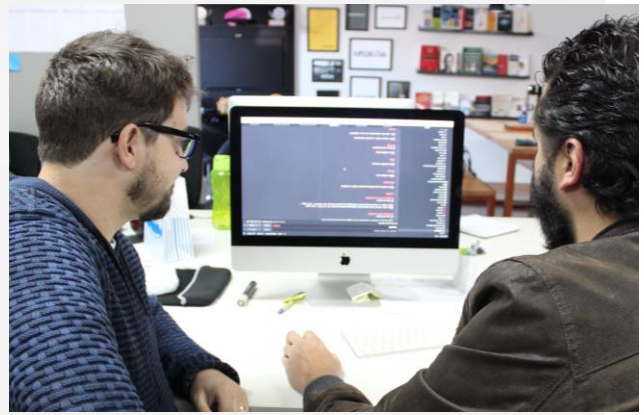
Engenharia de Software I

- Gerência de qualidade

Elicitação/
Especificação
de requisitos

Projeto de software

Plano de testes



Codificação

Testes de software

Implantação

Manutenção
Evolução

GERÊNCIA DE PROJETOS

Introdução

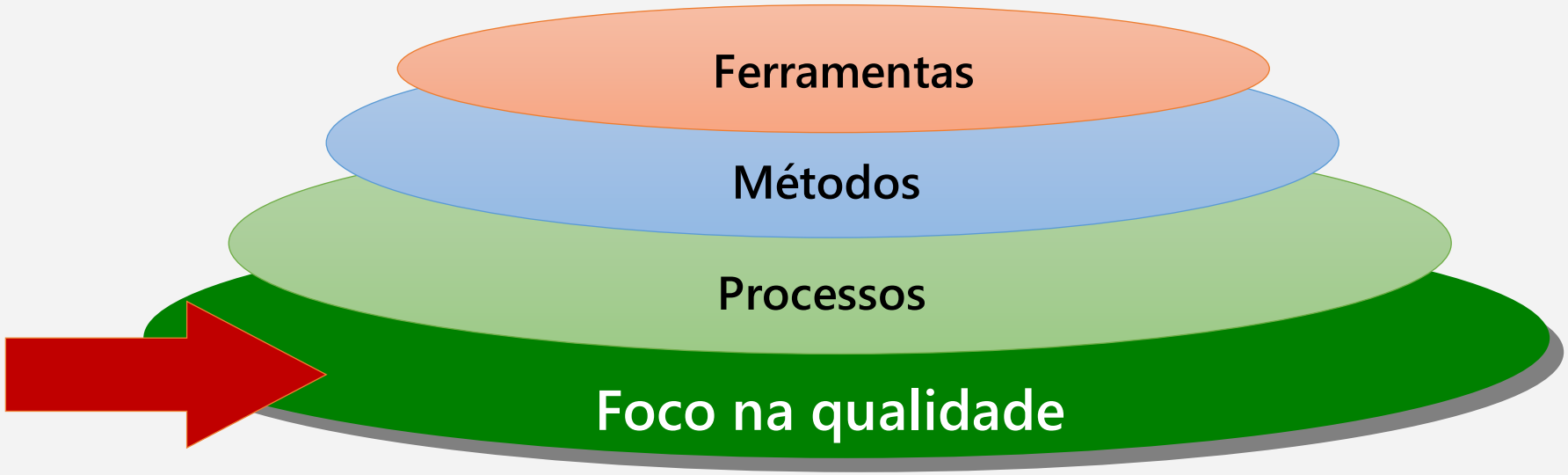
Qualidade é fator crítico de sucesso para a indústria de software

- Necessário para ter um setor de software competitivo, nacional e internacionalmente
- Necessidade de estar adequado a padrões internacionais de qualidade.

Introdução

Engenharia de Software é uma disciplina relacionada com a solução de problemas práticos da indústria de software.

A Engenharia de Software pode ser vista em camadas



Qualidade ?

?

?

?

Mas ... O que é qualidade?

Por exemplo, o que é um carro de qualidade?

Para responder a essa questão, devem ser considerados diversos fatores, como: segurança, desempenho, beleza, conforto, tamanho e custo, dentre outros.

O que é um software de qualidade?

O que é Qualidade de Software

Então,qualidade é um conceito relativo. Está diretamente relacionada à:

- **Conformidade com requisitos**: requisitos são especificados e espera-se que sejam atendidos.
- **Satisfação do cliente**: requisitos são especificados por pessoas para satisfazer outras pessoas.
- **Produto sem defeitos:** desenvolvido corretamente.

Qualidade de software

➔ **Conjunto de características** a serem **satisfeitas** em um determinado grau de modo que o software atendam **às necessidades** de **seus usuários**

CONCEITO DE MÚLTIPLAS FACETAS:
usabilidade, confiabilidade, eficiência, manutenibilidade, portabilidade, segurança, produtividade, entre outras.

➔ **Note que esse conceito foca produto.**

Como garantir que o produto de software sempre tenha a qualidade esperada?

Qualidade do produto X Qualidade do processo

Melhorando a qualidade do processo de software, é possível melhorar a qualidade dos produtos resultantes.

O que é qualidade de software?

Qualidade de Software *é uma subárea da Engenharia de Software que trata de aspectos relacionados a* ***obtenção*** *e* ***avaliação*** *da* ***qualidade do produto*** *e do* ***processo de software***.

Processo de software

CARACTERÍSTICAS DE UM PROCESSO IMATURO:
- Ad hoc -Improvisado
- Fortemente dependente dos profissionais
- Indisciplinado

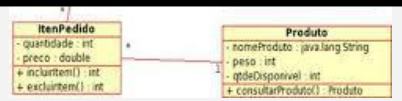
Consequências:
- pouca produtividade
- qualidade de difícil previsão
- alto custo de manutenção
- risco na adoção de novas tecnologias

CARACTERÍSTICAS DE UM PROCESSO MADURO:
- Processo conhecido e seguido por todos;
- Apoio visível da alta administração;
- Auditagem da fidelidade ao processo;
- Medidas do produto e do processo;
- Adoção disciplinada de tecnologias.

Consequências:
- Papéis e responsabilidades claramente definidos;
- Acompanhamento da qualidade do produto e da satisfação do cliente;
Expectativas para custos, cronograma, funcionalidades e qualidade do produto são usualmente alcançadas.

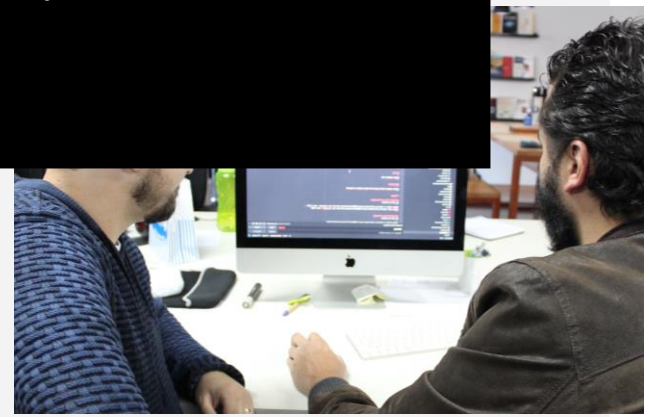
Como se comporta um processo MADURO em:
➔ Gerência de Requisitos

Elicitação/
Especificação
de requisitos



Projeto de software

Plano de testes



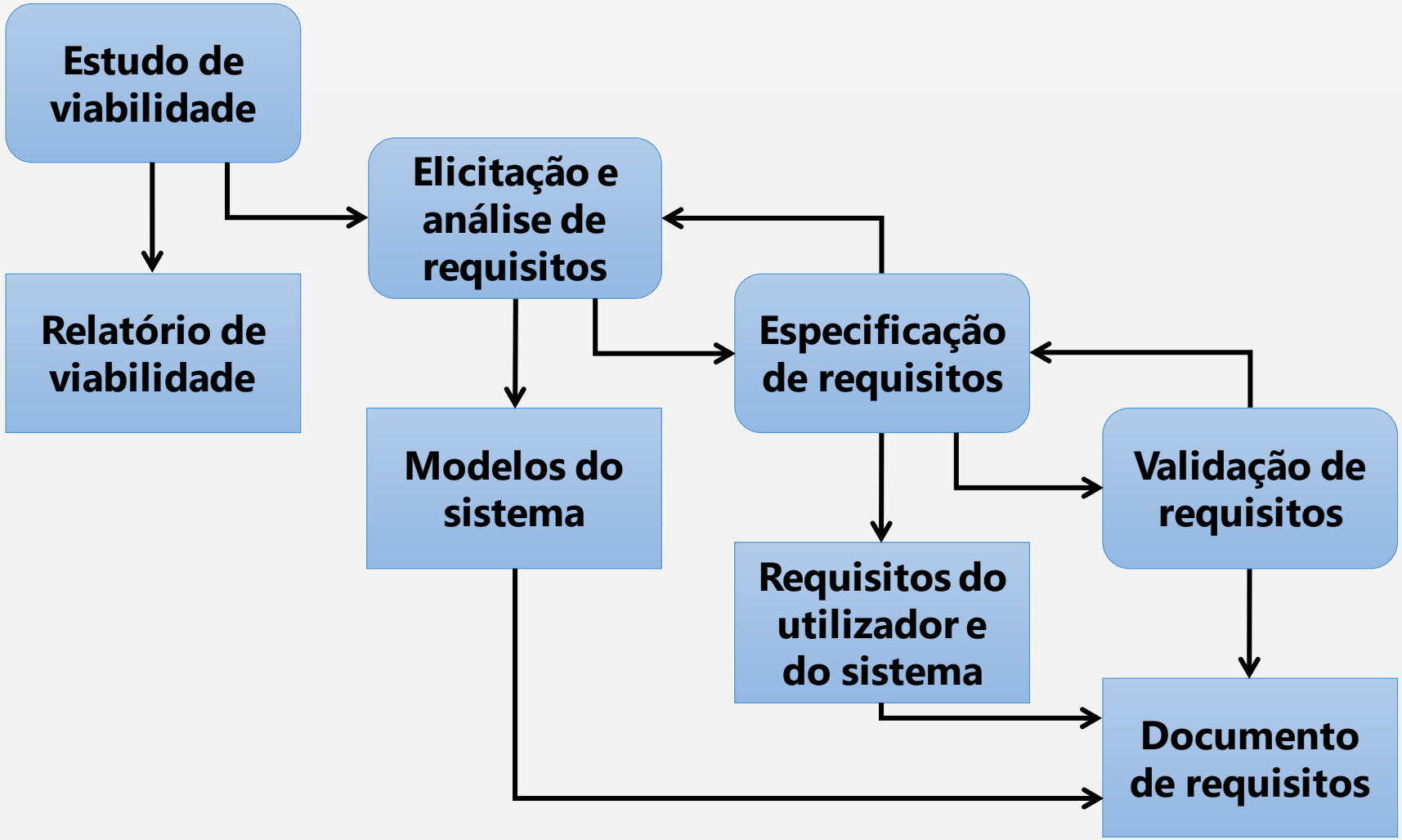
Codificação

Testes de software

Implantação

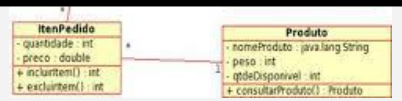
Manutenção
Evolução

GERÊNCIA DE PROJETOS



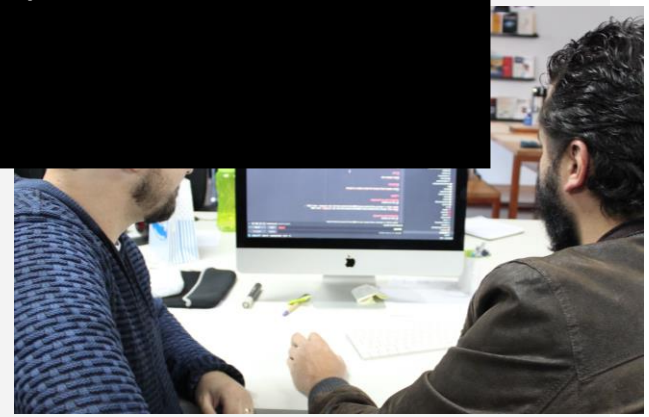
Gerência de Requisitos

Como se comporta um processo MATURO em:
➔ Gerência de Testes



Plano de testes

Projeto de software



Codificação

Elicitação/
Especificação
de requisitos

Manutenção
Evolução

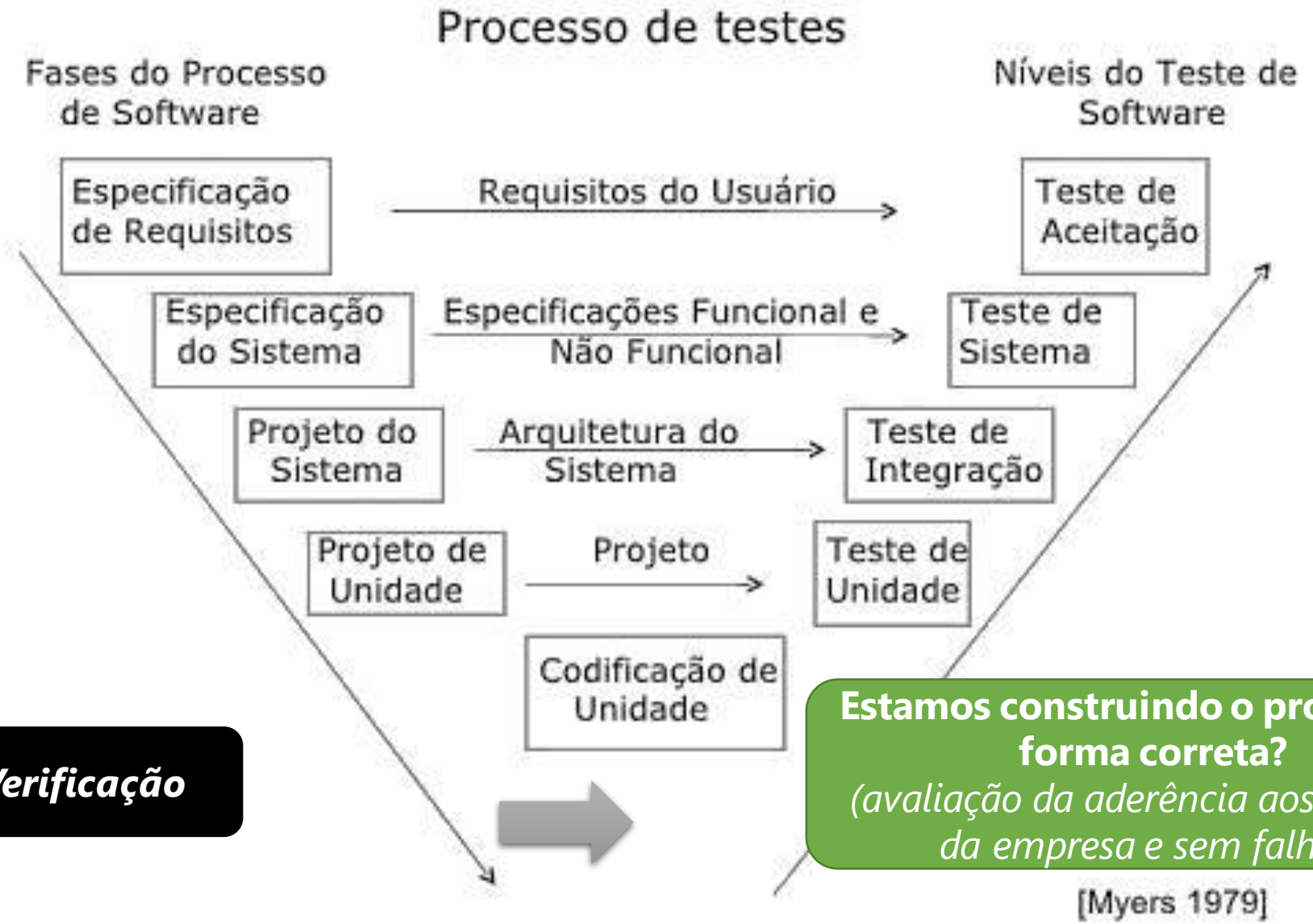
Implantação

Testes de software

GERÊNCIA DE PROJETOS

Validação

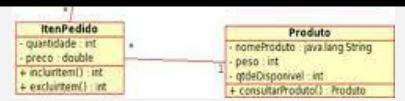
Estamos construindo o produto certo?
(avaliação do atendimento aos requisitos)

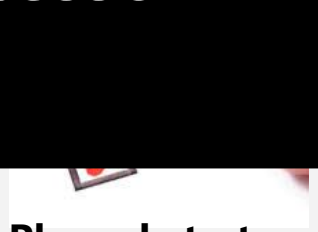


Verificação

Estamos construindo o produto de forma correta?
(avaliação da aderência aos padrões da empresa e sem falhas)

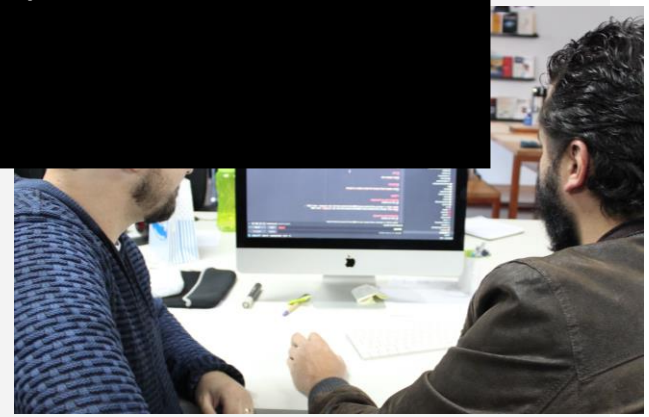
Como se comporta um processo MATURO em:
➔ Gerência de PROJETOS





Plano de testes

Projeto de software



Codificação

Elicitação/
Especificação
de requisitos

Manutenção
Evolução

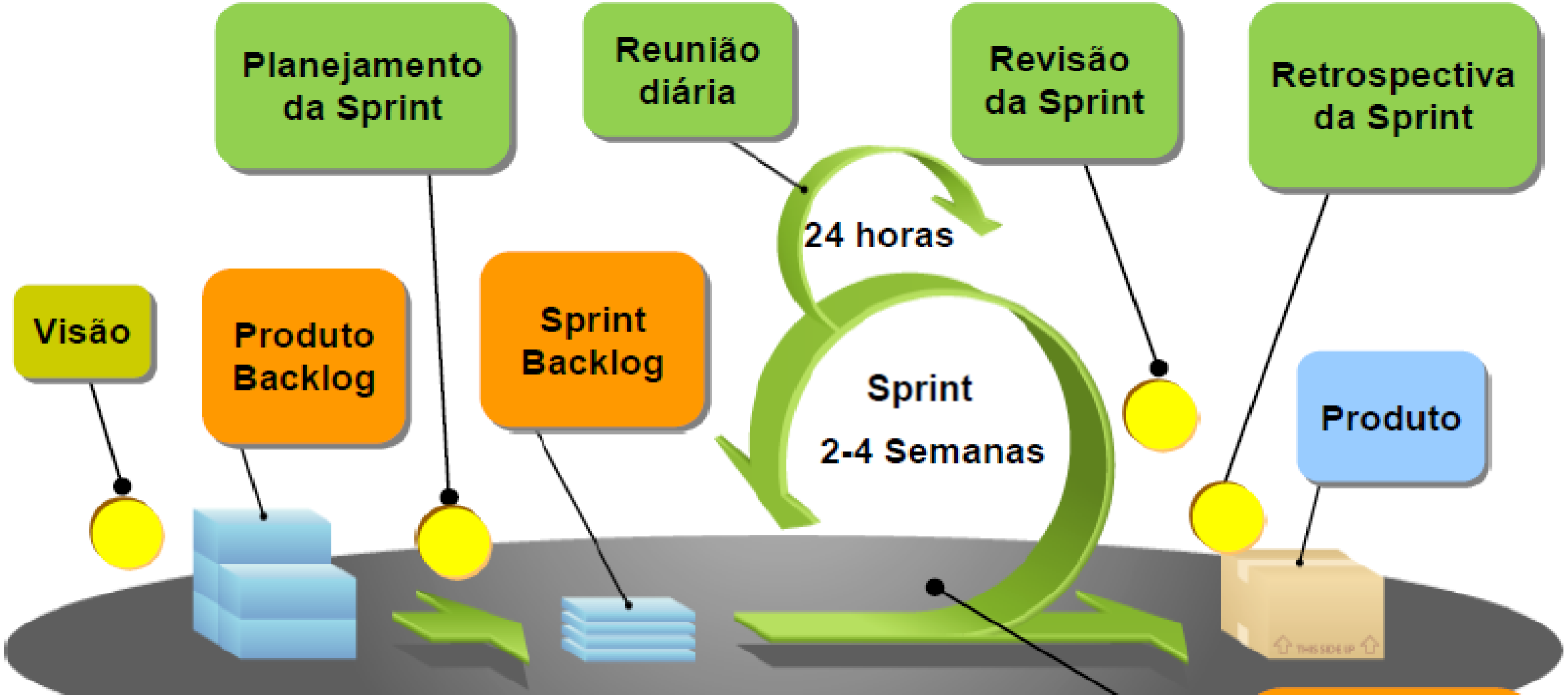
Implantação

Testes de software

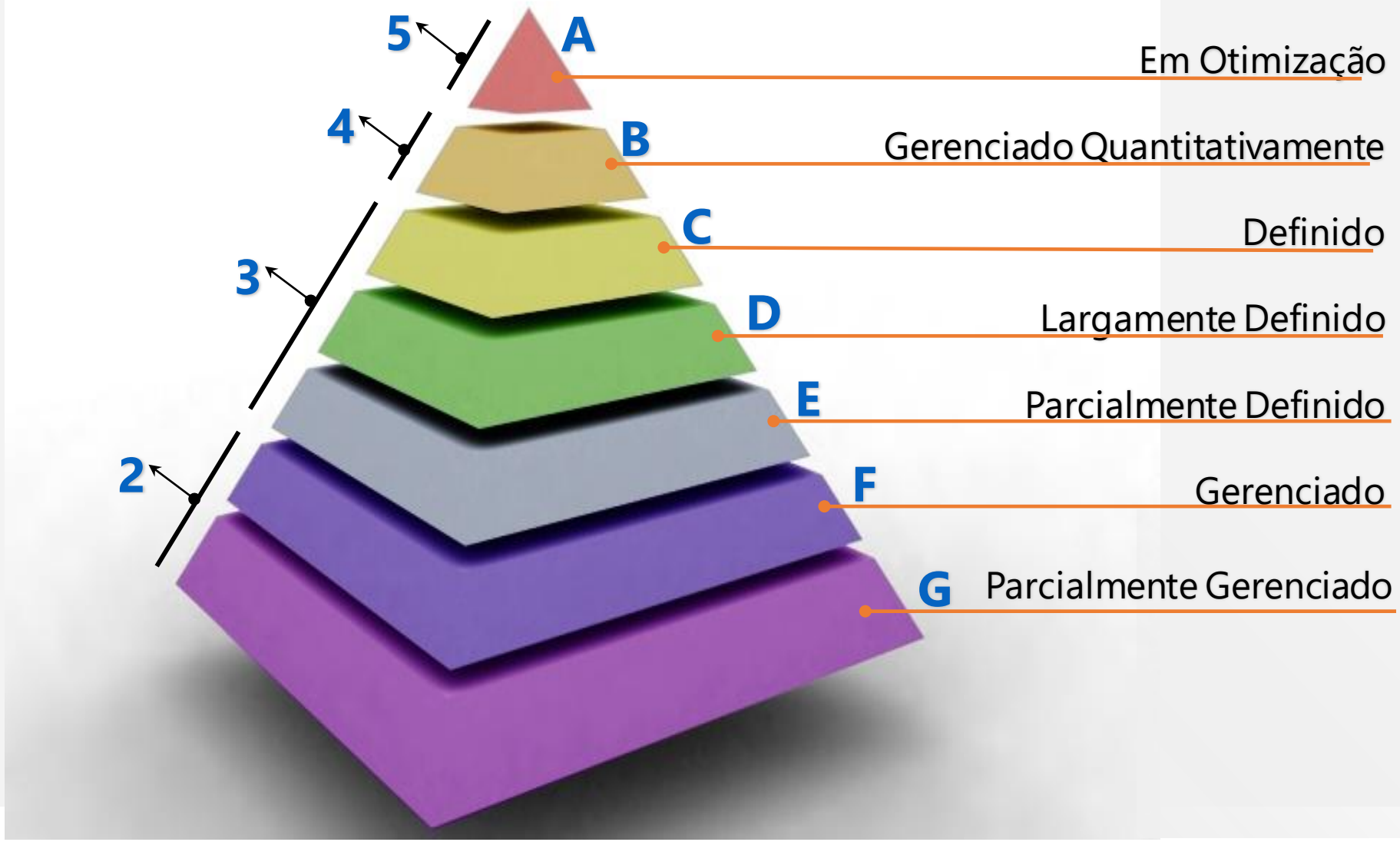
GERÊNCIA DE PROJETOS

SCRUM

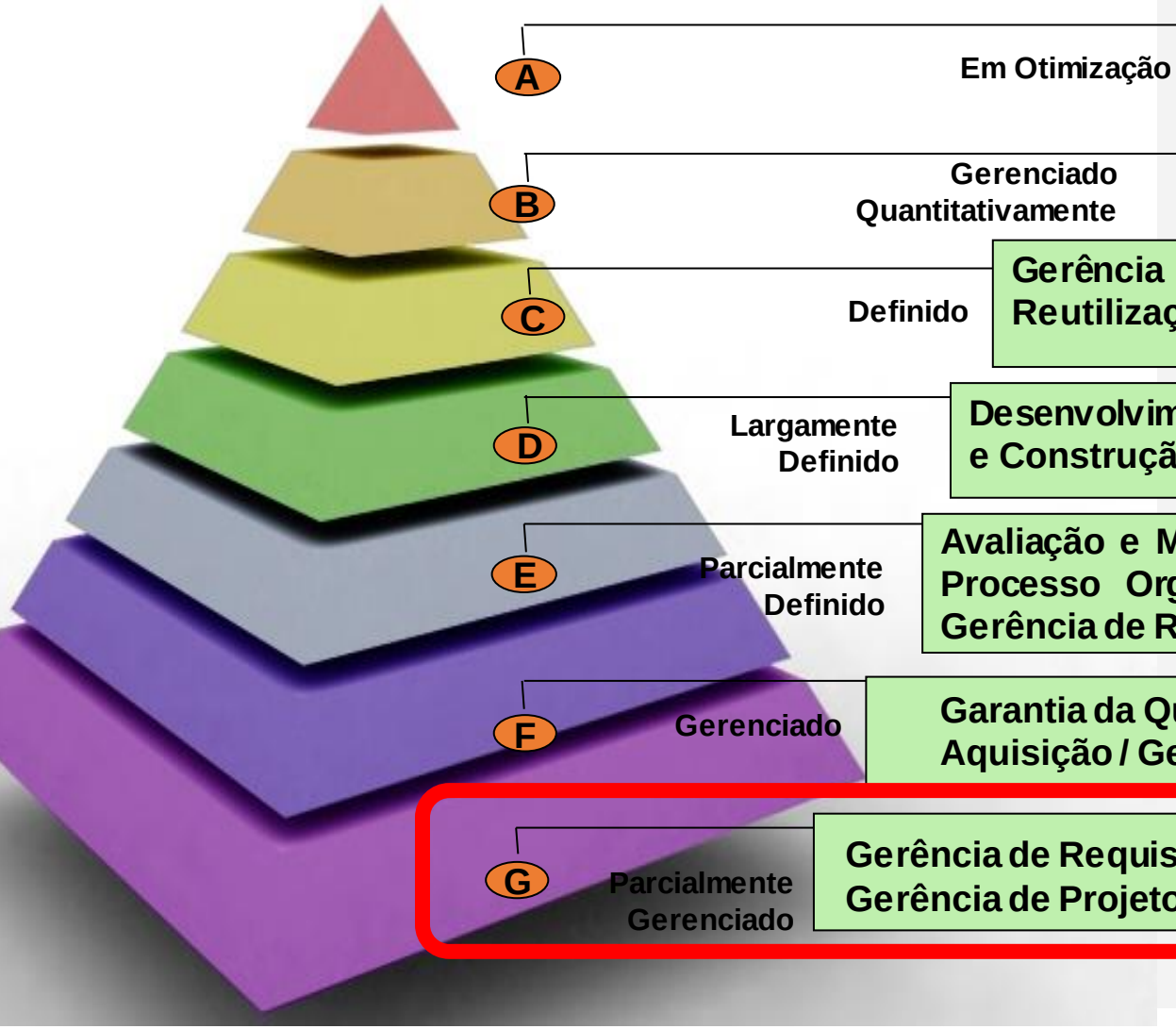
Método ágil para gestão de projetos



Níveis MPS.BR/ CMMI

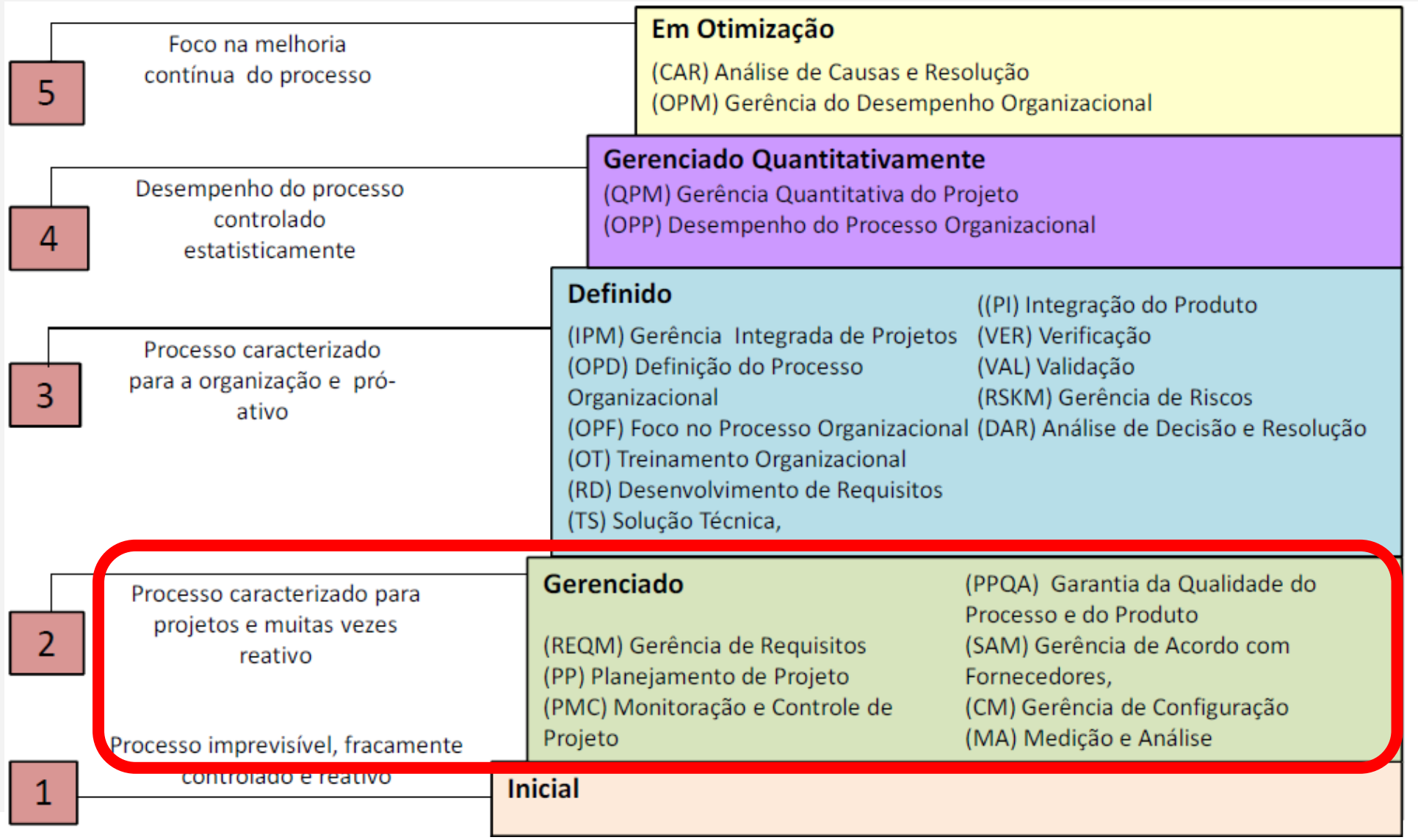


Níveis de Maturidade e Processos



A	Em Otimização	*(sem processos adicionais)*
B	Gerenciado Quantitativamente	Gerência de Projeto (evolução)
C	Definido	Gerência para Reutilização / Desenvolvimento para Reutilização / Gerência de Riscos
D	Largamente Definido	Desenvolvimento de Requisitos / Integração do Produto/ Projeto e Construção do Produto / Verificação / Validação
E	Parcialmente Definido	Avaliação e Melhoria do Processo Organizacional / Definição do Processo Organizacional / Gerência de Recursos Humanos / Gerência de Reutilização / Gerência de Projetos (evolução)
F	Gerenciado	Garantia da Qualidade / Medição / Gerência de Portfólio de Produto Aquisição / Gerência de Configuração
G	Parcialmente Gerenciado	Gerência de Requisitos Gerência de Projeto

Áreas de processo do CMMI



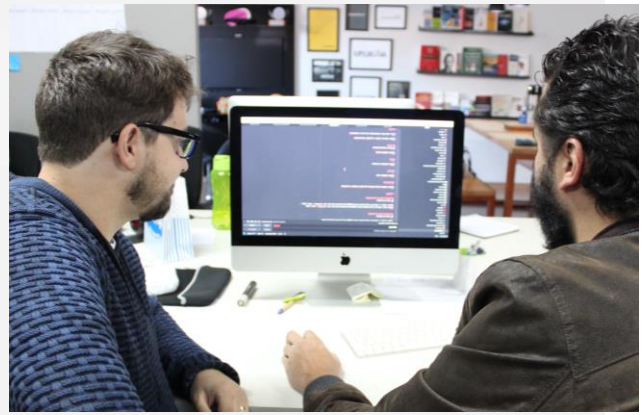
Engenharia de Software I

- Gerência de configuração

Elicitação/
Especificação
de requisitos

Projeto de software

Plano de testes



Codificação

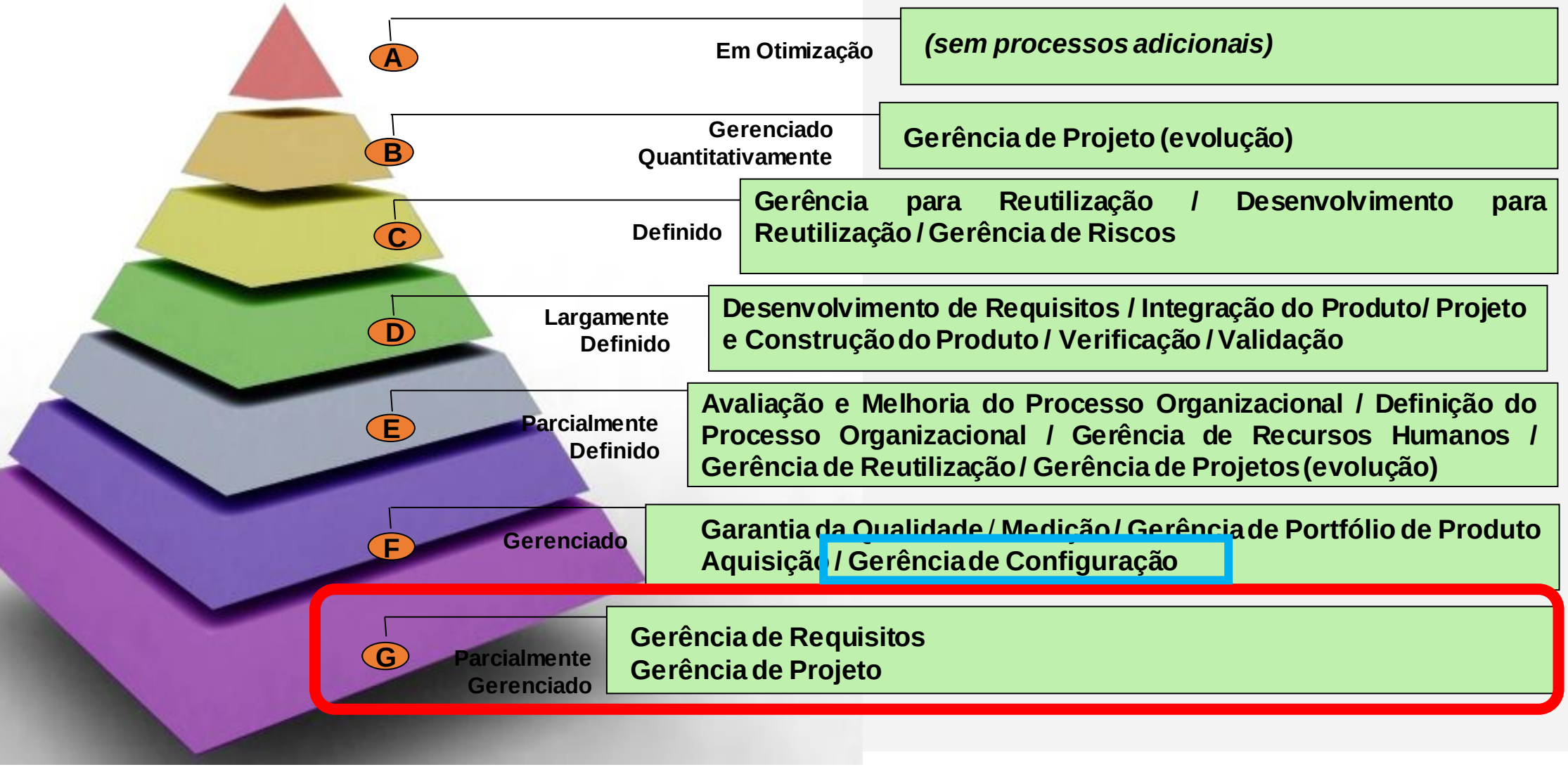
Testes de software

Implantação

Manutenção/
Evolução

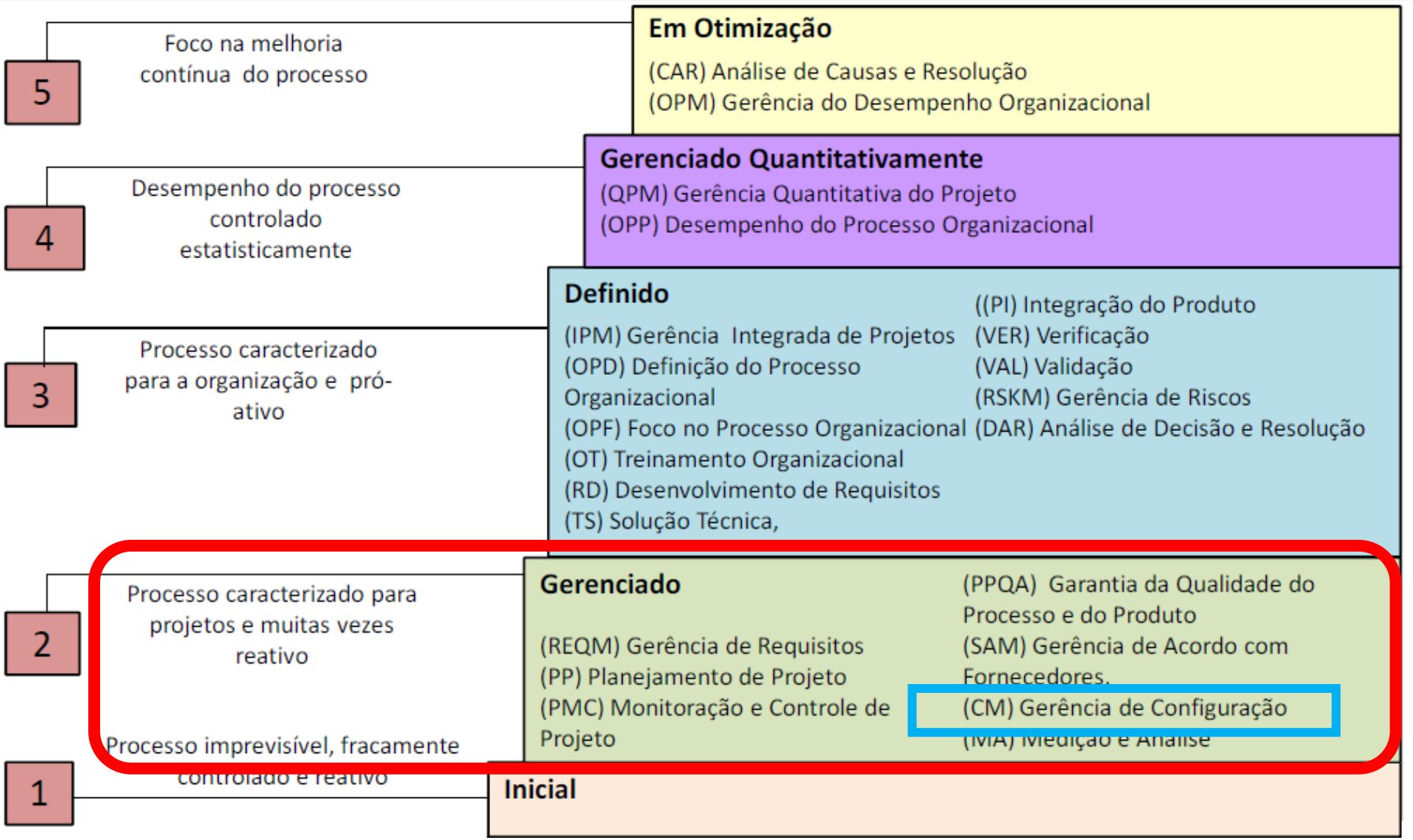
GERÊNCIA DE PROJETOS

Níveis de Maturidade e Processos



A	Em Otimização	*(sem processos adicionais)*
B	Gerenciado Quantitativamente	Gerência de Projeto (evolução)
C	Definido	Gerência para Reutilização / Desenvolvimento para Reutilização / Gerência de Riscos
D	Largamente Definido	Desenvolvimento de Requisitos / Integração do Produto/ Projeto e Construção do Produto / Verificação / Validação
E	Parcialmente Definido	Avaliação e Melhoria do Processo Organizacional / Definição do Processo Organizacional / Gerência de Recursos Humanos / Gerência de Reutilização / Gerência de Projetos (evolução)
F	Gerenciado	Garantia da Qualidade / Medição / Gerência de Portfólio de Produto Aquisição / Gerência de Configuração
G	Parcialmente Gerenciado	Gerência de Requisitos Gerência de Projeto

Áreas de processo do CMMI



Desafios 1
Gerenciamento de configuração

Representar e gerenciar a relação entre os artefatos

- Qualquer coisa que vamos desenvolver em computação envolve um conjunto relacionado de artefatos: software ou web page
- Desafio: como gerenciar (administrar) esta conjunto de artefatos e suas relações

Imagem.jpg

```
.slide-desc-bg {
        background: #334;
        opacity: 0.8;
        filter: alpha(opacity = 80);
}
.slide-desc-text {
        color: #fff;
        padding: 10px;
        text-align: left;
}
.slide-desc-text .slide-title {
        font-size: 1.5em;
        color: #eeee88;
        margin-bottom: 5px;
}
.slide-desc-text .slide-title a {
        color: #eeee88;
}
```

style.css

topo.php

Pagina index.php

Desafio 2
Gerenciamento de versões

Acompanhar e gerenciar a evolução dos artefatos ao longo do tempo

- Os artefatos evoluem como o passar do tempo
- Desafio é como acompanhar e gerenciar as alterações que vão acontecendo

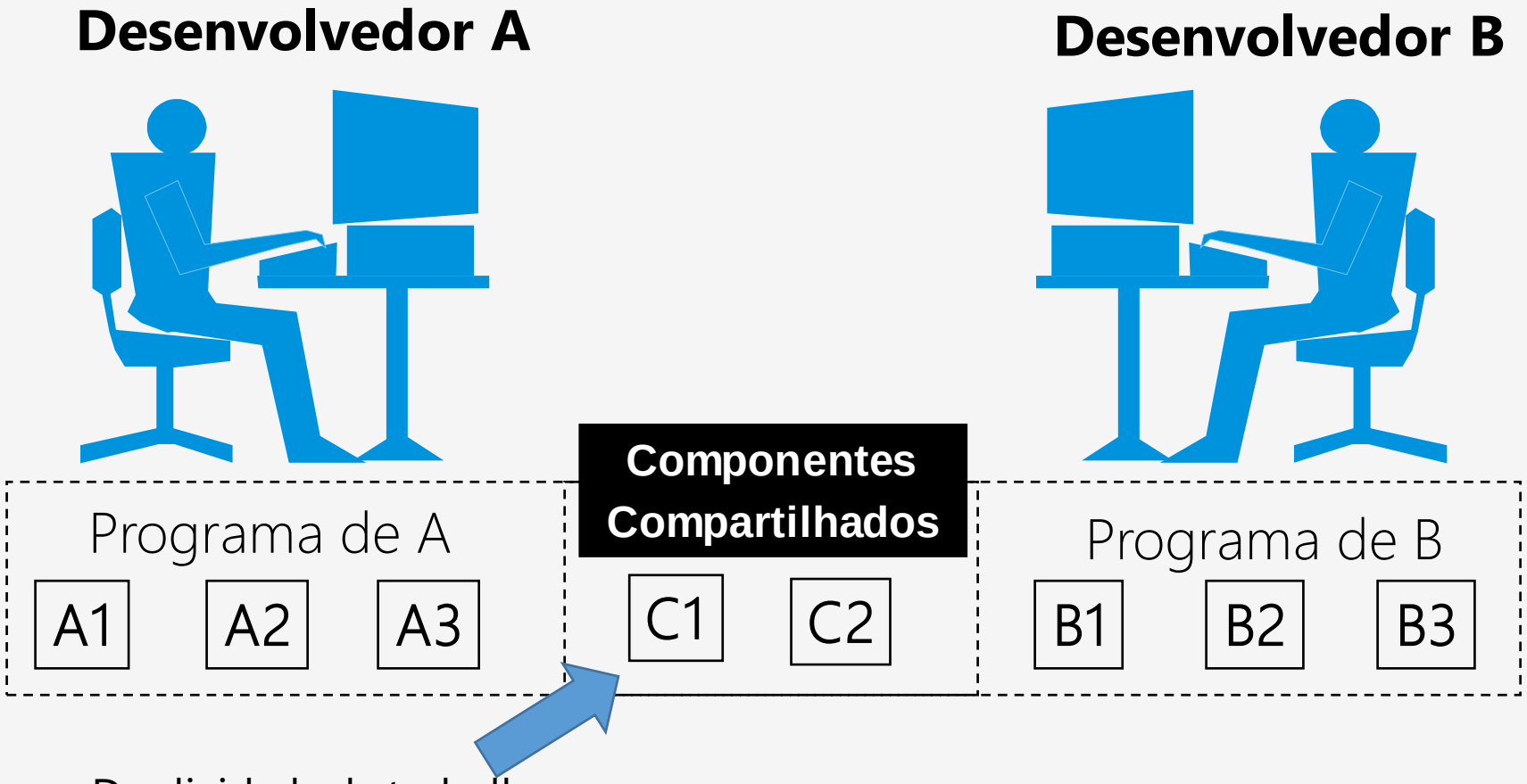
Imagem .jpg

Imagem .jpg

Imagem .jpg

Deve haver um registro sobre a mudanças dessa imagens

Problema



Duplicidade de trabalho
Arquivos sobrepostos
Trabalhos perdidos
Não tem como saber quem fez o que (histórico de mudanças)
Não é possível voltar numa versão anterior para corrigir um erro após a homologação

Problema dos dados compartilhados – Cenário comum

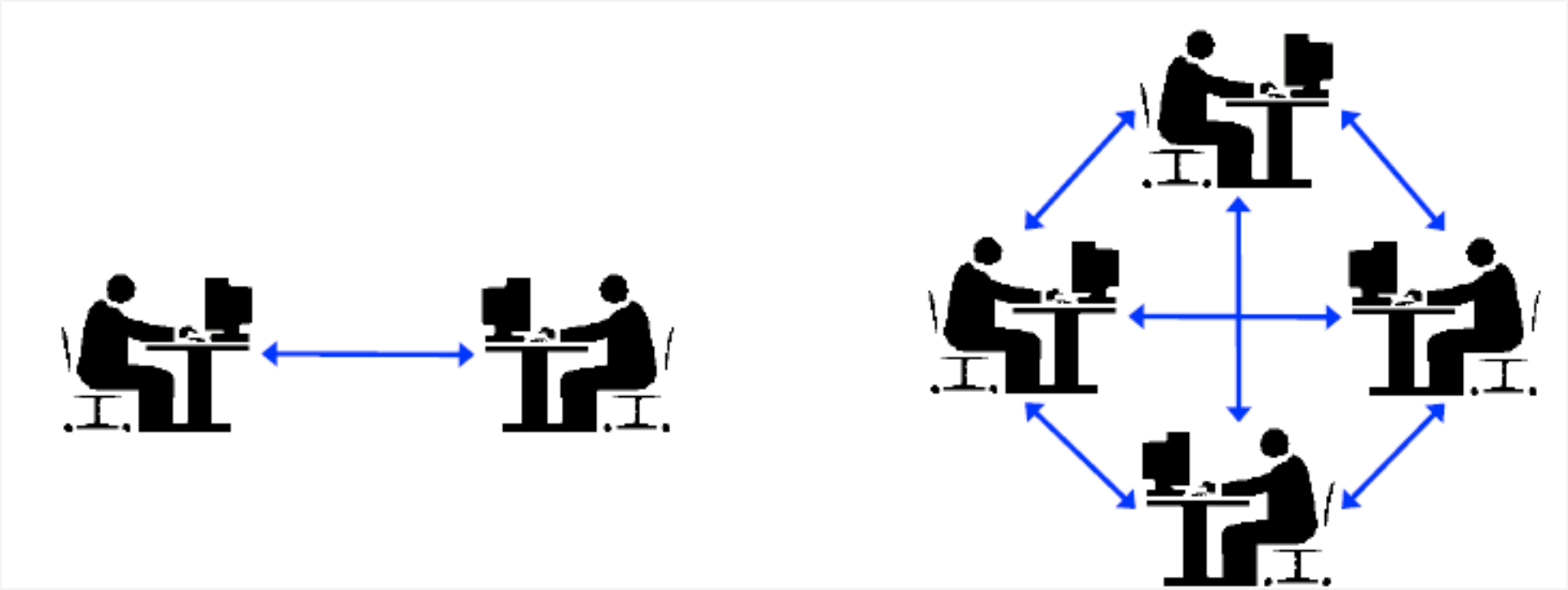
O desenvolvedor A modifica o componente compartilhado

Mais tarde, o desenvolvedor B realiza algumas alterações no mesmo componente

Ao tentar compilar o componente, erros são apontados pelo compilador, mas nenhum deles ocorre na parte que B alterou

O desenvolvedor B não tem a menor ideia sobre a causa do problema

Como surgiu a necessidade da Gerência de Configuração



- Normalmente se desenvolve software com várias pessoas trabalhando num mesmo projeto (equipe), em máquinas diferentes, mas que compartilham os mesmos códigos.
- Quanto maior o projeto maiores serão os problemas: maior será a complexidade do problema, mais código a ser gerado, mais pessoas gravando e compartilhando arquivos.

Problemas pela falta de GC

Perda de código fonte

Programas inesperadamente param de funcionar

- Alguém sobre escreveu numa versão estável

Impossível saber qual foi a evolução do desenvolvimento de um programa

Impossível saber quem, porque e quando foram realizadas as alterações no sistema

Uma classe simplesmente sumiu

Bugs corrigidos aparecem inesperadamente

Como resolver esses problemas?

Como resolver esses problemas?

➔Com uma boas práticas de Gerência de Configuração

Gerenciamento de configuração

Gerência de configuração (GC) é o processo de **identificar**, **organizar** e **controlar** modificações ao software sendo construído (MPS.BR)

É responsável por gerenciar como o software é modificado e construído através de técnicas que incluem **controle de mudanças**, **controle de versão**, **rastreabilidade na construção dos objetos** e **geração de versões de software**.

Por que Gerência de Configuração?

Evolução do software ➔ MUDANÇAS

75% do custo total do ciclo de vida do software é com manutenção.

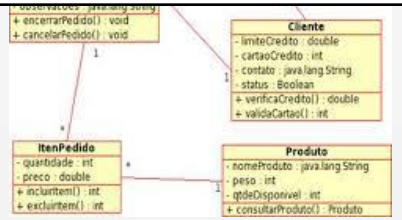
20% do tempo para consertar erros.

80% do tempo para modificações nos requisitos:

- Requisitos funcionais
- Regras de negócio
- Reengenharia da aplicação.

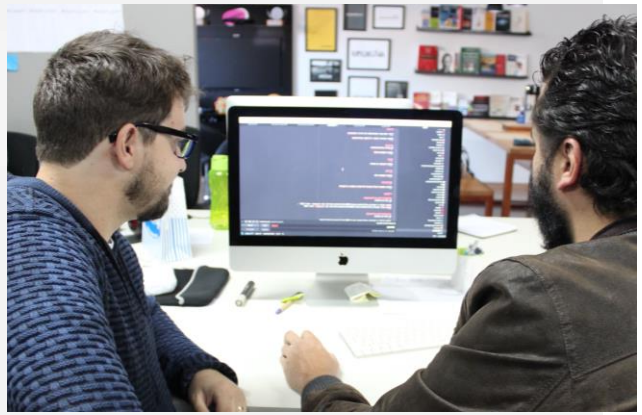
Quais artefatos irão evoluir?

Elicitação/
Especificação
de requisitos



Projeto de software

Plano de testes



Codificação

Testes de software

Implantação

Manutenção/
Evolução

GERÊNCIA DE PROJETOS

Configuração de software

Um projeto de software produz vários artefatos:

- Código fonte
- Programas executáveis
- Biblioteca de componentes
- Dados dos testes
- Documentação de requisitos
- Scripts do banco de dados
- Imagens, ícones ...
- Documentação do projeto do software: casos de uso, diagramas de classes, diagrama entidade-relacionamento, etc

➔ Esse conjunto de itens são chamados de **itens de configuração de software**

➔ Quando se criar um **processo** de gerenciamento desses itens se chama de **gerenciamento de configuração**

Responsabilidades da Gerência de configuração

- Identificação dos itens de configuração
- Obter controle das alterações e das versões de cada um desses itens de configuração, de modo efetivo e contínuo
- Definir o ambiente de desenvolvimento (ambiente de desenvolvimento e de homologação)
- Política para controle de versões (como o software será versionado através de builds, releases, versões no decorrer do tempo)
- Garantir a integridade e rastreabilidade dessas versões desses itens
- Garantir a consistência dos artefatos – determina onde as coisas serão guardadas e de que forma
- Facilitar a integração entre as partes dos sistema
- Definir procedimentos para solicitação de mudança
- Todo o histórico de mudanças deve ser recuperável e auditável

Benefícios

- Aumento da disciplina no processo de desenvolvimento e da organização (eficiência do processo)
- Aumento de produtividade e de eficiência no desenvolvimento (evitando problemas de duplicidade de trabalho, sobreposição de arquivos, visualizar as mudanças)
- Redução de defeitos (consequência do ambiente organizado)
- Menores custos de manutenção (pois eu não se perde código fonte e não ocorre retrabalho)
- Maior rapidez na identificação e correção dos problemas (consequência dos outros itens)
- Uma vez que se tem um ambiente de desenvolvimento organizado possibilitando o crescimento do software de maneira organizada, ocorre o aumento da produtividade do desenvolvimento, em consequência irá reduzir os defeitos, etc.

O que é a gerencia de configuração

➔ Gerenciamento de configuração é muito mais que controlar as versões, é controlar as mudanças do projeto

- Gerencia de configuração é o processo de identificar, organizar e controlar as modificações de um software em construção

- É um fluxo de apoio para todo o projeto (não apenas de apoio para a etapa de desenvolvimento)

➔ É um dos tópicos mais importantes da Engenharia de software

Itens de configuração

É o que se deseja controlar ao longo de ciclo de vida de um projeto de software. É o menor item dentro da GC:

- Código fonte
- Programas executáveis
- Biblioteca de componentes
- Dados dos testes
- Documentação de requisitos
- Scripts do banco de dados
- Imagens, ícones ...
- Documentação do projeto do software: casos de uso, diagramas de classes, diagrama entidade-relacionamento, etc
- etc

➔ É um engano pensar que a gerencia de configuração deve se preocupar apenas com os códigos fontes do sistema

Responsabilidades da Gerência de configuração

Definir o ambiente de desenvolvimento (ambiente de desenvolvimento, de homologação)

Política para controle de versões (como o software será versionado através de builds, releases, versões no decorrer do tempo)

Garantir a consistência dos artefatos – determina onde as coisas serão guardadas e de que forma

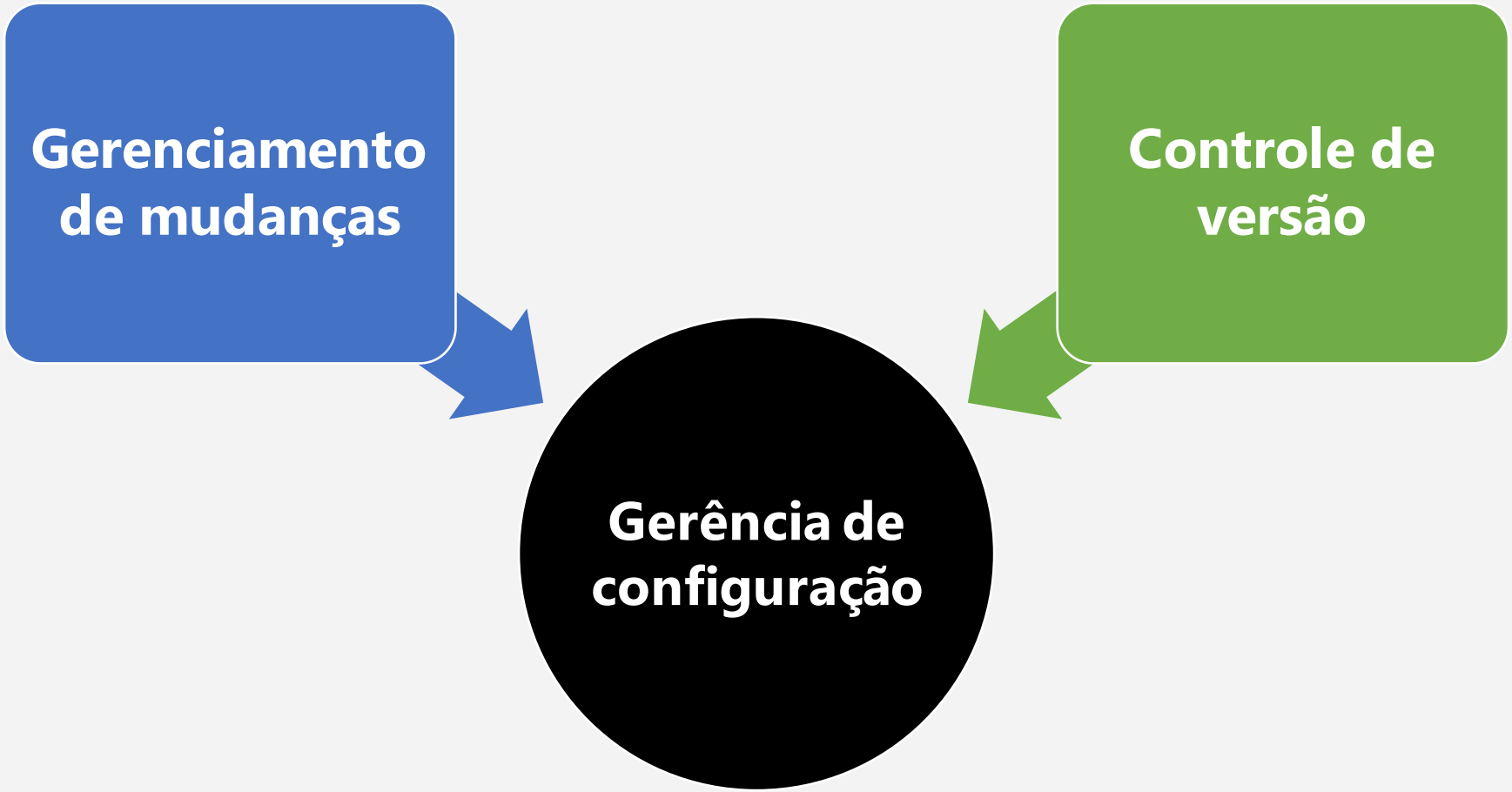
Definir procedimentos para solicitação de mudança

Administrar o ambiente e auditar mudanças

Facilitar a integração entre as partes dos sistema

Porque o sistema mudou?

Quais foram as mudanças?

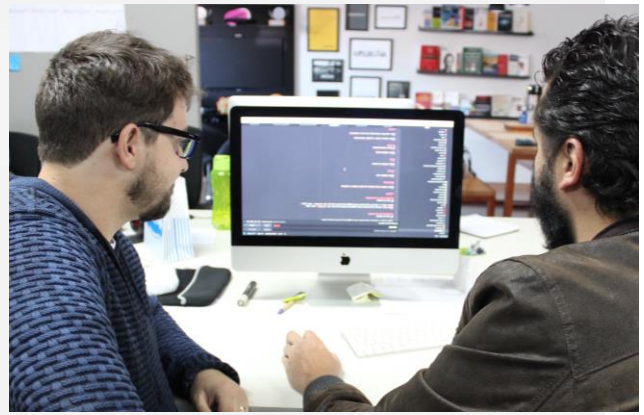


Gerenciamento de mudanças

Elicitação/
Especificação
de requisitos

Projeto de software

Plano de testes



Codificação

Testes de software

Implantação

Manutenção/
Evolução

GERÊNCIA DE PROJETOS

Gerenciamento de mudanças

Durante a vida útil de um sistema as necessidades organizacionais e os requisitos desse mudam, *bugs* precisam ser reparados e os sistemas têm de se adaptar às mudanças em seu ambiente.

O gerenciamento de mudanças **visa garantir que a evolução do sistema seja um processo gerenciado**

Gerenciamento de mudanças

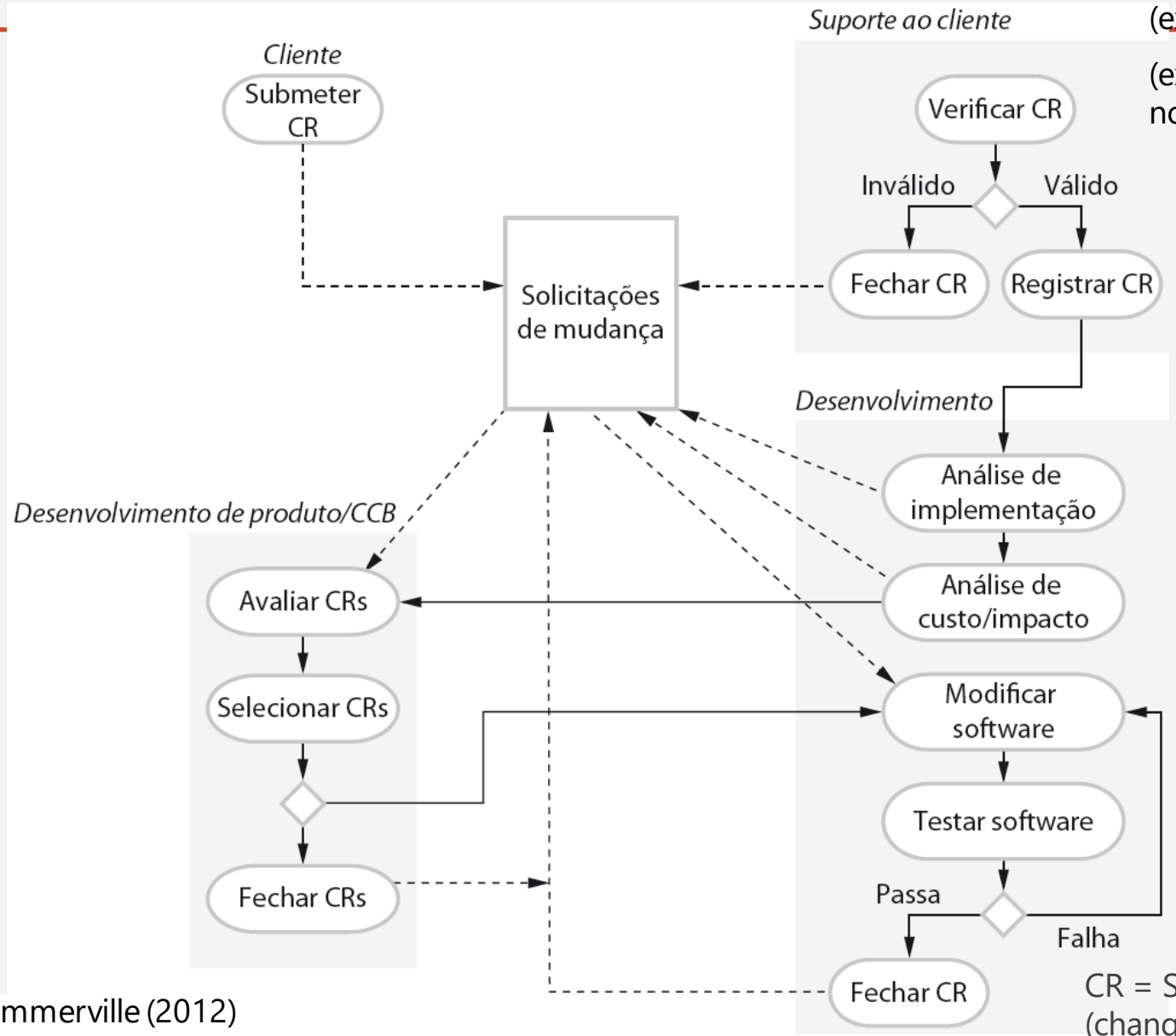
O gerenciamento de mudanças **visa garantir que a evolução do sistema seja um processo gerenciado** e que **seja dada prioridade às mudanças mais urgentes e de custo-benefício.**

O processo de gerenciamento de mudanças **está relacionado com a análise dos custos** e **benefícios das mudanças** propostas, a aprovação das **mudanças que valem a pena** e o **acompanhamento das alterações** nos componentes do sistema.

O processo de gerenciamento de mudanças (Sommerville)

(ex. correção de um bug)

(ex. Solicitação de nova funcionalidade)



CR = Solicitação de mudança (change request)

Fonte: Sommerville (2012)

Fatores na análise de mudança

Quais as consequências de não realizar a mudança

Os benefícios da mudança

O número de usuários afetados pela mudança (impacto nos clientes)

Os custos de se fazer a mudança

O ciclo de release de produto

Um formulário de solicitação de mudança parcialmente concluído (a)

Projeto: SICSA/AppProcessing
Número: 23/02
Solicitante de mudança: I. Sommerville
Data: 20/jan./2009
Mudança solicitada: O *status* dos requerentes (rejeitados, aceitos etc.) deve ser mostrado visualmente na lista de candidatos exibida.

Analista de mudança: R. Looek
Data da análise: 25/jan./2009
Componentes afetados: ApplicantListDisplay, StatusUpdater

Componentes associados: StudentDatabase

Avaliação de mudança: Relativamente simples de implementar, alterando a cor de exibição de acordo com ***status***. Uma tabela deve ser adicionada para relacionar ***status*** a cores. Não é requerida alteração nos componentes associados.

Um formulário de solicitação de mudança parcialmente concluído (b)

Prioridade de mudança: Média
Implementação de mudança:
Esforço estimado: 2 horas
Data para equipe de aplicação de SGA: 28/jan./2009 **Data de decisão do CCB**: 30/jan./2009
Decisão: Aceitar alterar. Mudança deve ser implementada no *Release* 1.2
Implementador de mudança: **Data de mudança:**
Data de submissão ao QA: **Decisão de QA:**
Data de submissão ao CM:
Comentários:

Ferramentas de controle de mudanças

Open source:

Redmine

Mantis

Bugzilla

Trac

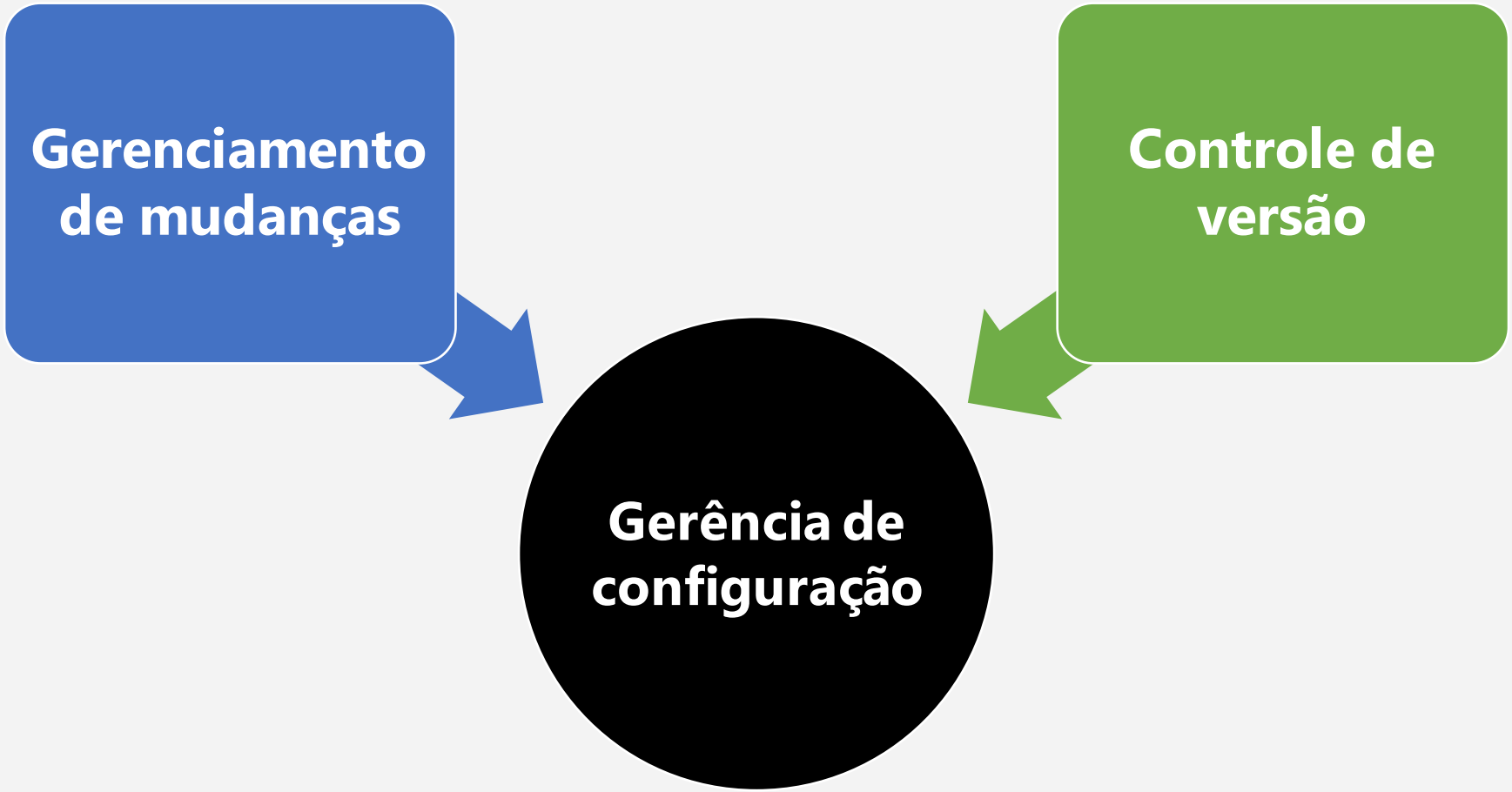
Comerciais:

Jira

IBM Rational ClearQuest

Porque o sistema mudou?

Quais foram as mudanças?



Controle de versões

Gerenciamento de versões

O **gerenciamento de versões** (VM – *Version Management*) **é o processo de manter o controle das diferentes versões dos componentes do software ou itens de configuração** e os sistemas em que esses componentes são usados.

Também envolve **assegurar que as mudanças sejam feitas por desenvolvedores diferentes** para que **essas versões não interfiram umas com as outras**.

Codelines e Baselines

O gerenciamento de versões pode ser pensado como o processo de gerenciamento de ***codelines*** e ***baselines***.

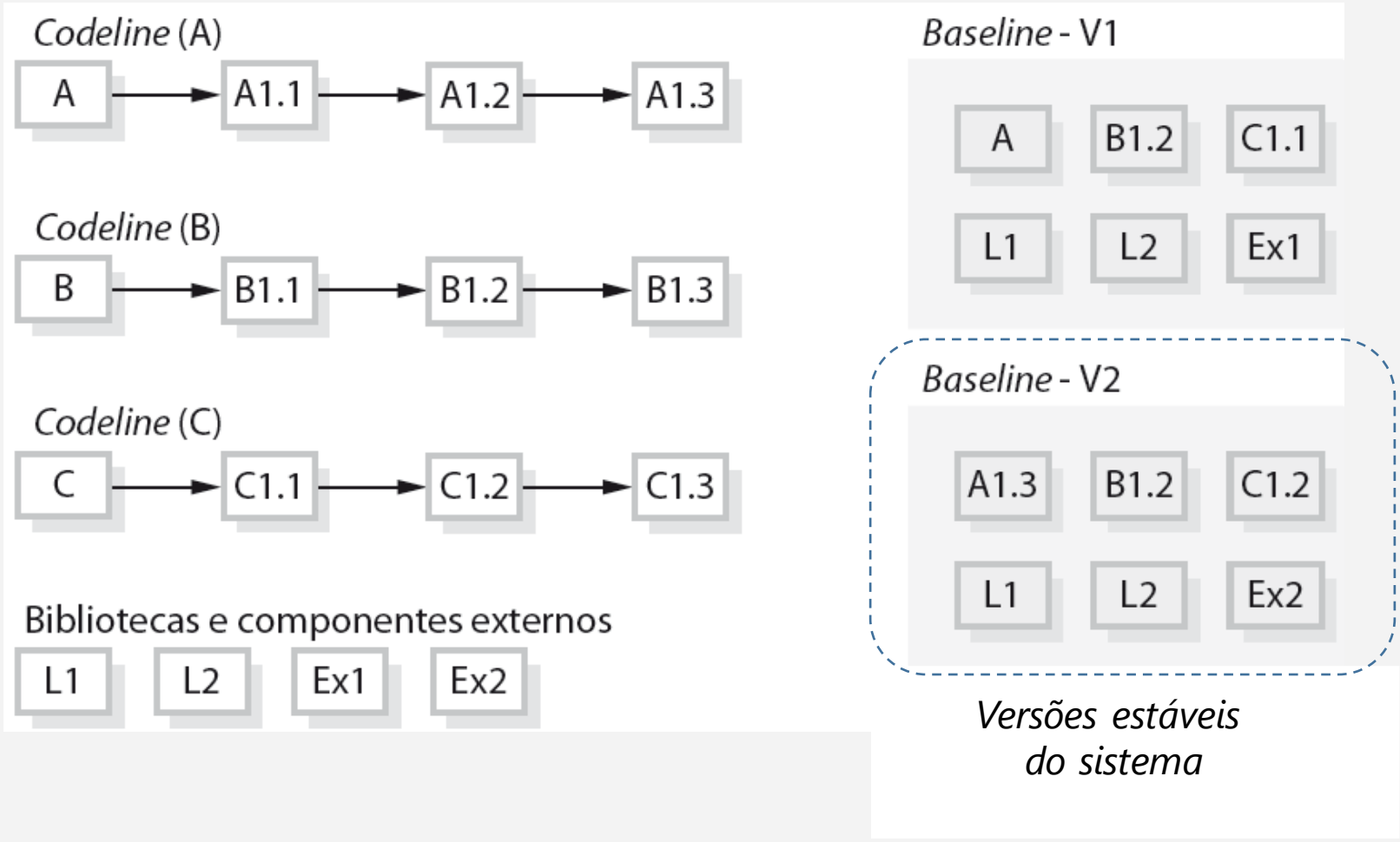
Um **codeline** é uma sequência de versões de código-fonte com as versões posteriores na sequência derivadas de versões anteriores.

Codelines normalmente se aplicam a componentes de sistemas de modo que existem diferentes versões de cada componente.

Um **baseline** é uma definição de um sistema específico.

Um **baseline**, portanto, especifica as versões dos componentes que estão incluídos no sistema além de uma especificação das bibliotecas usadas, arquivos de configuração, etc.

Codelines e Baselines



Baseline

Baseline (linha de base):

- É a configuração do software em um ponto do tempo (faz uma marco como uma bandeira em determinados momento do tempo do projeto, ou como tirar uma foto para registrar a situação num determinado momento)
- As baselines representam conjuntos de itens de configuração formalmente aprovados que servem de base para as etapas seguintes de desenvolvimento
- Serve como base para os passos posteriores de desenvolvimento
- Baselines são considerados marcos no processo de desenvolvimento
- **Definição: é uma coleção de versões de componentes que compõe um sistema**.

Razões para criar uma baseline:

- Reproducibilidade: A habilidade de reproduzir uma versão anterior do sistema (analogia de voltar o backup)
- Rastreabilidade: estabelece uma relação predecessor-sucessor entre os artefatos do projeto (analogia de percorrer o caminho até chegar a situação atual)
- Controle de mudanças: referencial para comparações, discussões e negociações (entender as decisões tomadas ao autorizar a mudança)

Build

Representa alguma **versão incompleta** do sistema em desenvolvimento, **mas com certa estabilidade** (não é uma entrega, mas uma versão menor até chegar na entrega final), conforme evoluímos (cada passo) no desenvolvimento vamos criando biulds, quando concluir tudo então vira uma release.

Espaço para integração de funcionalidades

Releases

- Versão de um sistema validada após os diversos tipos de teste
- Produto de software
- Supostamente sem erros (está pronto e foi testado)
- Entre ao cliente ou ao mercado (é uma entrega)
- Implantado no cliente

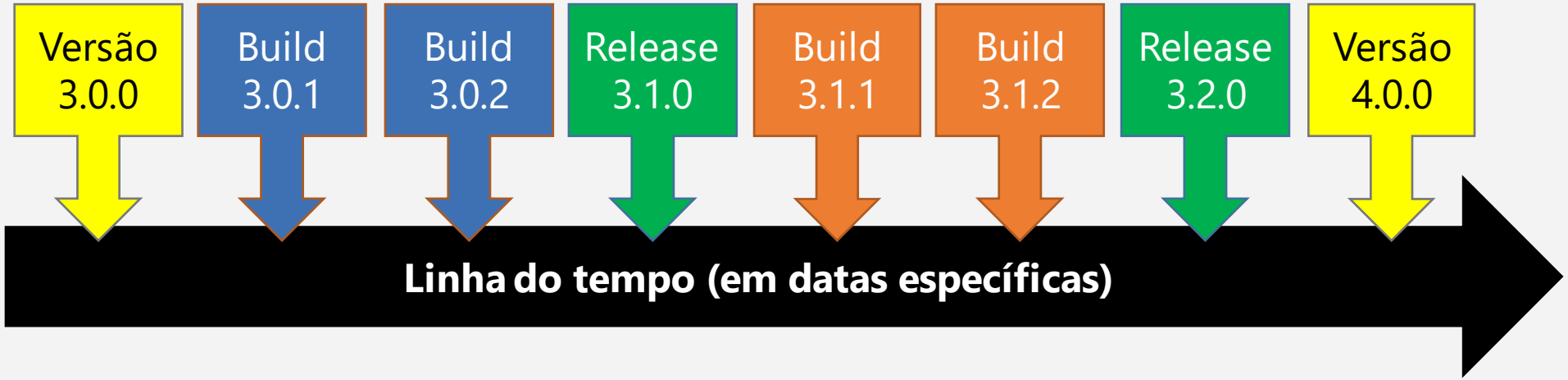
➔Diferença: release deve funcionar corretamente, builds não está completo e pode haver problemas

Versão

Uma política de evolução do produto

Evoluiu tanto que se tornou uma nova versão

Builds, Releases e Versões



Três níveis de gerenciamento de configuração

3.1.2 → Versão 3

3.**1**.2 → Release 1

3.1.**2** → Build 2

Ferramentas para controle de versão

Aplicativos:

- Git
- Github
- Gitlab

Sistema de controle de versões
Quem iniciou o desenvolvimento do Git foi Linus Torvals para facilitar o desenvolvimento do kernel linux

Serviço de hospedagem de projetos que usam Git
Serve como rede social de desenvolvedores pois podem disponibilizar seus projetos

➔ **Tutorial do Git**