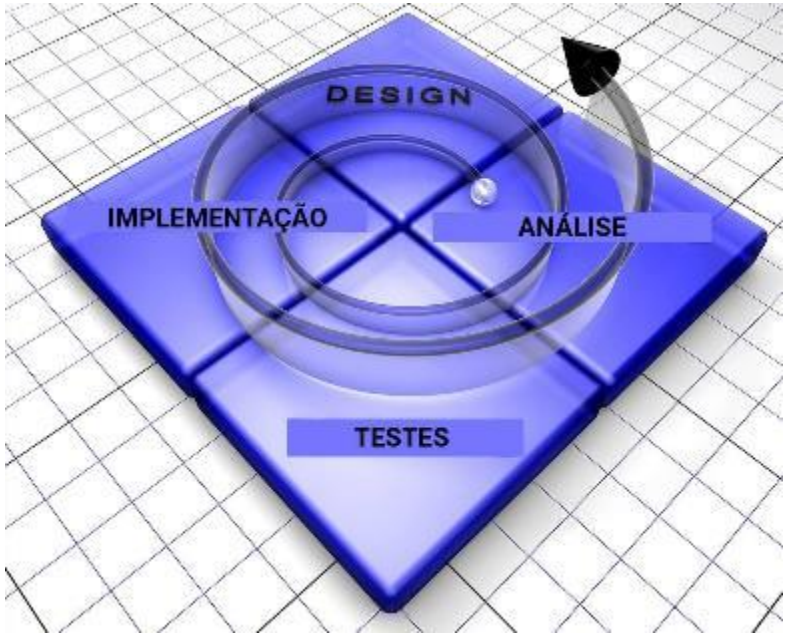


Modelo Incremental.

Modelos de Processos Prescritivos

Prototipação

- Barragem: construção com protótipo em escala reduzida.
- Prototipação: desenvolvedor interage com usuário, projeto rápido.
- Software gerado: primeira versão do protótipo.
- Reconstruída: normalmente necessária após validação de requisito.
- Refinada e integrada: possível para a mesma versão do protótipo.
- Prototipação: pode ser usada como modelo de processo em problemas de baixa complexidade.



O paradigma prototipação.

Modelos de Processos Prescritivos

Modelo Espiral

Modelo Espiral proposto por Barry Boehm em 1988. Cada quadrante corresponde a uma etapa de desenvolvimento, cada loop representa fase do processo de software.

Primeira Etapa

A fase de "Analysis" envolve definir objetivos, considerar alternativas, identificar riscos e restrições.

Segunda Etapa

Na fase de "Evaluation," são avaliadas alternativas, identificados riscos e realizada análise de risco.

Terceira Etapa

Development, ocorre o desenvolvimento do produto.

Quarta Etapa

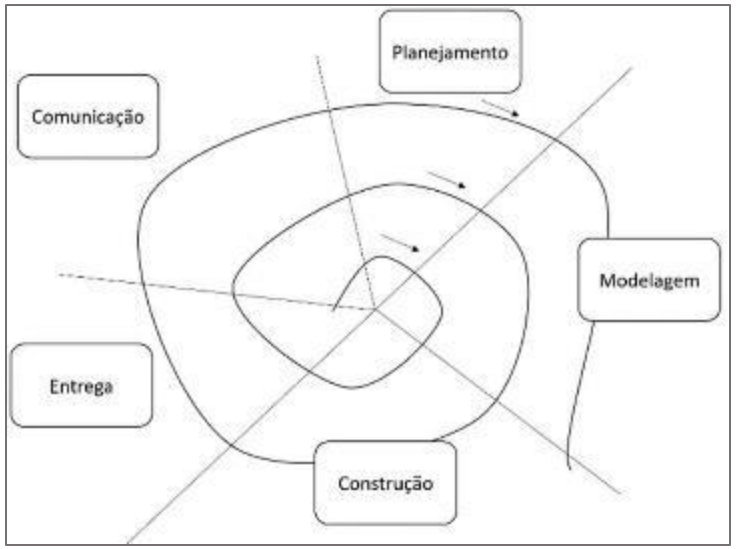
Planning, o produto é avaliado, sendo realizado o planejamento para início de um novo ciclo

Modelos de Processos Prescritivos

Modelo Espiral

Podemos fazer uso da abstração de forma a visualizarmos o modelo espiral como um metamodelo, tendo como exemplo o proposto por Pressman (2016), na qual a espiral do modelo é dividida nas atividades genéricas.

O modelo espiral.



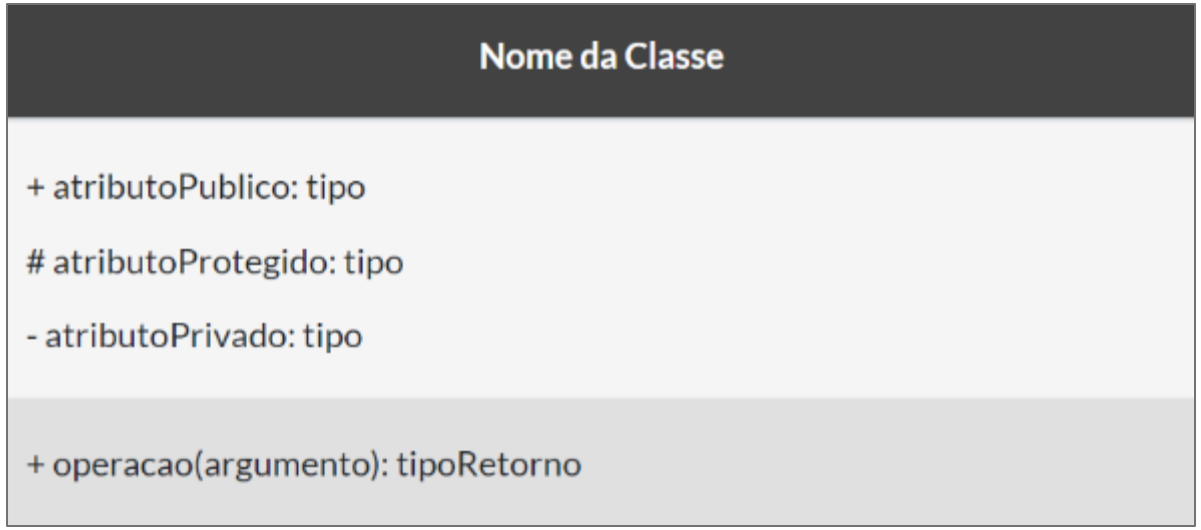
O modelo espiral.

Unified modeling language (UML)

- UML é uma linguagem de modelagem padrão para desenvolvimento orientado a objetos.
- Metodologistas perceberam que a orientação a objetos tratava melhor a complexidade.
- UML surgiu em 1996 como padrão unificador para modelagem de sistemas orientados a objetos.
- Aprovada pelo Object Management Group (OMG) em 1997, tornando-se padrão na indústria de software.
- UML é independente de linguagem de programação e processo de desenvolvimento, envolve criação de diagramas gráficos e descrições textuais.

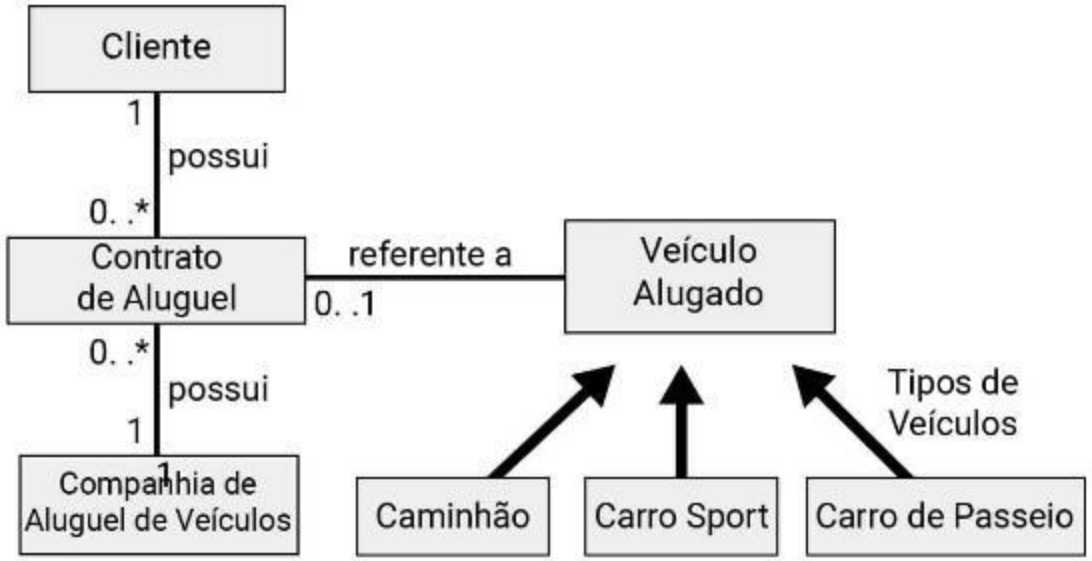
Unified modeling language (UML)

Linguagem envolve sintaxe e semântica. Sintaxe é o padrão de escrita, semântica é o significado. Na UML, a figura abaixo mostra a estrutura da classe, seguindo um padrão da UML para nome, atributos e métodos. A UML define a sintaxe do modelo de classes.



Unified modeling language (UML)

A partir do momento que construímos um diagrama com um conjunto de classes, temos então a formação de uma rede semântica, na qual os objetos e associações representam uma abstração da estrutura estática de certo domínio do problema.



Exemplo de Diagrama de Classes.

Unified modeling language (UML)

Linguagens, como a UML, facilitam a comunicação no desenvolvimento de software. Os diagramas UML abrangem diagramas comportamentais (funcionalidades) e estruturais (dados e componentes). Essa comunicação é essencial entre engenheiros de software, usuários e gerentes de projeto.

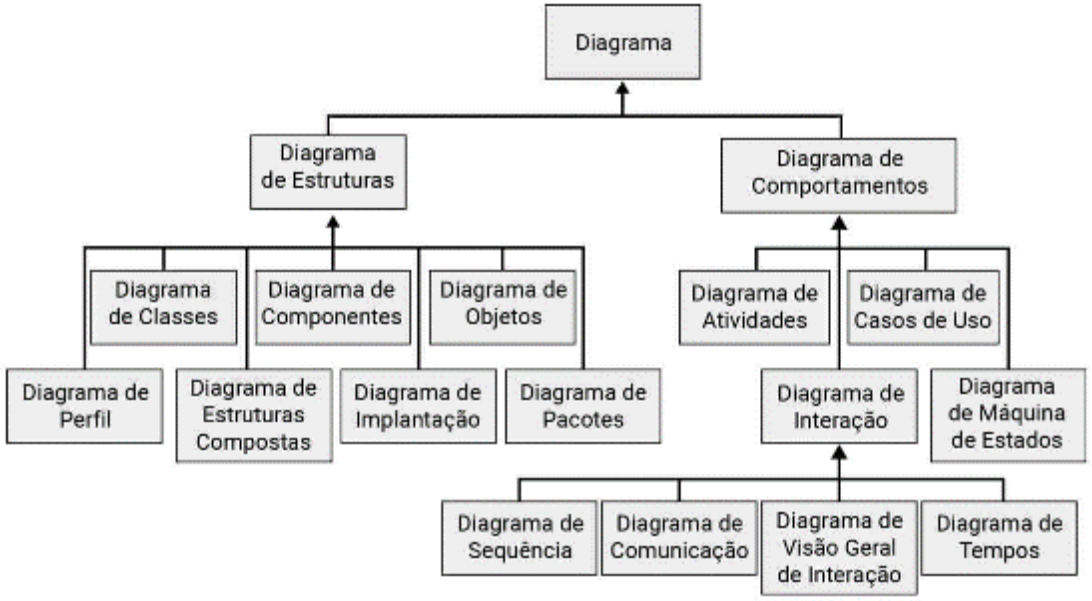


Diagrama da UML.

Processo Unificado

- O Processo Unificado, também denominado de Rational Unified Process (RUP), é um processo de desenvolvimento de software criado pela Rational Software Corporation, empresa fundada pelos criadores da UML, posteriormente adquirida pela IBM.

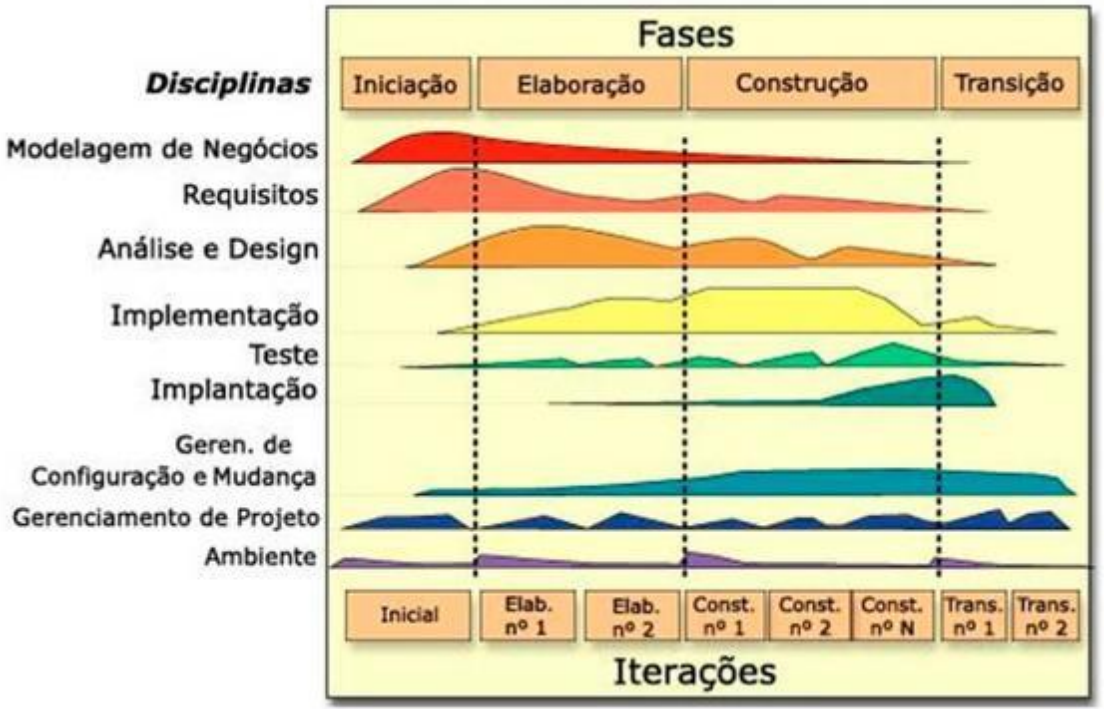
- O RUP é considerado um processo formal que permite ser utilizado na solução de problemas com alta complexidade, entretanto, pode ser customizado para projetos de qualquer escala.

Guiado por casos de uso

Centrado em Arquitetura

Interativo e Incremental

Processo Unificado



O processo unificado.

Processo Unificado

Mais detalhes sobre as etapas do modelo unificado.

Iniciação

Na fase de iniciação, o engenheiro modela o negócio, define requisitos e escopo, enfrentando desafios de viabilidade; o projeto segue adiante ou é encerrado com base nisso.

Elaboração

Na fase de elaboração, refinamos modelos, criamos protótipos e expandimos a arquitetura, minimizando riscos e planejando a construção.

Construção

Na fase de construção, desenvolvemos componentes, finalizamos modelos da elaboração, realizamos testes, incluindo a validação com testes beta pelos usuários em ambiente controlado.

Transição

A fase de transição implica na implantação do software e em testes beta não controlados pelos desenvolvedores. Além disso, envolve a elaboração de manuais, treinamento, instalação e suporte para a versão de produção.

Fluxo de Trabalho

- Atividades transversais do RUP.
- Relacionamento entre fases do projeto e fluxos de projeto.
- Atividades mais intensas em determinadas fases.
- Exemplo: modelagem de negócio na fase de concepção.

O RUP define três grupos de atividades de apoio ao desenvolvimento do software:

Ambiente	**Gerenciamento de Configuração e Mudança**	**Gerência de Projeto**

Desenvolvimento Ágil de Software

- Até meados dos anos 1990, o desenvolvimento de software seguia métodos prescritivos com planejamento detalhado e controle de qualidade.
- Em 2001, o Manifesto Ágil surgiu como alternativa, proposto por 17 profissionais de software em Utah.
- Os métodos ágeis focam no próprio software, usam processos iterativos e adaptam-se a requisitos voláteis.

Indivíduos e interações, mais do que processos e ferramentas.

Software em funcionamento, mais do que documentação abrangente.

Colaboração com o cliente, mais do que negociação de contratos.

Resposta a mudanças, mais do que seguir um plano.

Desenvolvimento Ágil de Software

A entrega de software rápida permite que os clientes validem de forma imediata o incremento, indicando possíveis erros ou alterações de requisitos, bem como descrevendo novos requisitos para as próximas iterações.

Atenção!

Métodos ágeis também enfatizam comunicações em tempo real entre equipe de desenvolvimento e clientes.

Extreme Programming - XP

- A Extreme Programming, ou simplesmente XP, é considerada um processo de desenvolvimento de software ágil, tendo sido proposta por Kent Beck em 1999.

- A XP adota valores que servem como critérios que norteiam as partes interessadas de um projeto, incluindo:

Comunicação

Busca compreender requisitos com pouca documentação e muita interação equipe-cliente.

Simplicidade

Defende soluções simples para reduzir custos de mudanças futuras.

Feedback

Oferece feedback durante testes, permitindo correções com base na validação funcional pelos clientes.

Coragem

Essencial para lidar com riscos de erro, promovendo confiança nos mecanismos de prevenção e proteção.

Respeito

É o valor fundamental que sustenta um projeto de software, sendo crucial para o sucesso.

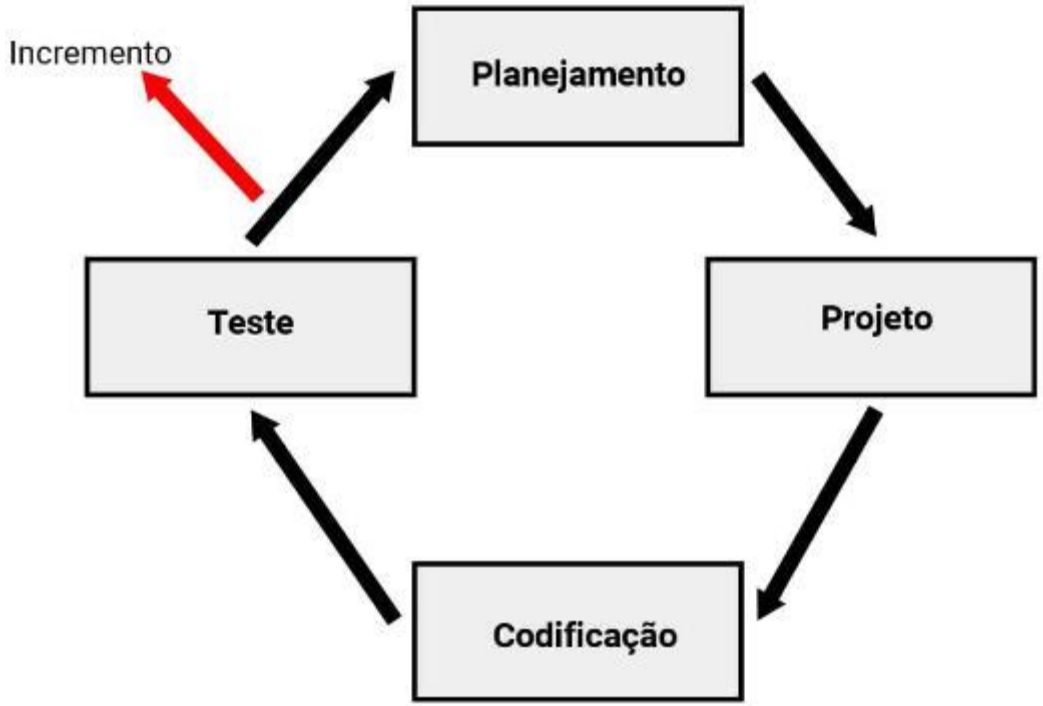
Extreme Programming - XP

A XP possui doze práticas que se enquadram nos valores e princípios

Jogo de Planejamento	**Pequenas Releases**	**Metáfora**	**Projeto Simples**
Cliente no Local de Trabalho	**Semana de 40 horas**	**Programação Pareada**	**Propriedade Coletiva**
Padronização do Código	**Desenvolvimento Orientado a Testes**	**Refatoração**	**Integração Contínua**

Extreme Programming - XP

Processo XP



O processo XP.

O Processo Ágil de Extreme Programming - XP

Veja mais detalhes sobre o processo XP e suas etapas:

Planejamento

Cliente define requisitos, equipe avalia, seleciona e calcula velocidade após a primeira versão para estimar prazos.

Projeto

XP enfatiza simplicidade com cartões CRC, prototipação para complexidade e refatoração para organização do código sem impacto funcional.

Codificação

XP incentiva programação em par com revisão de padrões e integração contínua para detecção precoce de erros.

Teste

Na XP, equipes dividem histórias em tarefas com testes de unidade, automação, integração contínua, e testes de aceitação são definidos pelos clientes para funcionalidades visíveis.

Metodologia Ágil Scrum

- Scrum originou-se como um método de gerenciamento de produtos fora da indústria de software.
- Jeffrey Sutherland e Ken Schwaber desenvolveram o Processo de Desenvolvimento Scrum nos anos 90, amplamente utilizado no desenvolvimento de software.
- Scrum é aplicado sistematicamente, combinando valores ágeis e princípios do manifesto, ideal para projetos complexos com requisitos evolutivos.
- O Scrum possui três pilares fundamentais.

Transparência

Inspeção

Adaptação

Papéis do Scrum

Uma equipe Scrum é composta pelo:

Product Owner (Dono do Produto)

Atua como cliente, define requisitos, comunica a visão do projeto e esclarece dúvidas.

Scrum Master

É o responsável por assegura a adesão às regras do Scrum, facilita a resolução de problemas e mantém as regras do processo, esclarecendo papéis da equipe.

Scrum Team (Equipe Scrum)

É multidisciplinar e inclui especialistas para desenvolver o produto independentemente de membros externos, como programadores, administradores de banco de dados e web designers.

Artefatos Scrum

Os Artefatos Scrum são concebidos com a finalidade de garantir que as equipes Scrum sejam bem-sucedidas na geração de um incremento. Vejamos os principais artefatos:

Product Backlog

O Product Owner mantém uma lista de requisitos e funcionalidades do produto, chamada Product Backlog, que evolui à medida que o produto e seu ambiente mudam.

Sprint Backlog

O Sprint Backlog é um subconjunto de requisitos do Product Backlog para uma iteração, incluindo o plano para atingir a meta da Sprint. A equipe de desenvolvimento pode fazer modificações conforme necessário.

Eventos Scrum

Os eventos no Scrum melhoras a deficiência das inspeções e iterações, evitando reuniões extras, graças a um tempo definido *(timebox)* para um planejamento eficaz.

Sprint

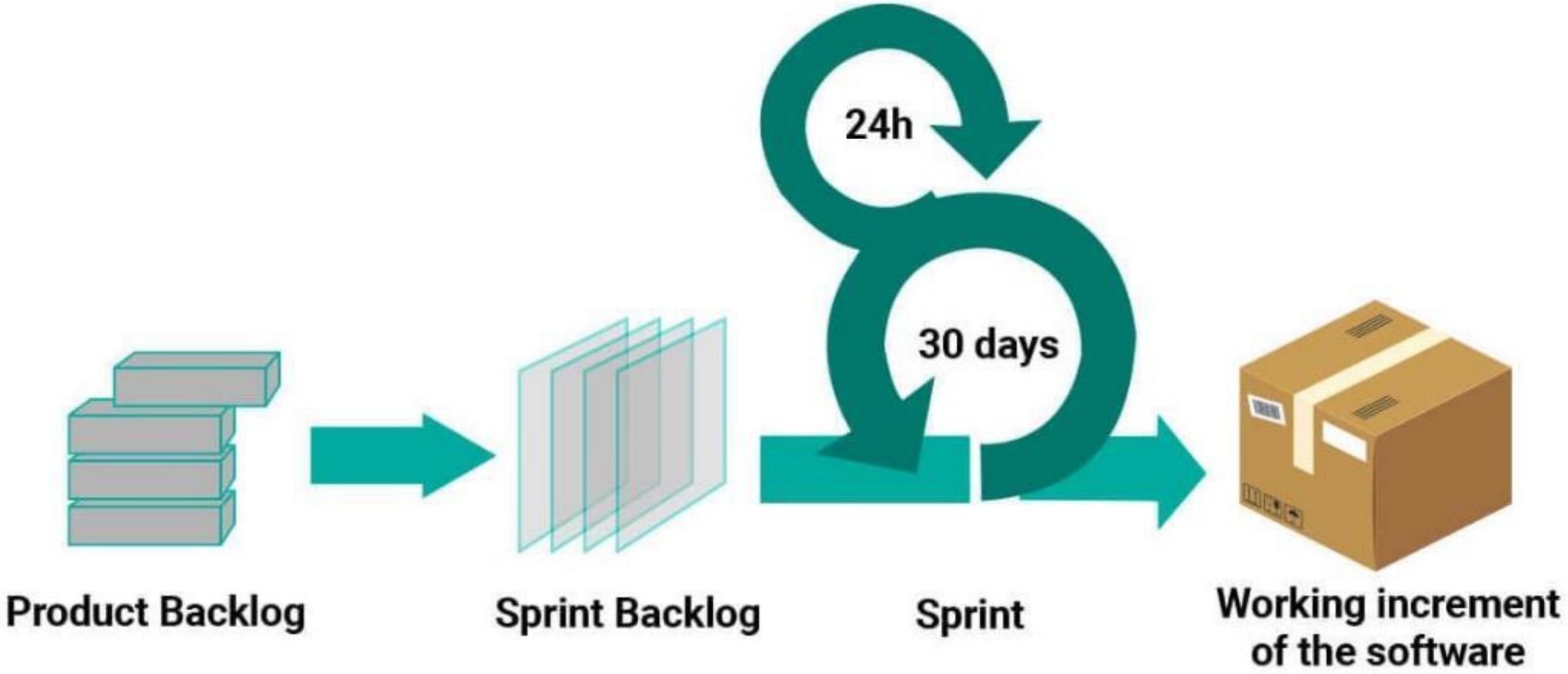
Reunião do Planejamento do Sprint

Reunião Diária do Sprint (Daily Scrum

Revisão do Sprint

Retrospectiva do Sprint

Fluxo de Processo do Scrum



Fluxo de Processo Scrum.

O Processo de Desenvolvimento Ágil Scrum

Os requisitos que definem as funcionalidades a serem entregues são agrupadas pelo product owner no product backlog.

Reunião de Planejamento de Sprint

No início da sprint, ocorre uma reunião de planejamento para definir o sprint backlog.

Desenvolvimento do Sprint

A equipe inicia o desenvolvimento do sprint e realiza reuniões diárias scrum para acompanhar o progresso e prioridades.

Reunião de Revisão de Sprint

A revisão de sprint ajusta o backlog com a equipe scrum, scrum master e product owner.

Reunião de Retrospectiva do Sprint

A reunião de retrospectiva do sprint analisa lições aprendidas, pontos fortes e áreas para melhoria nas próximas iterações.

Processo Unificado Ágil - AUP

- A equipe sabe o que está fazendo. O AUP disponibiliza links de acesso à documentação com muitos detalhes, isso para os que estiverem interessados.
- Simplicidade. Tudo é descrito de forma concisa maximizando a simplicidade.
- Agilidade. O AUP está em conformidade com os valores e princípios da metodologia ágil de desenvolvimento de software.
- Concentre-se em atividades de alto valor. O foco está nas atividades que realmente agregam valor ao cliente.
- Independência de ferramenta. A recomendação é utilizar as ferramentas mais adequadas, que muitas vezes são ferramentas simples.
- Customização. O produto AUP é facilmente customizável às necessidades dos seus usuários.

Processo Unificado Ágil - AUP

O AUP adota uma natureza serial para o que é amplo, e iterativo para o que é particular. Vejamos a natureza serial do AUP que inclui as mesmas quatro fases adotadas pelo RUP:

Concepção

Tem como objetivos a identificação do escopo do projeto, de uma arquitetura em potencial para o sistema e a viabilidade do projeto.

Elaboração

Tem foco na comprovação da arquitetura do sistema.

Construção

Visa a construir um software funcional de forma iterativa e incremental que atenda às necessidades das partes interessadas do projeto.

Transição

Tem como objetivo validar e implantar o sistema em ambiente de produção.

Processo Unificado Ágil - AUP

As atividades que são realizadas de forma iterativa pelos membros da equipe de desenvolvimento para construir, validar e entregar software operacional.

Modelagem

Implementação

Testes

Implantação

Gerenciamento de Configuração

Gestão de Projetos

Ambiente

Qualidade de Software

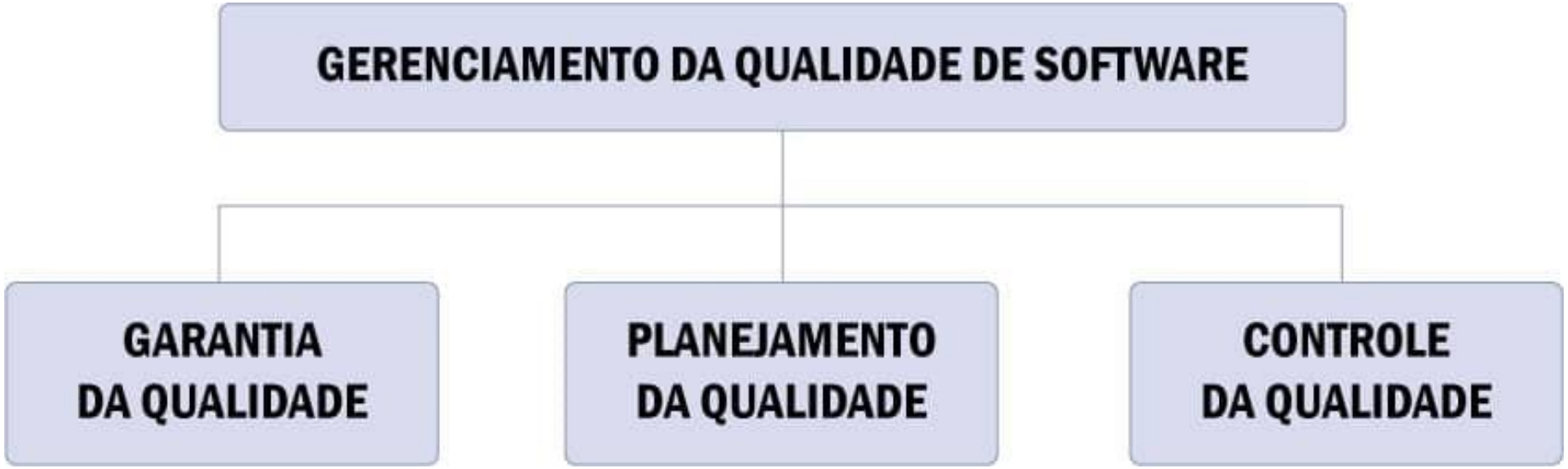
Engenharia de Software versus Qualidade

A engenharia de software é uma tecnologia em camadas.

Camadas da Engenharia de Software.

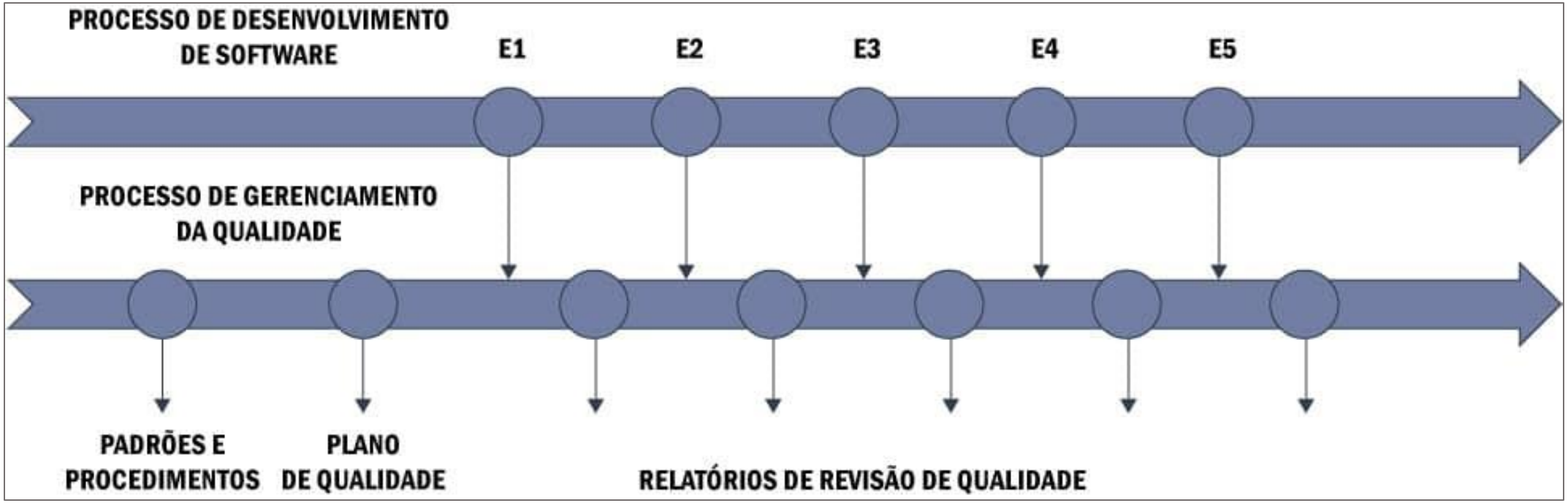
Qualidade de Software

O gerenciamento de qualidade de software usualmente é estruturado em três atividades principais:



Gerenciamento da Qualidade.

Qualidade de Software



Gerenciamento de qualidade versus processo de desenvolvimento.

Qualidade de Software

Podemos visualizar a qualidade de software sob duas dimensões:

Processo

Para garantir software de qualidade, é essencial tomar decisões bem informadas em todas as etapas do desenvolvimento.

Produto

Softwares mal testados prejudicam empresas com erros que impactam tomadas de decisão, produtividade e eficiência. A qualidade das informações é crucial.

Qualidade do Processo

Processo é uma sequência de etapas que permitem a geração de um produto, no nosso caso, o software.

Atenção!

O processo permite uma melhor tratativa em relação à complexidade de obtenção de um determinado produto, pois na maioria das vezes é um trabalho multidisciplinar realizado por analistas, programadores, gerentes de projeto, gerentes de teste e outros.

Qualidade do Processo

Uma metodologia de processo genérica, segundo Pressman (2016), compreende cinco atividades:

Comunicação

Facilita a compreensão do problema, objetivos e requisitos do projeto.

Planejamento

Inclui o gerenciamento de projeto.

Modelagem

Abrange a geração de modelos, tal como o modelo de casos de uso.

Construção

Inclui, codificação e testes baseados em modelos gerados e no planejamento.

Entrega

Disponibiliza o produto em produção de acordo com o planejado.

Qualidade do Processo

- Estruturar processos com mecanismos para detectar defeitos em artefatos e evitar propagação de erros.
- Os testes podem ser aplicados a todas as tarefas do processo de desenvolvimento, chamados testes de verificação ou testes estáticos.
- Os testes visam avaliar artefatos gerados ao longo do processo, como diagramas de classes ou de casos de uso.

Atenção!

A produção de software com qualidade exige o estabelecimento de um processo de desenvolvimento de software, pois não poderemos garantir a qualidade de algo que não existe.

Qualidade do Produto

Pressman (2016) descreve os fatores de qualidade do produto software de McCall, sendo uma proposta de categorização dos fatores que afetam a qualidade de software. Esses fatores de qualidade de software focam em três etapas de um produto de software:

Fatores de qualidade de software de McCall.

Qualidade do Produto

A qualidade do produto pode ser garantida por meio de um sistemático plano de testes executado durante a etapa de codificação do processo de software, sendo esses testes denominados de testes de validação.

Inicia-se com a validação da estrutura interna e aderência aos requisitos da unidade de software.

Em seguida, valida-se a integração com unidades anteriores, com foco nas interfaces dos componentes de software.

Por fim, realiza-se a validação funcional e não funcional do software, envolvendo os usuários.

Custos de Qualidade

O custo da qualidade.

Processo de Garantia da Qualidade

- Sistema de Garantia da Qualidade de Software (SQA) é uma estrutura organizacional para garantir a qualidade de produtos e serviços de software.
- Inclui procedimentos, responsabilidades, processos e recursos.
- O SQA envolve definição de padrões e a escolha de ferramentas e métodos para apoiar esses padrões no processo de desenvolvimento de software.

Atenção!

A equipe de qualidade deve examinar o software sob o ponto de vista do cliente, assertiva alinhada com a máxima de que “A qualidade é conformidade aos requisitos”, ou seja, o software deve atender às necessidades do cliente.

Processo de Garantia da Qualidade

A garantia da qualidade de software é composta pelas atividades que se concentram na gestão da qualidade de software.

Padrões	**Revisões e Auditorias**	**Testes**	**Coleta e Análise de Erros/Defeitos**	**Gerenciamento de Mudanças**
Educação	**Gerencia dos Fornecedores**	**Administração Segura**	**Proteção**	**Administração de Riscos**

Padrões de Software

Podemos destacar alguns aspectos relacionados aos padrões de software:

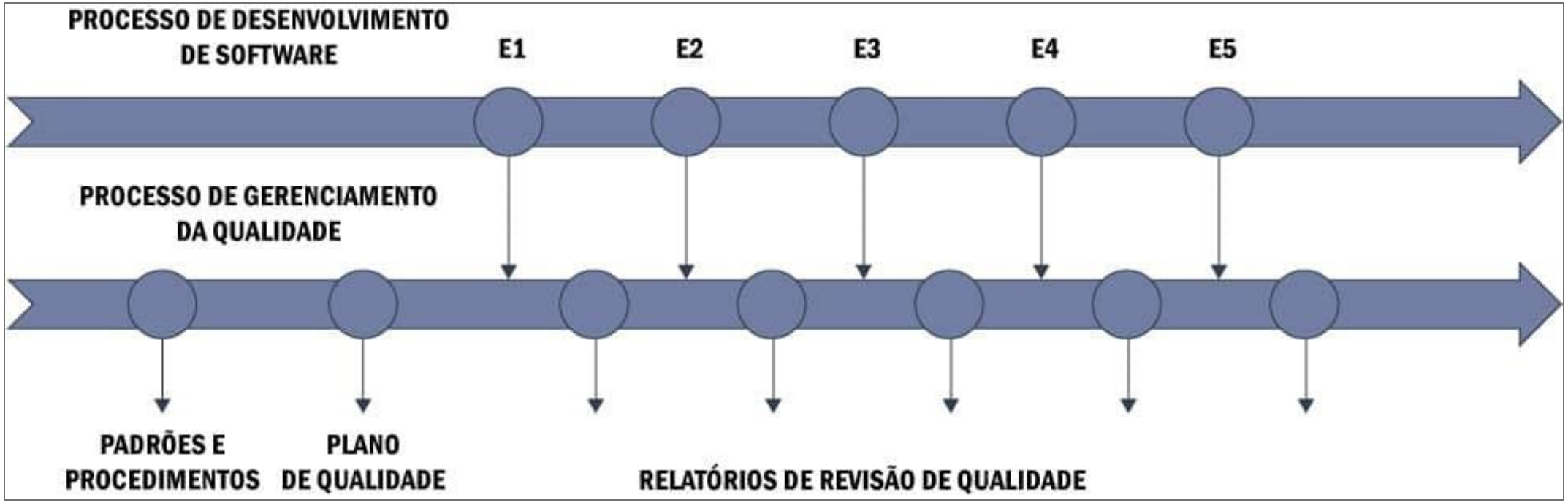
Padrões de Produto

Inclui padrões de documentação, tal como um cabeçalho de comentário padronizado para uma definição de uma classe que permitirá instanciar objetos, e padrões de codificação, que definem como uma linguagem de programação deve ser usada.

Padrões de Processo

Definem os processos que devem ser seguidos durante o desenvolvimento de software, tal como o processo da engenharia de requisitos, e uma descrição dos documentos que devem ser escritos ao longo desses processos.

Padrões de Software



Gerenciamento de qualidade versus processo de desenvolvimento.

Padrões de Software

Alguns aspectos relacionados aos padrões de software:

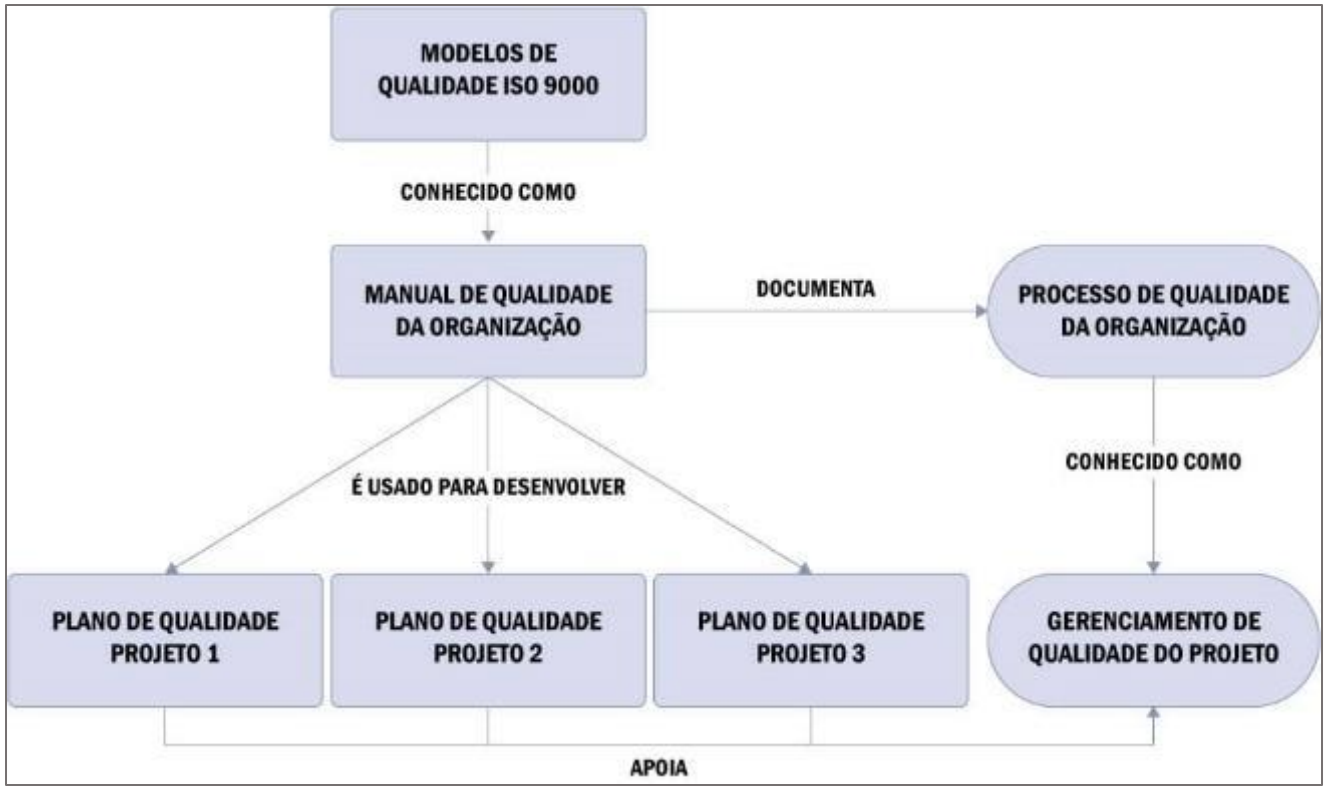
Incluem melhores práticas, evitando a repetição de erros.

Oferecem um framework para implementar a garantia de qualidade e assegurar a conformidade com melhores práticas.

Permitem a continuidade quando da ocorrência de substituições na equipe.

Padrão ISO 9000

O manual de qualidade segue o padrão ISO 9000 e guia os planos de qualidade de cada projeto de acordo com o processo de desenvolvimento de software. Isso permite a adaptação do gerenciamento de qualidade a cada projeto.



ISO 9000 versus gerenciamento da qualidade.

Padrão ISO 9126

O padrão ISO 9126 foi desenvolvido como uma tentativa de identificar os atributos fundamentais de qualidade para software. O padrão identifica seis atributos fundamentais de qualidade:

Funcionalidade	**Confiabilidade**	**Usabilidade**
Eficiência	**Facilidade de Manutenção**	**Portabilidade**

Gerenciamento da Qualidade de Acordo com o *PMBOK*

Uma das dez áreas de conhecimento do PMBOK é o Gerenciamento da Qualidade do Projeto que inclui os processos para incorporação da política de qualidade da organização com relação ao planejamento, gerenciamento e controle dos requisitos de qualidade do projeto e do produto para atender às expectativas dos clientes.

Planejamento

O processo de planejamento identifica requisitos de qualidade e cria o plano de gerenciamento da qualidade para demonstrar a conformidade no projeto.

Execução

Na execução, o plano de gerenciamento da qualidade é transformado em atividades que seguem as políticas de qualidade da organização.

Monitoramento e Controle

No planejamento, monitora-se a execução das atividades de qualidade para garantir que as saídas do projeto atendam às expectativas do cliente.

Gerenciamento da Qualidade de Acordo com o *PMBOK*

Atenção!

O Gerenciamento da Qualidade do Projeto trata do gerenciamento do projeto e das entregas do projeto, adotando, entre outras, a premissa de que a prevenção é preferível à inspeção, pois é melhor obter a qualidade nas entregas, em vez de identificar não conformidades em uma determinada inspeção. Nesse caso, o custo de prevenção é geralmente muito menor do que o custo de corrigir os erros identificados.

Gerenciamento da Qualidade de Acordo com o *PMBOK*

Existem cinco níveis de gerenciamento da qualidade cada vez mais eficaz:

1. Deixar que o cliente encontre os defeitos é uma abordagem que pode gerar problemas de garantia, recalls, retrabalho e outros, sendo em geral uma abordagem mais cara.

2. Detectar e corrigir os defeitos antes que as entregas sejam enviadas para o cliente como parte do processo de controlar a qualidade, pois esse processo tem custos relacionados, que são principalmente os custos de avaliação e os custos internos de falhas.

3. Usar a garantia da qualidade para examinar e corrigir o processo em si e não apenas defeitos especiais.

4. Incorporar a qualidade no planejamento e design do projeto e do produto.

5. Criar uma cultura na organização que esteja ciente e comprometida com a qualidade em processos e produtos.

Planejamento da Qualidade

- A equipe de SQA auxilia a equipe de software a obter alta qualidade no produto final.
- O SEI recomenda ações de garantia da qualidade que incluem planejamento, supervisão, manutenção de registros, análise e relatórios.
- A equipe de qualidade prepara um plano de qualidade de software como parte do planejamento do projeto.
- O plano orienta ações de garantia da qualidade, incluindo avaliações, auditorias, padrões, relatórios, produtos do SQA e feedback.
- O plano de qualidade serve como guia para instituir garantia da qualidade de software em cada projeto.

Planejamento da Qualidade

Foi publicado pelo IEEE um padrão para planos de qualidade de software.

- O propósito e o escopo do plano.
- Uma descrição de todos os artefatos resultantes de engenharia de software (por exemplo, modelos, documentos, código-fonte).
- Todos os padrões e práticas que são aplicados durante a gestão de qualidade.
- As ações e tarefas da garantia da qualidade (incluindo revisões e auditorias) e sua aplicação na gestão de qualidade.
- As ferramentas e os métodos que dão suporte às ações e tarefas da garantia da qualidade.
- Procedimentos para administração de configurações de software.
- Métodos para montagem, salvaguarda e manutenção de todos os registros relativos à garantia da qualidade.
- Papéis e responsabilidades dentro da organização relacionados com a qualidade do software.

Planejamento da Qualidade segundo o *PMBOK*

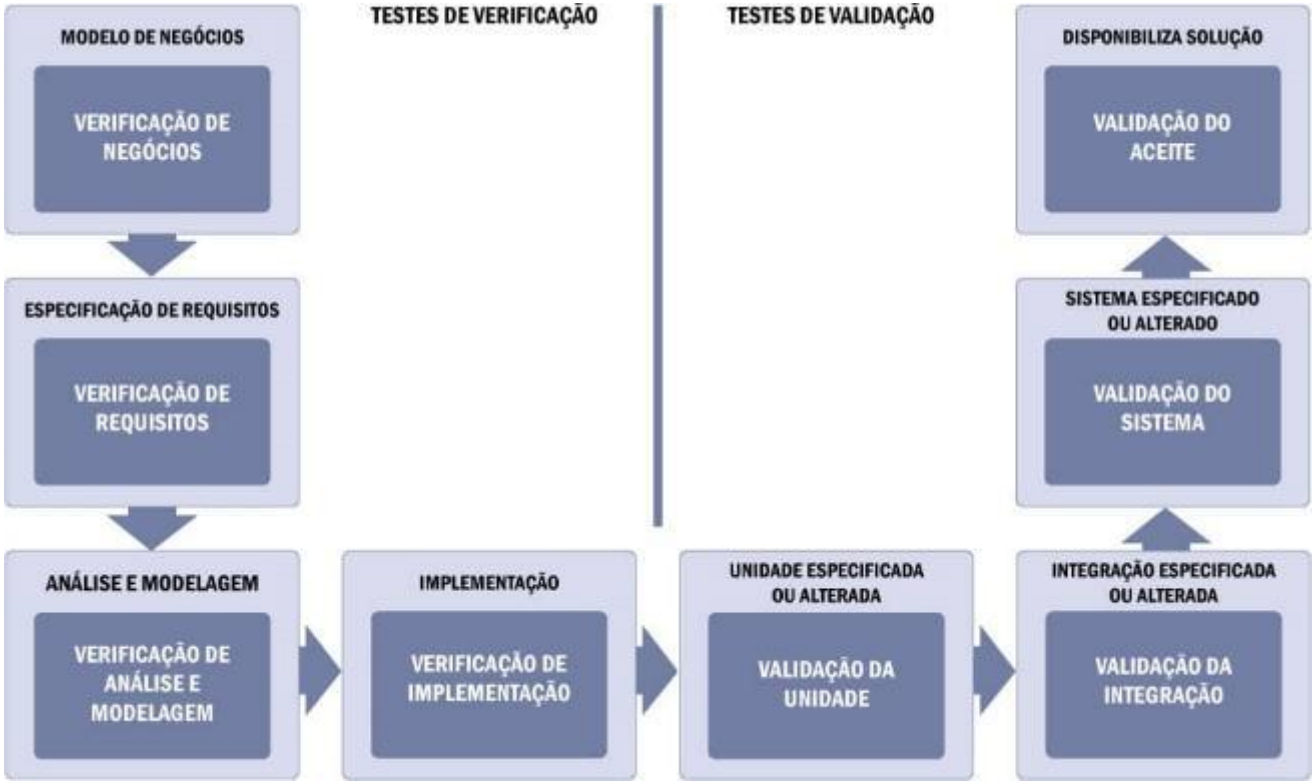
- O PMBOK destaca a importância da qualidade em projetos por meio do Gerenciamento da Qualidade do Projeto.
- A área de conhecimento de Gerenciamento da Qualidade do Projeto possui três processos em diferentes grupos de processos.
- O processo "Planejar o Gerenciamento da Qualidade" identifica requisitos e padrões de qualidade do projeto e documenta como o projeto atenderá a esses requisitos.
- Esse processo fornece orientação sobre como a qualidade será gerenciada e verificada ao longo de todo o projeto.

Planejamento da Qualidade segundo o *PMBOK*

Entradas	Técnicas e Ferramentas	Saídas
Termo de abertura do projeto	Opinião especializada	Plano de gerenciamento da qualidade
Plano de gerenciamento do projeto	Coleta de dados	Métricas da qualidade
Documentos do projeto	Análise de dados	Atualizações do plano de gerenciamento do projeto
Fatores ambientais da empresa	Tomada de decisões	Atualizações de documentos do projeto
Ativos de processos organizacionais	Representação de dados	
	Planejamento de testes e inspeções	
	Reuniões	

Planejar o Gerenciamento da Qualidade.

Planejamento de Testes e Revisões



Processo de Garantia da Qualidade em “U”.

Planejamento de Testes e Revisões

Testes e Revisões

Os testes de verificação avaliam o desenvolvimento de software para garantir conformidade com padrões e reduzir riscos de falhas. Eles se dividem em:

Métodos Estruturados de Verificação.

Planejamento de Testes e Revisões

Testes de Validação

Testes de Validação

Podemos dividir o aceite em três momentos distintos:

Teste Alpha

Os usuários finais são convidados a operar o software dentro de um ambiente controlado, sendo realizado na instalação do desenvolvedor.

Teste Beta

Os usuários finais são convidados a operar o produto utilizando suas próprias instalações físicas, ou seja, o software é testado em um ambiente não controlado pelo desenvolvedor.

Aceite Formal

É uma variação do Teste Beta, cabendo aos próprios clientes e usuários determinarem o que deverá ser testado e validar se os requisitos foram adequadamente implementados.

Testes de Sistema

Após ser validado, o software deve ser combinado com outros elementos do sistema, tais como hardware, base de dados etc.

Teste de Recuperação

Teste de Segurança

Teste por Esforço

Teste de Desempenho

Teste de Disponibilização

Controle de Qualidade

Controlar a qualidade é o processo de monitorar e registrar a conformidade das entregas com os requisitos, garantindo a integridade do produto antes da entrega.

Atenção!

Na coluna de Entradas, destacam-se os "Documentos de testes e avaliação," incluindo os testes aplicados na garantia do processo e do produto. Já as solicitações de mudança na coluna de Saídas resultam em atualizações no registro do projeto, abrangendo correções de defeitos e ajustes nos métodos e cronogramas.

Medição

- Medição é crucial na engenharia de software, seguindo o princípio de que "não se controla o que não se mede".
- A medição envolve termos como medida, medição e métricas para quantificar atributos de produtos ou processos.
- Métricas de software são medidas quantitativas para avaliar atributos de sistemas, componentes e processos, sendo essenciais para a qualidade e o gerenciamento de software.

Processo de Medição

A medição é essencial na engenharia de software, envolvendo medidas quantitativas para avaliar atributos de produtos e processos. Métricas de software são usadas para avaliar a qualidade e eficácia de processos e ferramentas, permitindo a tomada de decisões informadas.

Escolher as medições a serem feitas

Selecionar os componentes a serem avaliados

Mediar as características dos componentes

Identificar medições anômalas

Analisar componentes anômalos

Método GQM

O método GQM (Goal – Question – Metric) é uma técnica aplicada no planejamento do trabalho de medição, ou seja, permite a identificação de métricas significativas para qualquer parte do processo de software. O GQM organiza o planejamento de uma medição de software em etapas.

Objetivo

O objetivo de medição deve ser específico para uma atividade ou característica do produto e em conformidade com os requisitos do software.

Questões

Definir um conjunto de questões para alcançar o objetivo, como "Qual é a cobertura dos testes?", estabelecendo uma conexão entre os objetivos planejados e as métricas.

Métricas

Deverão ser formuladas em função das respostas às questões elaboradas, tal como “Número de requisitos testados”.

Métricas

- Métricas de software são ferramentas quantitativas que asseguram a qualidade do software, analisando atributos mensuráveis e permitindo melhorias nos requisitos, modelos e processo de desenvolvimento.

- Métricas representam características mensuráveis em sistemas, documentação ou processos de desenvolvimento de software. De acordo com Sommerville (2019), as métricas de software podem ser:

Métricas de Controle ou de Processo

Métricas de Previsão ou de Produto

Métricas

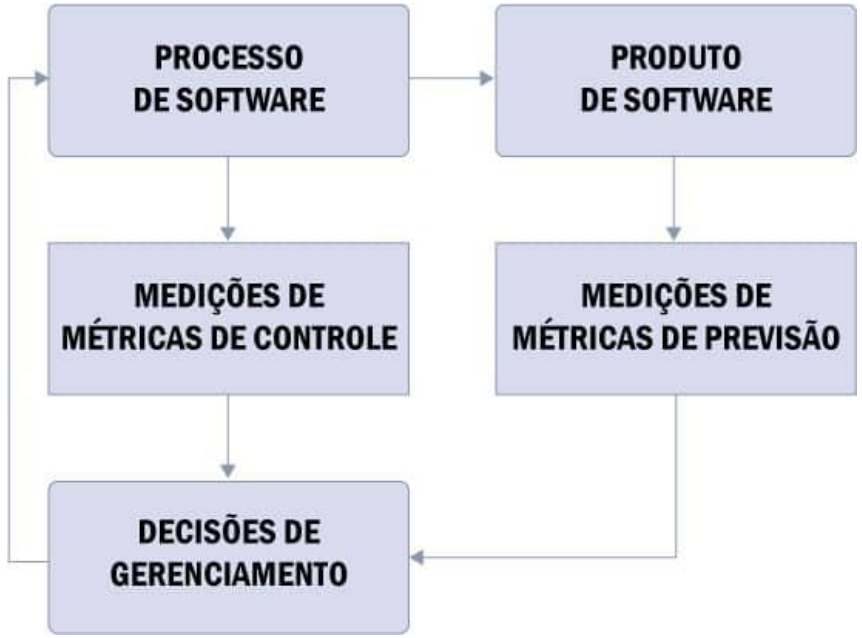
As métricas estáticas ajudam a avaliar a complexidade, a compreensibilidade e a manutenibilidade de um sistema ou de seus componentes.

Métrica	Descrição
Fan-in / fan-out	• Fan-in: Número de funções ou métodos que chamam outra função ou método (por exemplo, x). • Fan-out: Número de funções que são chamadas pela função x. • Um valor elevado para o fan-in significa que x está fortemente acoplado ao resto do projeto e as mudanças em x terão um amplo efeito dominó. • Um valor elevado para fan-out sugere que a complexidade geral de x pode ser alta por causa da complexidade da lógica de controle necessária para coordenar os componentes chamados.
Comprimento do código	• Medida do tamanho de um programa. • Geralmente, quanto maior o tamanho do código de um componente, mais complexo e propenso a erros ele tende a ser.
Complexidade ciclomática	• Visa a analisar a estrutura do código-fonte.

Métricas estatísticas de produto.

Métricas

As métricas de controle e de previsão podem influenciar a tomada de decisões de gerenciamento. Os gerentes usam as medições de processo para decidir se as alterações desse processo devem ser feitas e as métricas de previsão para decidir se as alterações de software são necessárias e se o software está pronto para entrar em produção.



Medições de previsão e controle.

Gerenciamento de Configurações

Conceito de Gerenciamento de Configurações

- O GCS (Gestão de Configuração de Software) envolve atividades que controlam e gerenciam alterações em artefatos de software, estabelecendo relações, controlando versões e auditando mudanças.

- É uma parte fundamental da garantia de qualidade do software e pode ser realizada pela equipe de qualidade ou por uma equipe designada, destacando a importância de controlar alterações para manter a qualidade do software.

Elementos de Componentes

Sistema de ferramentas integradas para gerenciar itens de configuração de software.

Elementos de Processo

Ações e tarefas que estabelecem uma abordagem eficaz para a gestão de alterações em software para todas as partes envolvidas.

Elementos de Construção

Ferramentas que automatizam a construção de software, garantindo a montagem correta de componentes validados.

Elementos Humanos

Ferramentas e processos para implementar um GCS eficaz na equipe de software.

Planejamento de Gerenciamento de Configurações

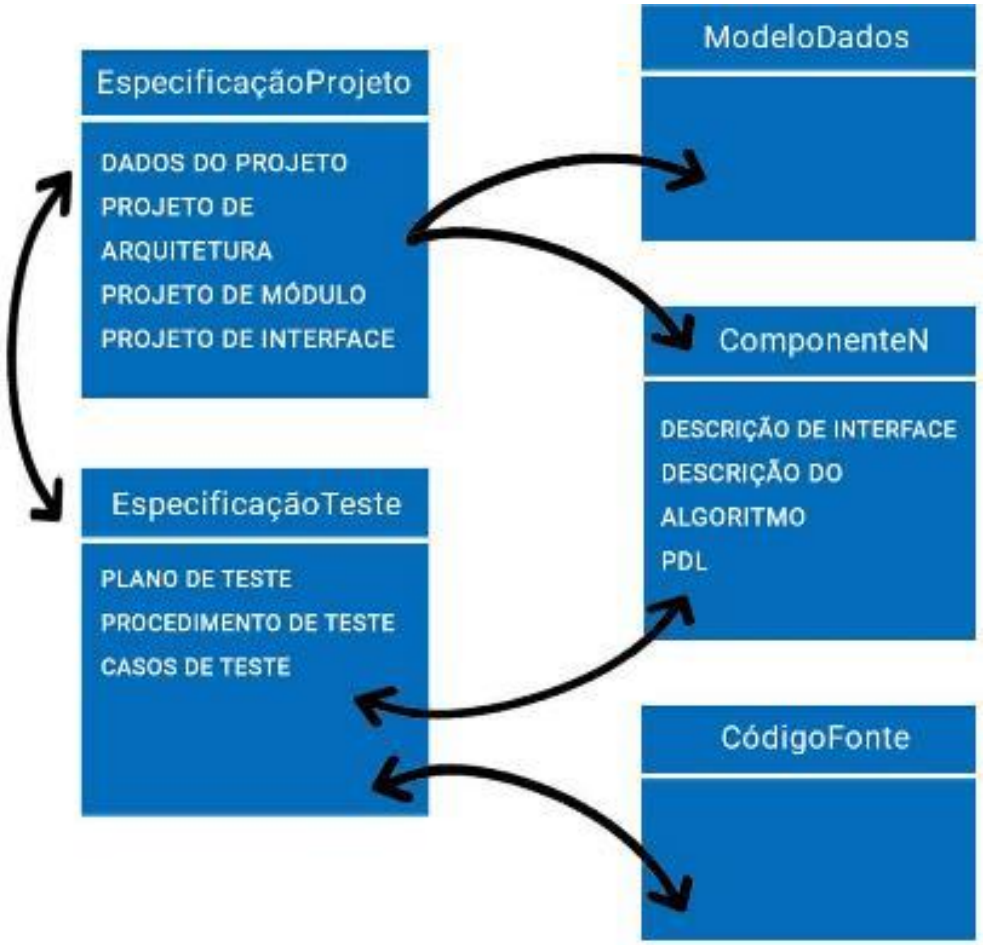
O plano de gerenciamento de configurações descreve os padrões e procedimentos que devem ser adaptados para atender aos requisitos e às restrições de cada projeto específico. O plano de GCS deve incluir:

- Definição dos itens de configuração de software (ICS) e do esquema de identificação a ser utilizado para gerenciá-los.
- Nomeação do responsável pelos procedimentos de gerenciamento de configuração e submissão de itens controlados à equipe de gerenciamento de configurações.
- Definição de políticas de gerenciamento de configurações para a equipe, visando a padronização no controle de mudanças e versões.
- Especificação das ferramentas aplicadas no gerenciamento de configurações e os processos para uso dessas ferramentas.
- Descrição da estrutura do banco de dados de configuração para registro e manutenção das informações de configuração.

Gestão de Item de Configuração de Software (ICS)

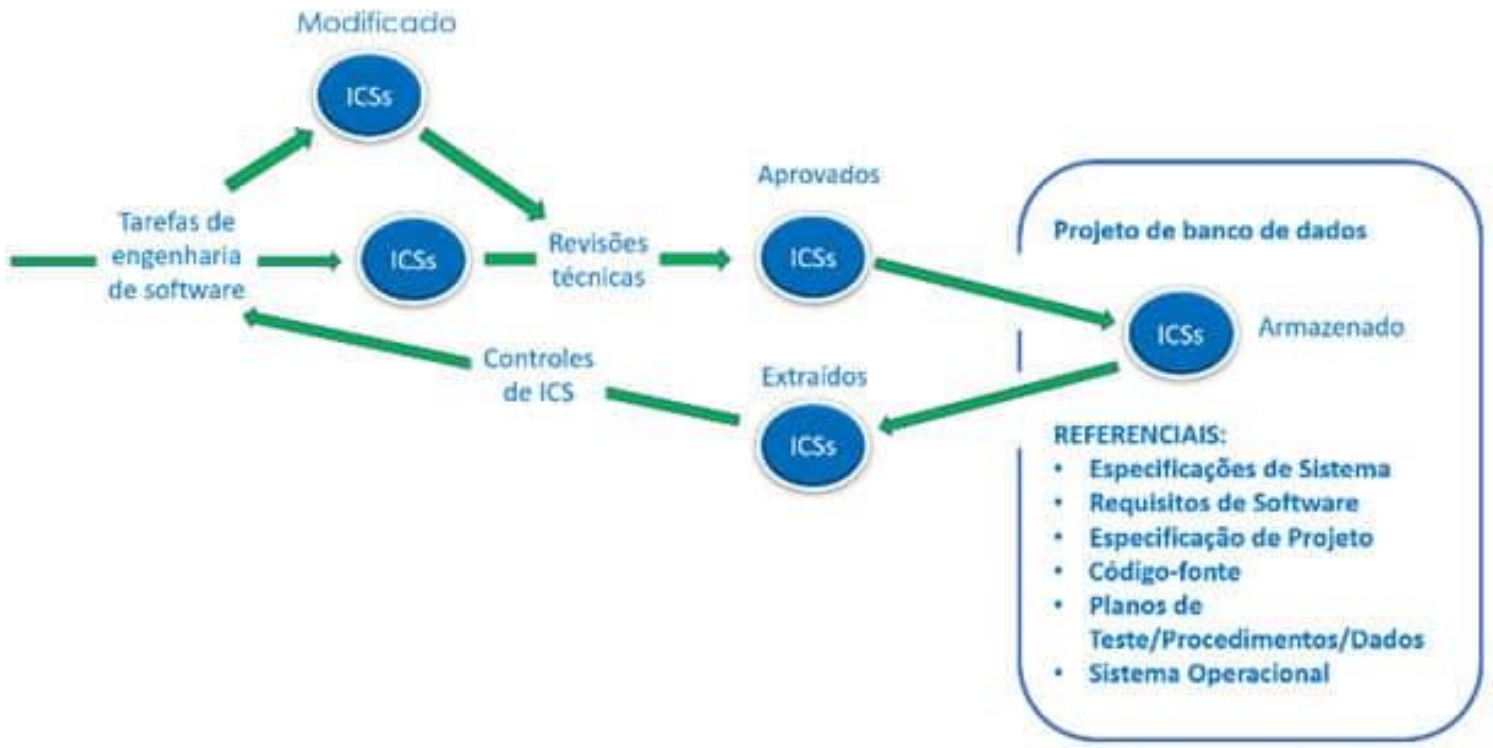
- Mudanças são comuns no desenvolvimento de software devido à evolução do entendimento do problema.
- O item de configuração de software (ICS) abrange documentos, casos de teste e ferramentas de software e é definido no plano de GCS.
- A disponibilidade de ferramentas de software é crucial para manter a consistência durante as alterações e evitar impactos negativos.
- ICS são objetos de configuração organizados em um banco de dados, como EspecificaçãoProjeto e CódigoFonte, com relações de composição.
- Mudanças em ICS requerem um processo formal de avaliação; uma referência é um marco no desenvolvimento, marcada por ICS aprovados após revisão técnica.

Gestão de Item de Configuração de Software (ICS)



Objetos de configuração.

Gestão de Item de Configuração de Software (ICS)”.



ICS versus banco de dados de projeto.

Processo de Gerenciamento de Configuração de Software

O processo de gerenciamento de configurações de software (GCS), proposto por Pressman (2016), inclui tarefas que têm quatro objetivos primários:

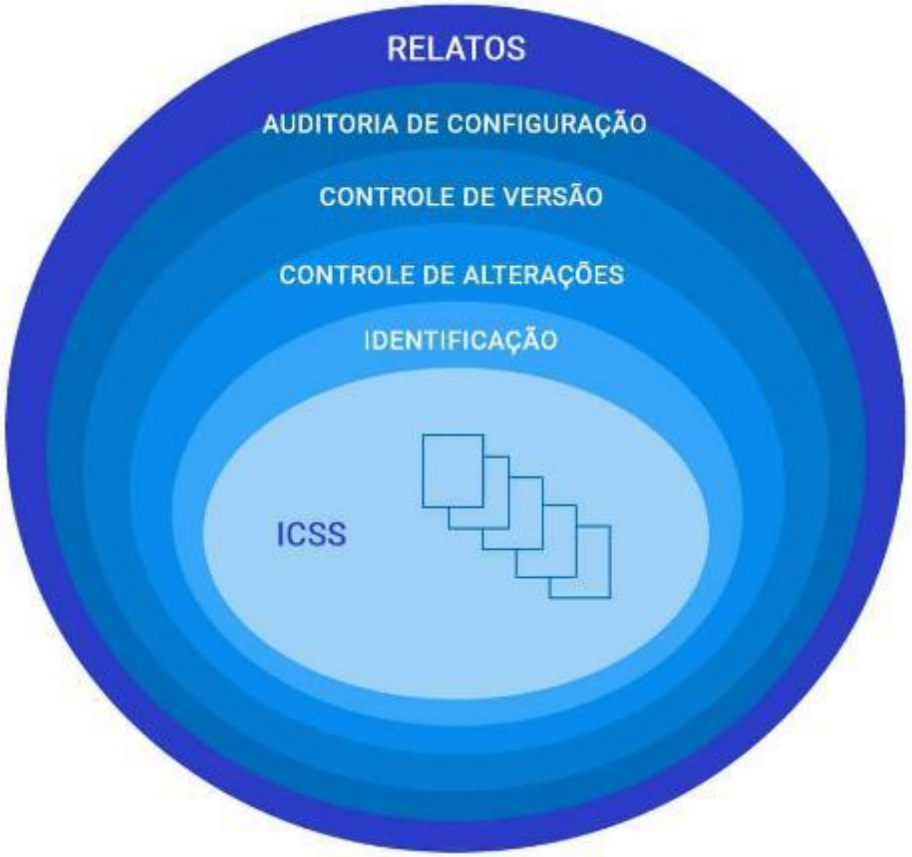
Identificar todos os itens que coletivamente definem a configuração do software.

Gerenciar alterações de um ou mais desses itens.

Facilitar a construção de diferentes versões de uma aplicação.

Assegurar que a qualidade do software seja mantida à medida que a configuração evolui com o tempo.

Processo de Gerenciamento de Configuração de Software



Camadas do processo GCS.

Processo de Gerenciamento de Configuração de Software

A aplicabilidade das ações das tarefas GCS em um ICS depende se alterações são solicitadas. Cada ICS tem uma versão específica com registro de informações para fins de auditoria e relatórios.

Atenção!

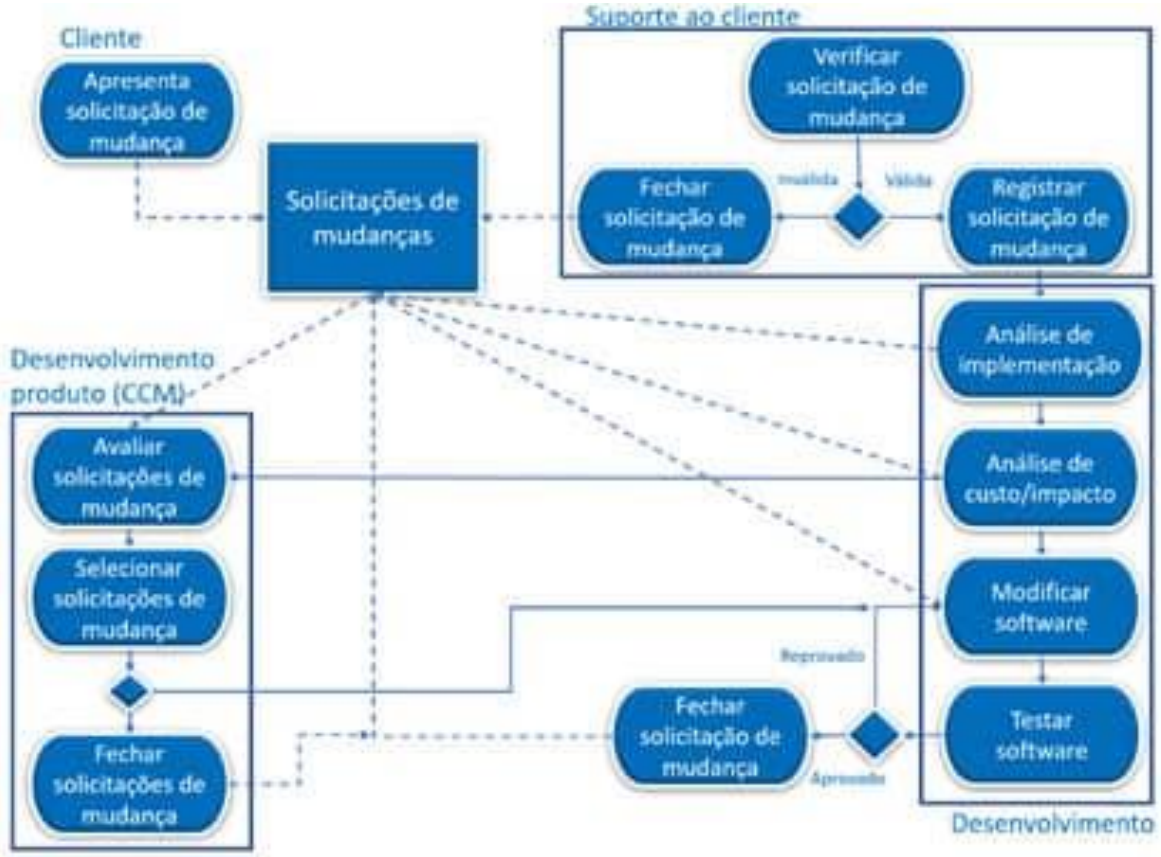
As camadas são concêntricas, de modo que os itens de configuração de software (ICS) transitam da camada mais interna para as camadas externas, tornando-se parte da configuração do software de uma ou mais versões da aplicação ou sistema.

Gerenciamento de Mudanças

- O controle de alterações gerencia mudanças de requisitos e defeitos de forma controlada, priorizando as mais urgentes e eficazes.

- O processo é ativado após a entrega do software, adaptando-se à complexidade do projeto e requer ferramentas como sistemas de rastreamento de defeitos e pacotes de gerenciamento de configuração.

Gerenciamento de Mudanças



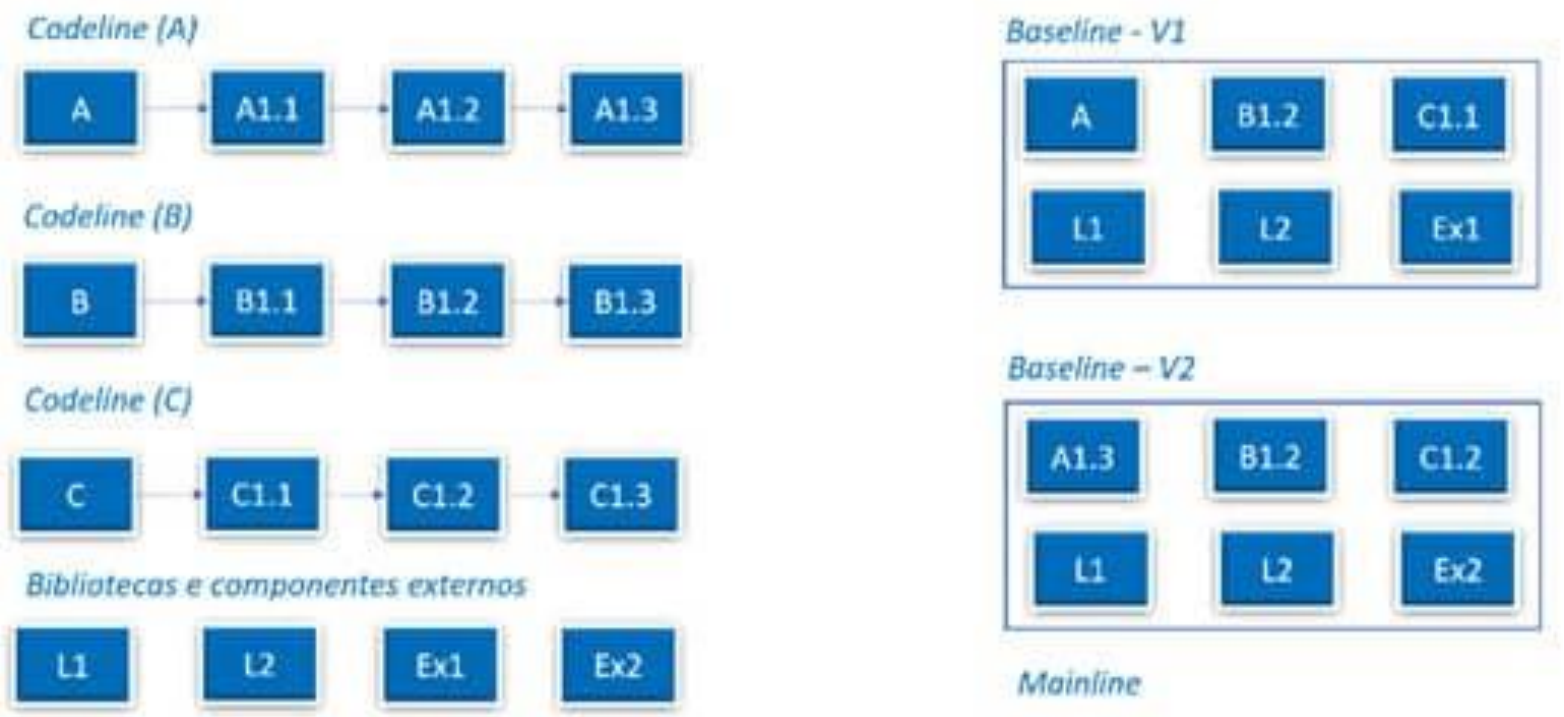
Gerenciamento de mudança.

Gerenciamento de Mudanças

- A implementação de mudanças depende de fatores como consequências, usuários beneficiados, custos e disponibilidade de novas versões.

- Solicitações de mudança em produtos de software vêm de diversas fontes, e em métodos ágeis, os clientes colaboram com a equipe de desenvolvimento para avaliar e implementar mudanças nos requisitos do sistema.

Gerenciamento de Versões



Codelines e baselines.

Gerenciamento de Versões

- As baselines são importantes porque, muitas vezes, é preciso recriar uma versão individual de um sistema, tal como a instanciação de um software em versões específicas para cada cliente, ou mesmo recriar a versão entregue a um cliente, caso relate defeitos em seu sistema que necessitem ser alterados.

- Existem dois tipos de sistemas de controle de versão modernos:

1

Sistemas centralizados compostos por um único repositório mestre que mantém todas as versões dos componentes de software.

2

Sistemas distribuídos compostos por várias versões do repositório de componentes ao mesmo tempo.

Gerenciamento de Versões

- Ambos os sistemas possuem funcionalidades semelhantes, mas as implementam de maneiras diferentes.
- As principais características destes sistemas são:

Identificação de Versão e Lançamento

Registro do Histórico de Mudanças

Desenvolvimento Independente

Apoio de Projeto

Gerenciamento de Armazenamento

Gerenciamento de Versões

Check-in e check-out de repositório centralizado.

Gerenciamento de Releases

- Uma release de software pode ser uma versão principal com novas funcionalidades (major release) ou uma versão secundária para correção de defeitos (minor release), dependendo das necessidades dos clientes.

- O gerenciamento de lançamentos pode ser complexo, especialmente para muitos clientes com diferentes hardwares, exigindo sistemas de gerenciamento de configurações para rastrear versões e relacionamentos entre lançamentos.

- Em caso de problemas, o GCS deve ser capaz de recuperar todas as versões dos componentes usados em um sistema específico.

Construção de Sistemas

Construção de sistemas.

Construção de Sistemas

Funcionalidades usualmente disponíveis em ferramentas para integração e construção de sistemas:

Gestão do Script de Construção

Integração com o Sistema de Controle de Versão

Recompilação Mínima

Criação do Sistema Executável

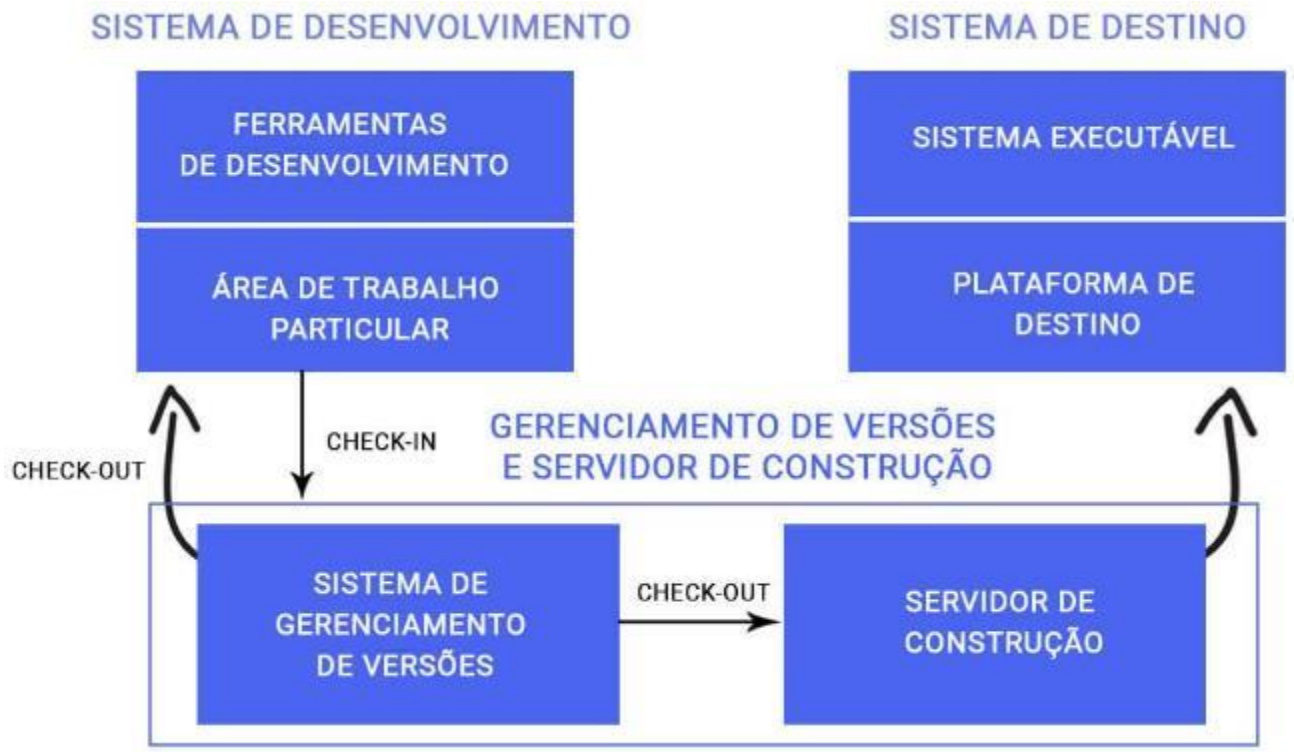
Automação dos Testes

Emissão de Relatórios

Geração da Documentação

Construção de Sistemas

As plataformas de sistema envolvidas, o que confere um grau de complexidade no processo de construção.

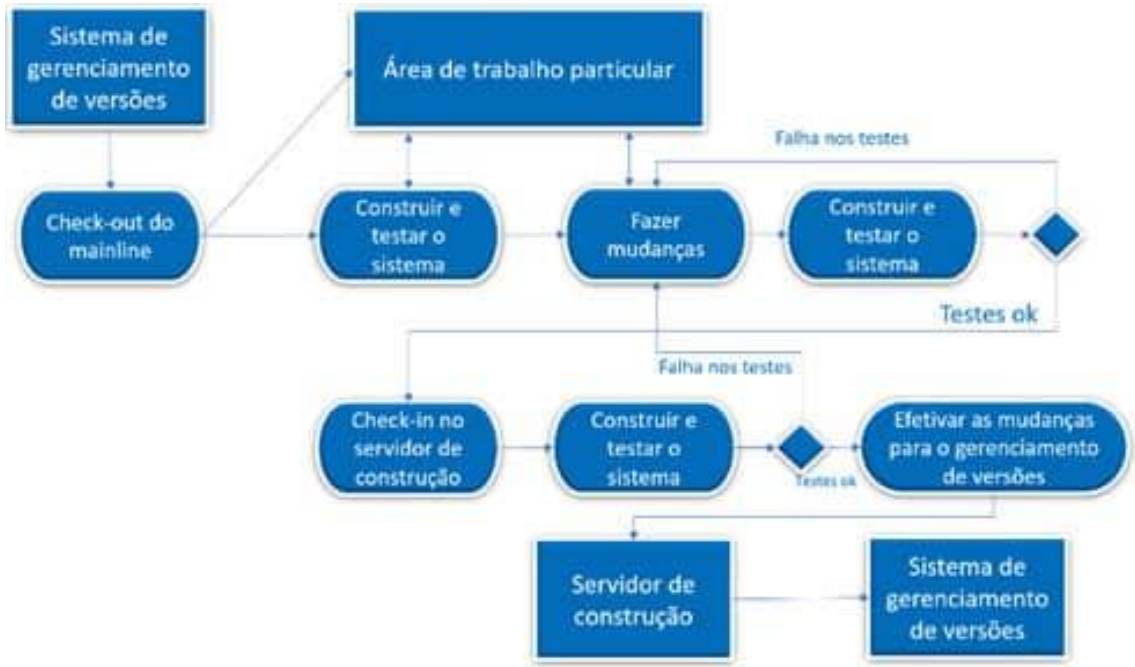


Plataformas de desenvolvimento, construção e destino.

Integração Contínua

- Os métodos ágeis recomendam que a construção do sistema seja realizada com bastante frequência, com testes automatizados utilizados para descobrir problemas de software.

- As construções frequentes fazem parte de um processo de integração contínua.



Integração contínua.

Integração Contínua

- A vantagem da integração contínua é que ela permite a descoberta e a alteração imediata dos problemas causados pelas interações entre diferentes desenvolvedores, sendo o sistema mais recente no mainline o sistema funcional definitivo.

- Podem ocorrer restrições na aplicação da integração contínua:

Em sistemas extensos com integração de aplicações, a construção e teste frequentes podem ser demorados e inviáveis para várias vezes ao dia.

Se a plataforma de desenvolvimento difere da plataforma de destino, os testes podem ser afetados pela incompatibilidade de hardware, sistema operacional e software, exigindo mais tempo para a realização dos testes.

Integração Contínua

Alternativas considerando um sistema de construção diária:

1. A equipe de desenvolvimento estabelece um horário de entrega, tal como 14h, dos componentes de sistema, cabendo aos desenvolvedores que tiverem novas versões dos componentes realizar e entrega no prazo estipulado.

2. A construção do sistema ocorre por meio da compilação e ligação dos componentes, sendo gerado um sistema completo.

3. O novo sistema gerado passa por um conjunto de testes pela equipe de testes.

4. Os defeitos que foram descobertos durante o teste do sistema são documentados e devolvidos para os desenvolvedores do sistema que realizam a manutenção devida.

Ligação do Código-Fonte ao Código Aberto

- Ferramentas de construção minimizam a compilação ao verificar se existe uma versão compilada de um componente antes de recompilá-lo.

- A ligação entre código-fonte e código objeto é baseada em assinaturas exclusivas, que mudam quando o código-fonte é editado.

Ligando código-fonte e código objeto.

Ligação do Código-Fonte ao Código Aberto

Timestamps de modificação

- A assinatura do código-fonte é a data e hora de sua última modificação.
- Quando o código-fonte é alterado, o sistema recompila automaticamente para criar um novo código objeto.
- Exemplo com Comp.java e Comp.class mostra timestamps de 15 de fevereiro de 2018.
- O sistema recompila automaticamente o código-fonte se a versão compilada tiver data de modificação anterior à do código-fonte mais recente.

Ligação do Código-Fonte ao Código Aberto

Somas de verificação (checksum) de código-fonte

- A assinatura do arquivo de código-fonte é uma soma de verificação que identifica exclusivamente o arquivo de origem, mudando sempre que o texto é alterado.
- O código objeto gerado é marcado com a assinatura do código-fonte.
- Se não houver um código objeto com a mesma assinatura que o código-fonte, é necessária a recompilação.
- Abordagem de timestamps mantém apenas o código objeto compilado mais recentemente.
- A abordagem de soma de verificação permite manter várias versões do código objeto, usando a assinatura como vínculo entre código-fonte e objeto.

Ambientes de Gerenciamento de Configurações

Usualmente, os processos de gerenciamento de configurações são padronizados, ou seja, incluem aplicações de procedimentos predefinidos que requerem o gerenciamento de grande quantidade de dados.

Ambientes Abertos

Utiliza-se ferramentas de código aberto em cada etapa do processo, integradas por meio de procedimentos organizacionais padronizados para funções como gerenciamento de mudança, versões e construção de sistemas.

Ambientes Integrados

Fornecem recursos integrados para controlar versões, a construção de sistemas e o rastreamento de mudanças.

Apoio para Gerenciamento de Mudanças

Uma melhor prática sugere que um modelo de processo de mudanças deva ser projetado e integrado com um sistema de gerenciamento de versões, de modo que os documentos gerados nas atividades relacionadas com as mudanças sejam enviados para pessoas certas no tempo certo.

Editor de Formulários

Permite às pessoas criar ou completar formulários com propostas de mudanças.

Sistema de Workflow

O sistema permite ao gerenciamento de configuração definir responsabilidades e encaminhar formulários de solicitação de mudança automaticamente.

Bando de Dados de Mudança

Usado para gerenciar propostas de mudança, pode ser vinculado a um sistema de gerenciamento de versões, com recursos de consulta para localizar propostas de mudança específicas.

Elementos

Gera relatórios gerenciais sobre o status das solicitações de mudança enviadas.

Apoio para Gerenciamento de Versões

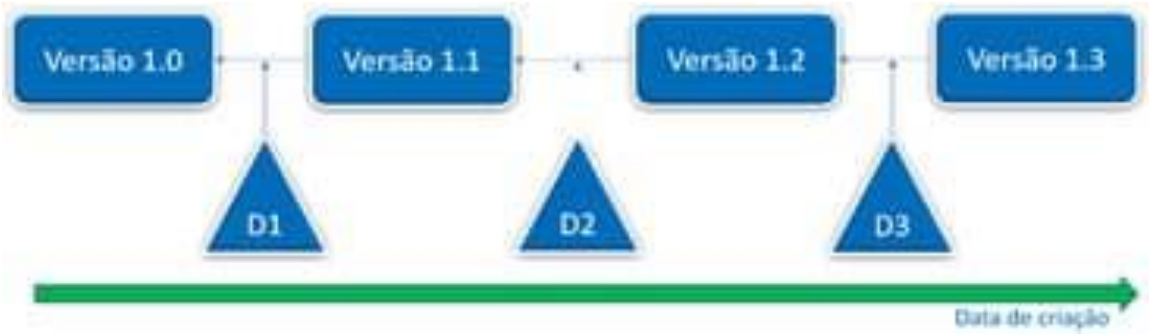
As ferramentas de gerenciamento de versões controlam um repositório de itens de configuração no qual os conteúdos dos repositórios não podem ser alterados.

Identificação de Versões e Releases

Gerenciamento de Armazenamento

Registro do Histórico de Mudanças

Suporte a Projetos



Ligando código-fonte e código objeto.

Suporte para a Construção de Sistemas

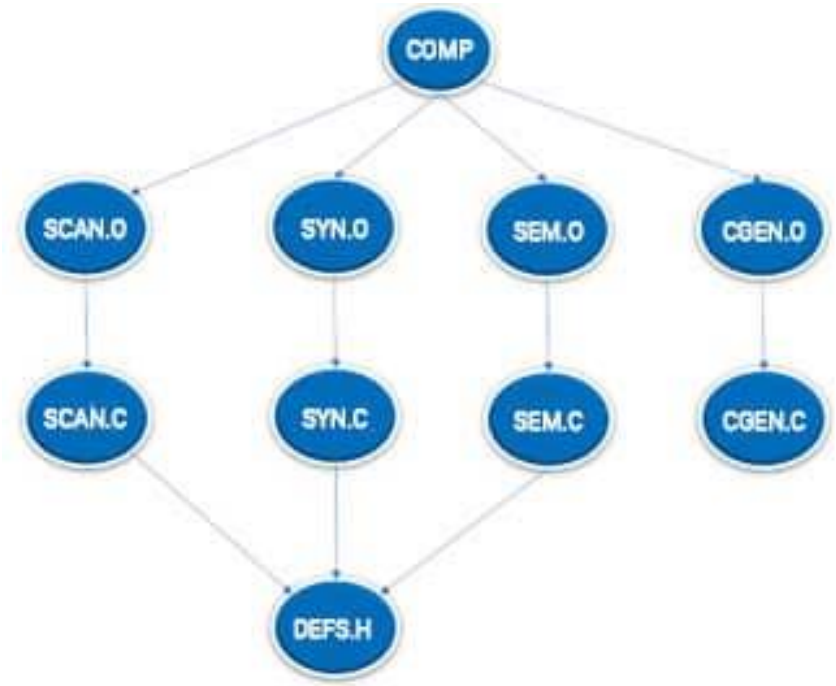
A construção de sistemas é um processo computacional intensivo, podendo incluir centenas de arquivos.

Atenção!

As ferramentas de construção de sistemas automatizam o processo de construção para reduzir a possibilidade de ocorrência de erro humano e, quando possível, minimizar o tempo necessário para a construção de sistemas.

Suporte para a Construção de Sistemas

Ferramentas de construção de sistemas usam a data de modificação para determinar a recompilação. Algumas mantêm registros da última versão do código objeto correspondente ao código-fonte modificado, enquanto outras usam abordagens sofisticadas, como marcas nos objetos derivados, para gerenciar versões e dependências.



Dependências entre componentes.

Ferramentas Case para Gerenciamento de Configuração

As ferramentas CASE permitem que as atividades fundamentais de gerenciamento de configuração resultem em ganhos de produtividade no gerenciamento e controle de itens de configuração gerados ao longo do processo de desenvolvimento de software.

Ferramentas de Suporte Individual

Ferramentas de Suporte ao Projeto

Ferramentas de Suporte Completo à Organização

Ferramentas Case para Gerenciamento de Configuração

Ferramentas de suporte individual:

- Controle de versões permite a identificação, armazenamento e gerenciamento dos itens de configuração.
- Controle de mudança tem como foco os procedimentos pelos quais as mudanças de um ou mais itens de configuração são propostas, avaliadas, aprovadas e implantadas.
- Integração contínua permite a identificação, empacotamento e preparação de uma baseline, garantindo que as mudanças sejam construídas, testadas e relatadas.