GERÊNCIA DE CONFIGURAÇÃO E EVOLUÇÃO

AULA 2

Profª Adriana Bastos da Costa

CONVERSA INICIAL

Já vimos que o controle da criação e das alterações dos componentes de um *software* é essencial para manter a organização do desenvolvimento de um projeto de *software*, para evitar retrabalho ou perdas de dados. Esse controle é fundamental, principalmente quando trabalhamos em equipes, em que o desenvolvimento do *software* ocorre de forma colaborativa e o código é propriedade coletiva da equipe de desenvolvimento. Quanto mais integrado e padronizado estiver o código, melhor e mais produtivo será o desenvolvimento.

E, para que o gerenciamento das configurações seja o melhor possível, existem modelos de qualidade que ajudam a definir o processo adequado ao projeto. E é justo sobre os modelos para definir o processo de gerenciamento de requisitos que vamos conversar nesta etapa, discutindo sobre a importância da gerência de configuração para a qualidade do *software*.

Esta etapa está dividida em cinco tópicos principais, sendo eles:

1. O que é o Modelo Integrado de Capacidade e Maturidade (CMMI)
2. O que é a Melhoria de Processo do Software Brasileiro (MPS.BR)
3. Conhecendo mais modelos, como a Melhoria de Processo de Software e Determinação de Capacidade (Spice) e o Processo Unificado da Rational (RUP)
4. Planejamento das mudanças e da configuração e controle da construção, das modificações e do versionamento do projeto de *software*
5. Papéis e responsabilidades do comitê de mudança

Vamos explorar como os processos ajudam a organizar um projeto de *software*, organizando o trabalho e melhorando a qualidade do que é construído.

TEMA 1 – O QUE É O CMMI

Já vimos que construir um *software* de qualidade e que atenda à necessidade dos clientes não é uma tarefa trivial e, muito menos, simples. É por isso que modelos de qualidade são definidos. Eles ajudam a pensar de forma sistemática uma forma de organizar todas as etapas de um processo.

O CMMI é um modelo que ajuda a definir um processo de *software* completo e adaptável às demandas dos projetos e das empresas que o utilizam sem perder as boas práticas de mercado, na busca de alcançar a qualidade. Ele

tem como objetivo, portanto, ser um modelo de melhoria dos processos de *software*.

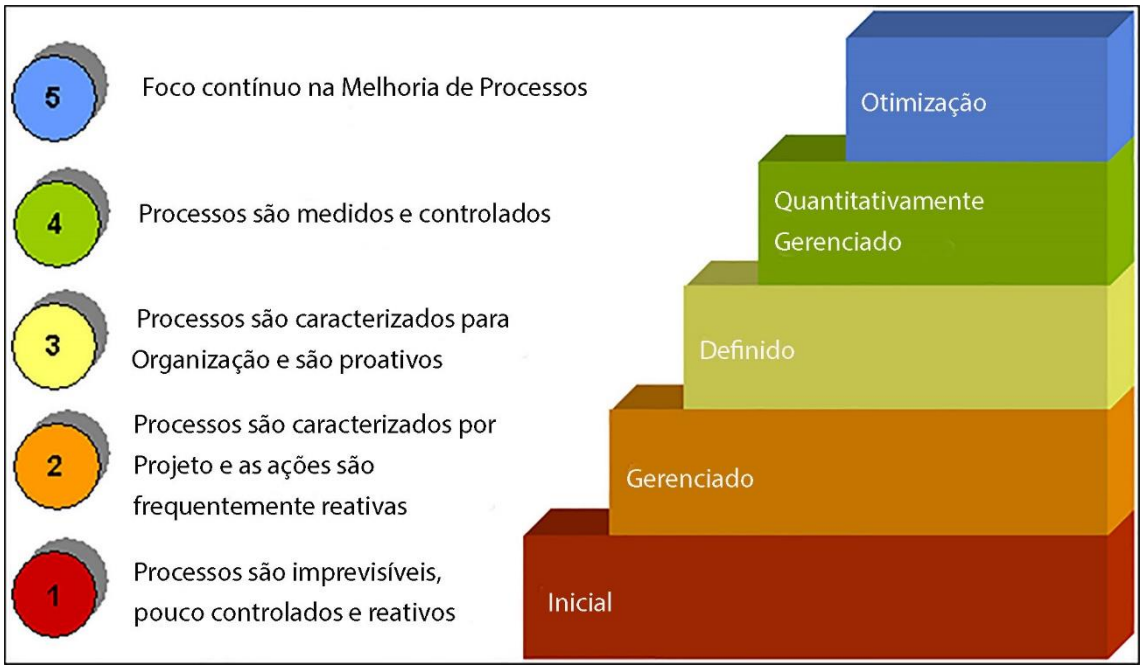
O CMMI se subdivide em três modelos:

1. *CMMI for Development* (CMMI para desenvolvimento – CMMI-DEV): define melhores práticas para desenvolvimento de produtos e serviços de *software*.
2. *CMMI for Services* (CMMI para serviços – CMMI-SVC): define melhores práticas para fornecimento de serviços de tecnologia.
3. *CMMI for Acquisition* (CMMI para aquisição – CMMI-ACQ): define melhores práticas para aquisição de produtos e serviços de tecnologia.

Os três modelos são mantidos pelo CMMI Institute, sob os cuidados da Information Systems Audit and Control Association (Isaca), localizada nos EUA. A Isaca é um órgão que cuida de uma série de certificações e modelos e agora está também mantendo o CMMI. O CMMI foi criado, há cerca de 25 anos, a pedido do Departamento de Defesa dos EUA, para ajudar na seleção de fornecedores de *software*. Atualmente, ele é um padrão internacional reconhecido como um modelo voltado para boas práticas de desenvolvimento de *software* em empresas de várias áreas de negócio e porte.

O modelo é organizado em níveis de maturidade, sendo um modo evolutivo de atingimento da maturidade nos processos necessários para planejar e executar de maneira adequada a construção de um *software*. Além disso, cada nível de maturidade é organizado em áreas de processo, com práticas específicas a cada uma dessas áreas, de forma a cobrir todo o ciclo de vida do desenvolvimento de um *software*. O conjunto de níveis de maturidade apresentados pelo CMMI funcionam como um processo de melhoria contínua, no qual algumas práticas do processo podem ser analisadas e melhoradas, de forma contínua, de acordo com as necessidades dos projetos, clientes e mercado. O CMMI é, portanto, organizado da forma como nos mostra a Figura 1, segundo níveis de maturidade que vamos detalhar em seguida.

Figura 1 – Níveis de maturidade do CMMI



Fonte: http://www.isdbrasil.com.br/o-que-e-cmmi.php

1.1 Níveis de maturidade

Os níveis de maturidade são, portanto, a forma evolutiva como o CMMI organiza a prática de uma empresa ao desenvolver *softwares*, que deve melhorar continuamente com a aplicação, a análise de resultados e processos. Se fizéssemos uma analogia com a nossa vida, os níveis de maturidade são equivalentes à nossa evolução como pessoas. Nascemos e vamos aprendendo uma série de coisas, passamos de bebês para crianças, de crianças para adolescentes, de adolescentes para adultos e de adultos para idosos, sempre aprendendo novas coisas e evoluindo na arte de viver. Em alguns momentos, precisamos reaprender coisas, experimentar coisas novas, analisar os resultados disso e buscando melhorar continuamente. É dessa mesma forma que funcionam os níveis de maturidade, no CMMI.

As principais características de cada um dos níveis de maturidade são:

- Nível 1 – inicial: nível considerado inicial porque corresponde à imaturidade organizacional, quando os processos são improvisados e geralmente não são seguidos em todos os projetos, ficando sob a responsabilidade do gerente de projetos seguir ou não um processo organizado. Por isso mesmo, muitas vezes compromissos de prazo e

custo não são cumpridos, uma vez que o planejamento não é feito com base em estimativas confiáveis. Os projetos geram, assim, um resultado baseado nas pessoas que estão envolvidas no seu desenvolvimento, por isso a qualidade dos resultados obtidos, dos procedimentos utilizados e dos conhecimentos empregados varia de acordo com a experiência dos profissionais atuantes no projeto, fazendo com que as chances de sucesso dependam das habilidades pessoais desses profissionais. Por isso é tão difícil prever os resultados obtidos no final do projeto, nessa fase.

- Nível 2 – gerenciado: nível em que as políticas e procedimentos para gerenciar o desenvolvimento de *software* estão definidos e são obedecidos, nos projetos em execução. Isso não significa que todos os projetos utilizem o mesmo processo, mas todos seguem o mesmo direcionamento e algum processo, de forma a terem um desenvolvimento mais organizado e com resultado mais previsível. Dessa forma, o planejamento do projeto é baseado em estimativas e na experiência anterior, obtida com outros projetos realizados. Os projetos utilizam processos definidos, usados, disseminados, documentados, medidos e monitorados com rotinas que possuem o foco na melhoria contínua. Nesse nível de maturidade, o foco está nos processos com práticas gerenciais, já nos projetos e não nas pessoas, o que gera mais confiabilidade nos resultados a serem atingidos no final do projeto. Esse nível de maturidade ainda não se preocupa com os processos de cunho técnico, relacionados com a engenharia de *software*, por isso é que não cobre todo o ciclo de desenvolvimento de *software*, sendo necessário, para isso, implantar o nível 3 de maturidade.
- Nível 3 – definido: nível em que os processos utilizados são estabelecidos e padronizados em toda a organização, ou seja, todos os projetos seguem o mesmo processo padrão definido para a empresa como um todo. As adaptações nos processos, necessárias por conta da especificidade de algum projeto em particular, são discutidas e documentadas no processo para o projeto, que passa a conter a parte padronizada do processo organizacional e as adaptações específicas para atender a alguma necessidade de negócio do projeto. Esse nível de maturidade considera os processos técnicos, acrescentados ao conjunto de processos

gerenciais que já eram considerados desde o nível 2 de maturidade. Dessa forma, tanto os processos gerenciais quanto os processos técnicos passam a ser usados em todos os projetos, e os processos são considerados como da empresa como um todo e não mais, apenas, dos projetos. É quando ocorre o que se chama de *institucionalização dos processos*, quando esses são definidos e utilizados por todos na empresa.

- Nível 4 – quantitativamente gerenciado: nível de maturidade em que são estabelecidas metas quantitativas para os processos e produtos, e medidas de qualidade e produtividade são coletadas em todos os projetos, com o objetivo de propor melhorias controladas com base em informações concretas. É estabelecido um controle estatístico de processos e a gestão passa a ser feita com bases quantitativas, ou seja, é a evolução da tomada de decisão, que deixa de ser empírica, calcada no *eu acho que essa melhoria vai trazer ganho*, para ser pautada em fatos, em conclusões como *essa melhoria é necessária porque os resultados de X projetos mostraram um comportamento não adequado às metas definidas*. Esse já é um alto nível de maturidade, que dispõe de um controle estatístico que utiliza o conhecimento para definir as melhorias que oferecerão maior ganho para o processo.
- Nível 5 – otimização: nível complementar ao nível 4, que coloca em ação as melhorias identificadas pelo controle estatístico realizado no nível anterior da pirâmide de maturidade. O ciclo de melhoria vira uma rotina para a empresa, e o olhar de todos está voltado para alcançar a excelência. A organização como um todo mostra-se engajada na melhoria contínua dos processos, por meio da identificação dos seus pontos fracos e defeitos e da definição de ações preventivas das causas de defeito observadas. Dessa forma, mudanças mais significativas nos processos ou nas tecnologias adotadas são feitas mediante análise de custo e benefício, com base em dados quantitativos. Além disso, toda mudança precisa ser apresentada e ensinada aos profissionais de desenvolvimento de *software*, testada nos projetos e avaliada quanto à sua eficácia. Se a mudança for eficaz para o atingimento da meta pretendida, ela passará a fazer parte do processo organizacional adotado por todos os projetos realizados a partir daquele momento. Ou seja, trata-se de um ciclo de

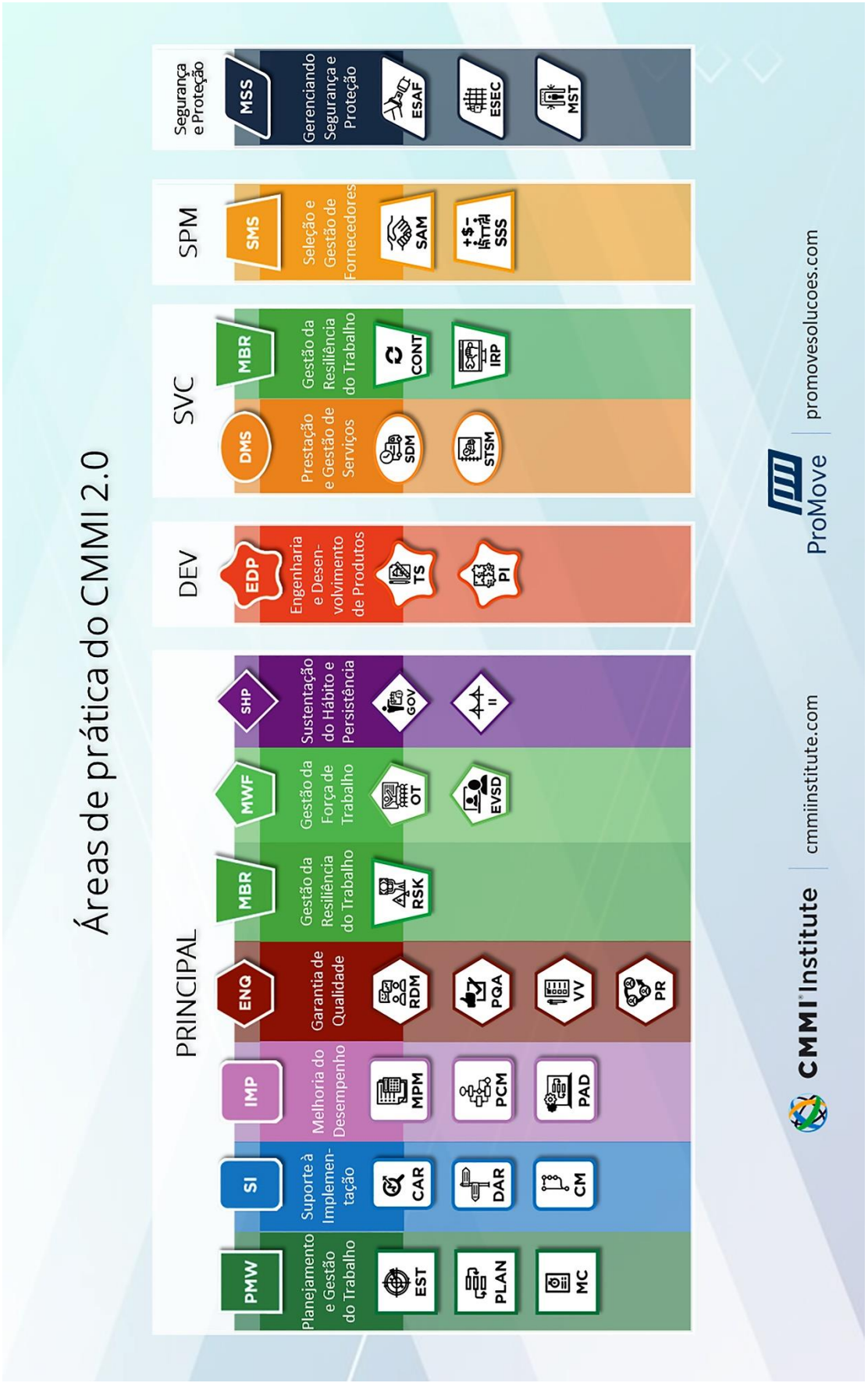
melhoria contínua, sempre em busca de resultados mais satisfatórios para os projetos.

Cada nível de maturidade é composto por áreas de processos e estas, por sua vez, por práticas que são implementadas como tarefas definidas para a execução do processo.

1.2 Área de prática ou área de processo do CMMI

A versão 2.0 do CMMI é formada por 29 práticas agrupadas em áreas de capacidade, dependendo da visão do modelo de que fazem parte. Como falamos anteriormente, o CMMI é composto por 3 modelos: o de desenvolvimento, o de serviços e o de aquisição. Logo, a organização é feita da seguinte forma: das 29 áreas de prática do CMMI, 18 delas fazem parte do grupo de áreas de capacidade principal, comuns às 3 visões do modelo CMMI. As demais áreas de capacidade são específicas a cada um dos modelos em particular, conforme representado na Figura 2 a seguir.

Figura 2 – Organização das áreas de prática do CMMI, v. 2.0



Fonte: https://promovesolucoes.com/cmmi-o-que-e-e-como-usar/

Dessa forma, detalhando a Figura 2, temos as áreas de práticas relacionadas a cada uma das áreas de capacidade, que são: planejamento e gestão do trabalho – PMW, suporte à implementação – SI, melhoria do desempenho – IMP, garantia de qualidade – ENQ, gestão da resiliência dos negócios – MBR, gestão da força de trabalho – MWF, sustentação do hábito e persistência – SHP, engenharia e desenvolvimento de produtos – EDP, prestação e gestão de serviços – DMS, seleção e gestão de fornecedores – SMS e gerenciando segurança e proteção – MSS.

Vamos entender cada uma destas áreas de capacidade, ordenadas em ordem alfabética, e suas respectivas áreas de prática.

a. **ENQ**

- Desenvolvimento e gestão de requisitos – RDM
- Garantia de qualidade do processo – PQA
- Verificação e validação – VV
- Revisão por pares – PR

b. **EDP**

- Solução técnica – TS
- Integração do produto – PI

c. **DMS**

- Gestão de prestação de serviço – SDM
- Gestão estratégica de serviço – STSM

d. **SMS**

- Seleção de fornecedor – SSS
- Gestão de contrato de fornecedor – SAM

e. **PMW**

- Estimativa – EST
- Planejamento – Plan
- Monitoramento e controle – MC

f. **MBR**

- Gestão de riscos e oportunidades – RSK
- Solução e prevenção de incidentes – IRP
- Continuidade – Cont

g. **MWF**

- Treinamento organizacional – OT

- Habilitando a entrega de soluções virtuais – EVSD

h. **SI**

- Análise causal e resolução – CAR
- Análise de decisão e resolução – DAR
- Gestão de configurações – CM

i. **SHP**

- Governança – GOV
- Infraestrutura de implementação – II

j. **IMP**

- Gestão de processos – PCM
- Desenvolvimento de ativos de processos – PAD
- Gestão de desempenho e medição – MPM

k. **MSS**

- Habilitando a proteção – Esaf
- Habilitando a segurança – Esec
- Gerenciando ameaças e vulnerabilidades de segurança – MST

É possível verificar que o CMMI é um modelo complexo, mas bastante completo, que orienta vários aspectos de um processo de desenvolvimento de *software*. Mas é um modelo que precisa ser bastante adaptado para ser utilizado por empresas menores e em projetos com complexidades mais baixas.

Por sua vez, a MPS.BR é o modelo brasileiro para melhoria de *software*, definido baseado no CMMI, porém com características que respeitam a realidade dos projetos realizados por empresas brasileiras. Dessa forma, vamos discutir a seguir as áreas de processo e as suas práticas, apresentando como são estruturadas no MPS.BR e fazendo uma comparação com o CMMI.

TEMA 2 – O QUE É A MPS.BR

Segundo o seu guia geral (Softex, 2016), o modelo MPS.BR foi criado em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (Softex), e conta com apoio de vários órgãos, tais como: o Ministério da Ciência, Tecnologia e Inovações (MCTI), a Financiadora de Estudos e Projetos (Finep), o Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (Sebrae) e o Banco Interamericano de Desenvolvimento (BID). O

modelo tem como objetivo a melhoria dos processos de *software* e dos serviços tecnológicos ofertados por empresas que desenvolvem *software* no Brasil.

A base teórica do modelo MPS.BR está associada à norma ISO 12207:2009 (no Brasil, atual ABNT NBR ISO/IEC/IEEE 12207:2021) e ao CMMI. A norma, subintitulada *Engenharia de sistemas e software – processos de ciclo de vida de software*, define uma estrutura e terminologias comuns a processos de ciclo de vida de *software*. Essa estrutura contempla processos, atividades e tarefas que serão realizados durante o fornecimento, desenvolvimento, operação, manutenção e descontinuidade dos produtos de *software*, assim como durante a aquisição de um produto ou serviço de *software* (ABNT, 2021).

A justificativa para a criação de um modelo brasileiro foi pautada nas mudanças que estão ocorrendo nos ambientes de negócios, que obrigam os processos a saírem da visão tradicional, baseada em áreas funcionais, em direção a processos centrados nas necessidades dos clientes. A competitividade depende, cada vez mais, de alcançar melhores resultados, por meio da busca pela qualidade. Ou seja, para as empresas de *software*, essa busca pela melhoria implica tanto a qualidade dos produtos de *software* e serviços correlatos, como a dos processos de produção e distribuição de *software*.

Se pararmos para analisar essa justificativa em mais detalhes, faz todo sentido definir e difundir um modelo de qualidade de processo de *software* nacional que se adeque à realidade das empresas brasileiras, que, no geral, são empresas com orçamentos mais restritos e que precisam de *softwares* menos complexos quando comparados aos *softwares* desenvolvidos para o Departamento de Defesa Americano ou para a Administração Nacional da Aeronáutica e Espaço dos EUA (Nasa). E, aqui, cabe relembrar que esse foi o aspecto motivador da criação do CMMI: organizar a contratação de fornecedores para o desenvolvimento desses tipos de *softwares*.

Fica claro, então, que o diferencial competitivo entre uma empresa que desenvolve *software* e outra é como seus processos estão organizados, dependendo então do nível de melhoramento do produto final que é criado e da facilitação das etapas que precisam ser seguidas para se garantir um caminho mais estruturado e com menos riscos, uma vez que desenvolver *software* é uma disciplina que precisa se preocupar e gerenciar várias influências externas, como alterações nas necessidades dos clientes, requisitos legais ou obrigatórios para o ramo de negócio da empresa do cliente, evolução da tecnologia; além de

influências internas, como premissas que foram utilizadas para planejar o projeto e que deixam de ser verdadeiras por imprevistos ao longo do processo de desenvolvimento do *software*. Estamos falando aqui, por exemplo, de aumentos inesperados de custos de recursos tecnológicos previstos para o projeto ou mesmo de profissionais cujo trabalho foi previsto e não fazem mais parte da equipe de desenvolvimento. Todos esses pontos e muitos outros que podem ocorrer ao longo do desenvolvimento de um projeto de *software* afetam o planejamento realizado para o projeto e precisam ser gerenciados para que não impactem o resultado final esperado para o projeto.

É por isso que é importante ter um modelo de desenvolvimento de *software* para se basear ao definir um processo que vai ser utilizado por todos os projetos de uma empresa. Um bom processo tende a gerar um bom produto de *software*. Por outro lado, se o processo não for o mais adequado às necessidades da empresa, ele prejudicará mais do que ajudará o desenvolvimento de *software*.

Mas é fundamental deixar claro que o CMMI e a MPS.BR são modelos que apoiam as empresas a definirem seus processos com base em paradigmas que precisam ser implementados e buscando garantir a qualidade no desenvolvimento de *software*; porém, em momento nenhum eles definem como essa implementação deve ser feita, pois como os processos serão implementados é algo de definição da empresa, que para fazê-lo se baseará nos seus recursos, conhecimentos e disponibilidades. Essa flexibilidade torna os modelos adequados a empresas de vários portes e áreas de atuação.

Vamos entender, agora, como os níveis de maturidade estão estruturados na MPS.BR.

2.1 Níveis de maturidade na MPS.BR

A MPS.BR pode ser entendida como um programa, uma vez que também é constituída por três diferentes modelos, sendo eles:

1. Modelo de Referência Associado à Melhoria de Processo de Software – **MR-MPS-SW**
2. Modelo de Referência Associado à Melhoria de Processo de Serviços – **MR-MPS-SV**

3. Modelo de Referência Associado à Melhoria de Processo de Gestão de Pessoas – **MR-MPS-RH**

O nosso foco será compreender como o MR-MPS-SW está estruturado e organizado, focando em como o modelo orienta a definição de processos para o ciclo de vida do desenvolvimento de um *software*. A MPS.BR é um modelo de referência de *software* baseado em níveis de maturidade, que são uma combinação entre processos e sua capacidade para atingir o resultado esperado. Assim, cada processo definido pelo modelo é caracterizado por seu propósito e seus resultados esperados – são as práticas que devem fazer parte do processo de desenvolvimento de *software*. O resultado esperado do processo é atendido por meio de atividades e tarefas. Cada nível do modelo é atendido por um conjunto de processos a serem implementados pela empresa que quer estar aderente ao modelo. Portanto, a capacidade do processo é a sua habilidade para alcançar os objetivos de negócio. Cada nível do modelo é atendido por um conjunto de atributos de processo associados.

O MR-MPS-SW está organizado em sete níveis de maturidade, apresentados a seguir do menor ao maior, pois são níveis incrementais, que são alcançados de forma contínua e mantendo a capacidade do nível imediatamente inferior. Os níveis de maturidade são identificados por letras, da seguinte forma:

- **G: parcialmente gerenciado** – tem como objetivo iniciar a organização do processo de desenvolvimento de *software*, contando já com parte dos processos necessários para se gerenciar um projeto de *software*.
- **F: gerenciado** – tem como objetivo complementar a organização inicial do processo de desenvolvimento de *software*, com uma segunda parte dos processos necessários para se gerenciar um projeto de *software*. A junção dos níveis F e G da MPS.BR corresponde ao nível 2 do CMMI.
- **E: parcialmente definido** – tem como objetivo definir os processos iniciais necessários para cobrir parte do ciclo de vida do desenvolvimento do *software*. Nesse nível de maturidade, o foco está em ter processos definidos para toda a organização e não apenas processos gerenciados, aplicados a alguns projetos.
- **D: largamente definido** – tem como objetivo definir mais alguns processos necessários para cobrir parte do ciclo de vida do desenvolvimento do *software*.

- **C: definido** – tem como objetivo definir os demais processos necessários para estabelecer de forma completa todo o ciclo de vida do desenvolvimento do *software*, isso envolvendo processos gerenciais e processos técnicos. A junção dos níveis E, D e C da MPS.BR corresponde ao nível 3 do CMMI.
- **B: gerenciado quantitativamente** – tem o mesmo objetivo e características do nível 4 do CMMI.
- **A: em otimização** – tem o mesmo objetivo e características do nível 5 do CMMI.

2.2 Processos dos níveis de maturidade da MPS.BR

Apesar de os modelos serem similares no seu objetivo, a forma como foram estruturados é bastante diferente. O CMMI é organizado em quatro níveis de maturidade, indo do nível 2 até o nível 5 (é importante ressaltar que, apesar de o modelo dizer que possui cinco níveis de maturidade, o nível 1 não abrange áreas de prática, portanto não define nenhum processo a ser seguido, o que o torna de efeito nulo), o que concentra mais processos em cada um dos níveis de maturidade. Por outro lado, a MPS.BR é organizada em sete níveis de maturidade, todos com processos definidos a serem seguidos. Dessa forma, a adequação de uma empresa ao modelo ocorre de maneira mais fluida e menos pesada para todos os profissionais. É uma implementação mais suave do que a que ocorre quando uma empresa implanta o CMMI, sem se perder a noção de completude dos processos, que, no final dos sete níveis, é tão aderente e mesmo equivalente aos níveis do CMMI.

Cada processo definido pela MPS.BR é composto por resultados esperados, que são as tarefas ou ações que devem ser definidas e executadas para se atingir o objetivo do processo. A MPS.BR ainda organiza o que é chamado de *atributos de processos*, que são atributos que devem ser cumpridos na implementação dos processos de cada nível de maturidade. Os atributos de processo reforçam se os processos implementados pelas empresas estão no nível adequado de maturidade, entregando a qualidade esperada para o nível em questão.

Vamos analisar os processos de cada um dos sete níveis de maturidade que compõem a MPS.BR:

1. **Nível G: parcialmente gerenciado** – nível composto por processos considerados gerenciais e básicos para a execução de qualquer projeto de *software*, sendo eles:

- **gerência de projetos (GPR)**: tem como propósito estabelecer e manter os planos que definem as atividades, recursos e responsabilidades do projeto, bem como prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho; esse processo é composto por 19 resultados esperados, que precisam ser implementados para se atingir seu objetivo, e é o maior processo do modelo MPS.BR.
- **gerência de requisitos (GRE)**: composto por cinco resultados esperados, tem como propósito gerenciar os requisitos do produto e dos componentes do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto e seu foco está em gerenciar o escopo definido para o projeto.

O nível G de maturidade deve atender aos seguintes atributos de processo:

- **AP 1.1**: nele ocorre a execução do processo, cujo objetivo é medir o quanto o propósito do processo é alcançado pela sua implementação e execução.
- **AP 2.1**: a execução do processo é gerenciada e seu objetivo é medir o quanto a execução do processo é gerenciada.

2. **Nível F: gerenciado** – nível composto por processos ainda considerados gerenciais, porém não tão básicos para a execução de qualquer projeto de *software*, nele já havendo, portanto, uma maior maturidade no gerenciamento das particularidades do projeto. Lembramos que, para atingir o nível F, é preciso atender a todos os processos do nível G e assim sucessivamente: quanto mais níveis de maturidade se quer atingir, a mais processos se deve atender. Seus processos são:

a. **aquisição (AQU)**: tem como propósito gerenciar a aquisição de produtos ou serviços necessários para a realização do projeto, se trata de uma área de processo não obrigatória, pois nem todo projeto depende de aquisição, e é composta por oito resultados esperados.

b. **gerência de configuração (GCO)**: tem como propósito estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos, pensando em controle de versão e controle de alteração, sendo uma área de processo que é nosso foco, composta por sete resultados esperados;
c. **garantia da qualidade (GQA)**: tem como propósito assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos, buscando a qualidade do processo e do produto de *software* a ser gerado como produto final do projeto, sendo composta por quatro resultados esperados;
d. **gerência de portfólio de projetos (GPP)**: tem como propósito iniciar e manter projetos que sejam necessários, suficientes e sustentáveis, de forma a atender aos objetivos estratégicos da organização e garantir a capacidade adequada para executá-los, sendo um processo composto por oito resultados esperados;
e. **medição (MED)**: tem como propósito coletar, armazenar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais em relação às metas de qualidade e de desenvolvimento definidas, sendo um processo composto por sete resultados esperados.

Nesse nível, a implementação dos processos também deve atender aos atributos dos níveis anteriores e ao seguinte atributo de processo:

- **AP 2.2**: os produtos de trabalho do processo são gerenciados e seu objetivo é medir o quanto os produtos de trabalho do processo são gerenciados, isto é, produzidos, controlados e mantidos.

3. **Nível E: parcialmente definido** – nível composto por processos chamados de *organizacionais*, voltados para a definição e a institucionalização dos processos que serão utilizados por todos os projetos da empresa. É nesse nível também que a empresa encontra processos que a orientam a documentar as adaptações necessárias para atender às especificidades de um determinado projeto, quando ocorrerem. Como em todos os níveis de maturidade, o nível E é composto pelos processos dos níveis anteriores (GPR, GRE, AQU, GCO, GQA,

GPP e MED), pelos resultados evoluídos da GPR e por mais quatro processos:

a. **avaliação e melhoria do processo organizacional (AMP)**: tem como propósito determinar o quanto os processos padrões da organização contribuem para se alcançar os seus objetivos de negócio e projetos e apoiar a organização a planejar, realizar e implantar melhorias contínuas nos processos, com base no entendimento de seus pontos fortes e fracos; nesse processo é que a melhoria é feita, ainda de forma avaliativa, sem o controle estatístico, que só vai ocorrer nos altos níveis de maturidade, como os níveis B e A, e esse processo contempla dez resultados esperados;
b. **definição do processo organizacional (DFP)**: tem como propósito estabelecer e manter um conjunto de processos organizacionais e padrões do ambiente de trabalho usáveis e aplicáveis às necessidades de negócio da organização e até mesmo a processos específicos ou partes de processos específicos a um determinado projeto, sendo esse processo composto por oito resultados esperados;
c. **gerência de recursos humanos (GRH)**: tem como propósito prover a organização e os projetos dos recursos humanos necessários e manter suas competências adequadas às necessidades do negócio, dado que as pessoas são um ativo de grande valor para os negócios, uma vez que os projetos são realizados, afinal, por elas, num processo composto por 11 resultados esperados e que envolve tanto a seleção como o desenvolvimento de profissionais, buscando a melhoria do trabalho executado por cada um deles nos projetos;
d. **gerência de reutilização (GRU)**: tem como propósito gerenciar o ciclo de vida dos ativos reutilizáveis, garantindo que componentes de qualidade possam ser reutilizados nos projetos, diminuindo assim o esforço de construção de novos *softwares*, num processo composto por cinco resultados esperados.

Nesse nível, a implementação dos processos também deve atender aos atributos de processo dos níveis anteriores e aos seguintes atributos de processo:

- **AP 3.1**: o processo é definido e com o objetivo de medir o quanto o padrão da organização é mantido de forma a apoiar sua adaptação para um projeto específico, quando necessário.
- **AP 3.2**: o processo está implementado e com o objetivo de medir o quanto o padrão está implementado na organização e sendo seguido pelos projetos realizados.

4. **Nível D: largamente definido** – nesse nível de maturidade entram os processos da engenharia de *software*, os processos relacionados às atividades técnicas de análise, projeto, construção e testes do *software*. Também é composto pelos processos dos níveis anteriores (GPR, GRE, AQU, GCO, GQA, GPP, MED, AMP, DFP, GRH, GRU) e por mais cinco processos, a saber:

a. **desenvolvimento de requisitos (DRE)**: tem como propósito definir os requisitos do cliente, do produto e dos seus e complementa o processo de GRE, modelando o entendimento sobre os requisitos a serem desenvolvidos no projeto, sendo composto por sete resultados esperados.
b. **integração do produto (ITP)**: tem como propósito unificar os componentes do produto, produzindo algo integrado e consistente com o projeto, e demonstrar que os requisitos funcionais e não funcionais são satisfeitos, pressupondo que o *software* é construído por partes e precisa ser integrado para que seja algo executável e sendo composto por nove resultados esperados;
c. **projeto e construção do produto (PCP)**: tem como propósito projetar, desenvolver e implementar soluções para atender aos requisitos e, composto por oito resultados esperados, é o processo que cobre as fases de análise e projeto técnico no ciclo de vida de desenvolvimento de *software*;
d. **validação (VAL)**: tem como propósito confirmar que um produto ou componente do produto atenderá ao seu uso pretendido quando colocado no ambiente para o qual foi desenvolvido e envolve a participação do cliente para validar o que foi construído, sendo composto por sete resultados esperados;
e. **verificação (VER)**: tem como propósito confirmar que cada produto de trabalho do projeto está sendo construído e atende apropriadamente aos

requisitos especificados para o *software*, sendo composto por seis resultados esperados.

- Nesse nível, a implementação dos processos deve atender aos atributos de processo dos níveis anteriores, não tendo nenhum nível de processo novo ou específico a ele.

5. **Nível C: definido** – esse nível de maturidade é composto pelos processos dos níveis de maturidade anteriores (GPR, GRE, AQU, GCO, GQA, GPP, MED, AMP, DFP, GRH, GRU, DRE, ITP, PCP, VAL, VER) e por mais três processos, com o objetivo de olhar de forma mais crítica para tópicos não fundamentais para o desenvolvimento do *software*, mas que agregam segurança e qualidade ao processo como um todo. Os processos acrescentados são:

a. **desenvolvimento para reutilização (DRU)**: tem como propósito identificar oportunidades de reutilização sistemática de componentes na organização e, se possível, estabelecer um programa de reutilização para desenvolver novos componentes, haja vista que a reutilização de componentes é um fator que aumenta bastante a produtividade das equipes, sendo composta por nove resultados esperados;

b. **gerência de decisões (GDE)**: tem como propósito analisar possíveis decisões críticas usando um processo formal, com critérios estabelecidos, para avaliação das alternativas identificadas e o foco de analisar as possíveis alternativas para um problema e tomar a melhor decisão a respeito, de modo embasado em informações sobre o cenário e as necessidades em questão, sendo composta por sete resultados esperados;

c. **gerência de riscos (GRI)**: tem como propósito identificar, analisar, tratar, monitorar e reduzir continuamente os riscos em nível organizacional e de projeto, dado que todo projeto de *software* implica riscos, mas, quando esses riscos são conhecidos e monitorados, tornam-se mais fáceis de serem gerenciados (esse processo é composto por nove resultados esperados).

- Nesse nível, a implementação dos processos deve atender aos atributos de processo dos níveis anteriores, não tendo nenhum nível de processo novo ou específico a ele.

6. **Nível B: gerenciado quantitativamente** – esse nível de maturidade é composto pelos processos dos níveis anteriores (GPR, GRE, AQU, GCO, GQA, GPP, MED, AMP, DFP, GRH, GRU, DRE, ITP, PCP, VAL, VER) e pelos resultados evoluídos do processo de GPR (sete resultados esperados). É nesse nível que os conceitos de controle estatístico são implementados nas empresas, com o objetivo de realizar melhorias embasadas em resultados obtidos nos projetos executados, buscando mais assertividade nas mudanças realizadas nos processos.

Nesse nível, a implementação dos processos também deve atender aos atributos de processo dos níveis anteriores e aos seguintes atributos de processo:

- **AP 4.1**: o processo é objeto de análise quantitativa cujo objetivo é medir o quanto as necessidades de informação são definidas, os relacionamentos entre os elementos de processo são identificados e dados são coletados, de maneira organizada e contínua em todos os projetos da empresa.
- **AP 4.2**: o processo é controlado quantitativamente e seu objetivo é medir os resultados dos projetos em relação às metas definidas, para se poder tomar ações caso o resultado não esteja de acordo com o esperado.

7. **Nível A: em otimização** – esse nível de maturidade é composto pelos processos dos níveis de maturidade anteriores (do G ao B), não apresentando processos específicos, mas sim atributos de processos que mostram a evolução da maturidade condizente ao nível mais alto do modelo.

Nesse nível, a implementação dos processos deve atender aos atributos de processo dos níveis anteriores e aos seguintes atributos de processo:

- **AP 5.1**: o processo é objeto de identificação de melhorias incrementais e inovações com o objetivo de medir o quanto as mudanças são observadas, com base na análise sistemática da causa raiz dos problemas, para a definição e implantação dos novos processos ou, até mesmo, adequações dos processos já existentes.
- **AP 5.2**: o processo é objeto de implementação de melhorias inovadoras e incrementais com o objetivo de medir o quanto as mudanças na

definição, gerência e desempenho alcançaram os objetivos previstos, criando um ciclo virtuoso de melhoria contínua.

Após estudar a organização e o funcionamento dos modelos CMMI e MPS.BR, é possível compreender a importância deles para o sucesso das empresas que desenvolvem *software*, de forma a definir um processo mais robusto e confiável, que direcione para as melhoras práticas de execução de um projeto de *software*.

Vamos, agora, explorar alguns outros modelos que apoiam também a construção organizada de um *software*.

TEMA 3 – CONHECENDO MAIS MODELOS: A SPICE E O RUP

O aumento contínuo na quantidade de *softwares* necessários para gerir os negócios e o incremento da complexidade desses *softwares* tornam a necessidade por padrões, modelos e metodologias cada vez maior. Essa é uma preocupação para todas as empresas que desenvolvem *software*, seja de forma profissional para outras empresas, seja para uso próprio.

Dessa forma, além do CMMI e da MPS.BR, existem outros padrões e modelos que direcionam a construção de *softwares*, sempre buscando oferecer boas práticas e um pouco mais de previsibilidade ao processo de construção de *software*. Vamos discutir mais sobre dois desses modelos, o RUP e a Spice.

3.1 Conhecendo o RUP

Dessa forma, além do CMMI e da MPS.BR, existem outros padrões e modelos que direcionam a construção de *software*, sempre buscando conferir boas práticas e um pouco mais de previsibilidade ao processo de construção de *software*. O RUP é uma metodologia completa, criada pela empresa Rational para viabilizar que grandes projetos de *software* sejam bem-sucedidos. A Rational foi comprada pela IBM em 2003 e, dessa forma, o RUP passou a ser mantido por um grupo de engenheiros de *software* da IBM.

Já discutimos que um dos grandes problemas nos projetos de *software* é o grande dinamismo e complexidade dos negócios nos dias de hoje, além da evolução rápida da tecnologia. Cada vez mais os sistemas são complexos e precisam estar prontos em menos tempo para atender às demandas do mercado. Além disso, as necessidades de negócio mudam rapidamente e a

especificação de um sistema de *software* provavelmente será alterada durante seu desenvolvimento, seja por questões legais, seja pelas demandas dos clientes e do mercado. Dessa forma e para minimizar todos esses pontos, o RUP está organizado em iterações, que são partes menores do *software*, construídas de cada vez, para diminuir o impacto das mudanças no projeto como um todo.

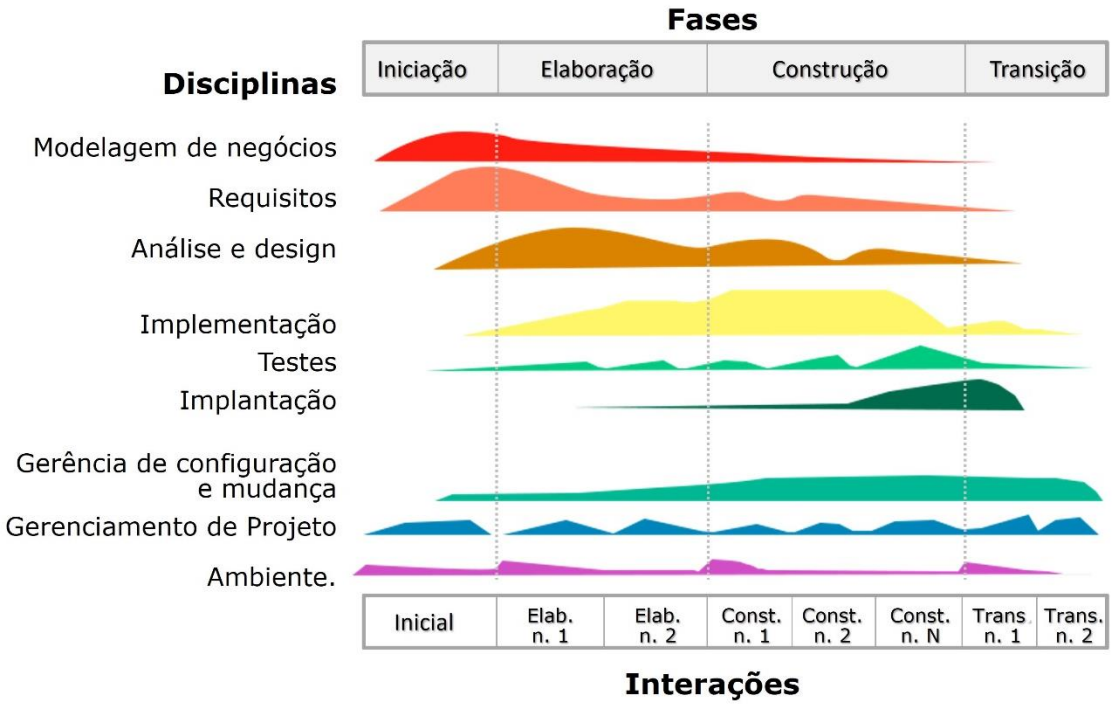
Ou seja, o escopo do *software* é dividido em partes que são planejadas, construídas, testadas e disponibilizadas para o usuário de maneira contínua, até que todo o escopo esteja pronto para ser utilizado pelo usuário final. Dessa forma, o principal objetivo do RUP é atender às necessidades dos usuários com qualidade e cumprindo o cronograma e o orçamento planejados. O RUP é um processo de engenharia de *software* que organiza o desenvolvimento em quatro fases. Em cada fase são tratadas questões sobre planejamento, levantamento de requisitos, análise, implementação, teste e implantação do *software*, em maior ou menor grau, conforme a fase que está sendo executada. Essas fases são:

- **Fase de concepção ou iniciação**: é a fase que abrange as tarefas de comunicação com o cliente, o alinhamento de expectativas e o planejamento inicial. É feito o plano de projeto avaliando-se os possíveis riscos e as estimativas de custos e prazos, estabelecendo-se as prioridades e fazendo o levantamento dos requisitos do sistema, de acordo com o escopo a ser atendido. Assim, haverá um acordo entre as partes interessadas na definição do escopo do projeto.
- **Fase de elaboração**: é a fase responsável pelo planejamento técnico do *software*, modelando a solução com base no escopo definido. O objetivo dessa fase é analisar de forma mais detalhada o problema a ser resolvido por meio do *software* que será construído. Os riscos são revisados e a arquitetura do projeto, modelada e definida. Os planos são revistos, verificando se o planejamento inicial está de acordo com o projeto técnico. Se for necessário, o planejamento é modificado e validado novamente com o cliente.
- **Fase de construção**: é a fase em que se desenvolvem ou adquirem os componentes de *software*. A decisão de desenvolver ou comprar os componentes do *software* deve ser embasada na capacidade da equipe de desenvolvimento e no custo envolvido. O principal objetivo dessa fase é a construção do *software* em si, baseada no projeto técnico modelado

na fase anterior. É na fase de construção que a maior parte da codificação ocorre, envolvendo também os testes para garantir a qualidade e a adequação do que está sendo construído.

- **Fase de transição**: abrange a entrega do *software* ao usuário e a fase de testes no ambiente de homologação, que deve simular, o mais próximo possível, o ambiente de produção. As atividades dessa fase incluem também o treinamento dos usuários finais no *software* e a realização de testes da versão beta do sistema visando garantir que ele possua o nível adequado de qualidade.

Figura 3 – Organização do RUP



Fonte: https://www.researchgate.net/figure/Figura-1-Arquitetura-Geral-do-RUP-A-Figura-1-mostra-como-a-enfase-dedicada-a-cada-fase_fig1_267337467

O RUP é um modelo de engenharia de *software* bastante completo, em que as disciplinas especificadas do lado esquerdo do desenho ocorrem em maior ou menor intensidade, dependendo da fase do projeto. E, em cada fase, as entregas dos projetos são organizadas em iterações, criando-se o *software* de maneira contínua, porém em pequenas partes, mais fáceis de gerenciar e de ir compreendendo e atendendo às expectativas do usuário.

As disciplinas previstas pelo RUP são:

- **Modelagem de negócio** – é quando o processo de negócio do cliente é entendido e modelado, para ser avaliado. Nesse momento, é possível propor melhorias ao processo do cliente, de forma a deixar o *software* mais funcional e adequado às necessidades do negócio.
- **Requisitos** – disciplina em que os requisitos ou funcionalidades que o *software* precisa executar são definidos com base na modelagem do processo de negócio e em que se trabalha a modelagem lógica do *software*.
- **Análise e *design*** – disciplina cujos requisitos levantados e compreendidos são projetados para virarem uma solução de *software*, trabalhando-se a modelagem física do *software*, definindo-se a melhor arquitetura, linguagem, *framework* e banco de dados a serem utilizados, além de se modelar o que é necessário para compreender melhor o funcionamento que o *software* deverá ter.
- **Implementação** – essa é a disciplina da construção do *software*, propriamente dita. A equipe de desenvolvimento implementa o modelo físico e lógico projetado nas disciplinas anteriores.
- **Testes** – disciplina para avaliar a qualidade e a coerência do que foi construído, executando testes que vão verificar os diversos cenários possíveis, de forma a melhorar a qualidade e minimizar defeitos.
- **Implantação** – essa disciplina disponibiliza o *software* projetado, construído e testado para ser utilizado no seu ambiente final de uso, ou seja, no ambiente de produção. A partir desse momento o *software* passa a ser intensamente usado por todos os seus usuários.
- **Gerência de configuração e mudança** – essa é uma disciplina de apoio e tem por objetivo controlar e monitorar os itens de configuração definidos para o *software*. O conceito é o mesmo dos já discutidos e aplicados pelo CMMI e pela MPS.BR.
- **Ambiente** – também é uma disciplina de apoio, que tem como objetivo disponibilizar o ambiente adequado à construção, teste e utilização do *software*. Além disso, é nessa disciplina que são feitos os monitoramentos constantes para se garantir que o ambiente continua sendo o adequado e possui a capacidade de suportar o *software* em seu pleno funcionamento. Manutenções devem ser feitas caso o ambiente não permita o funcionamento estável do *software*.

Analisando a Figura 3, vemos que as fases do RUP podem se dividir em mais de uma, cada, ao longo do ciclo de vida. Isso ocorre porque o projeto, seguindo o RUP, tem sua construção dividida em iterações nas quais os requisitos são construídos por partes, até que o escopo esteja completamente pronto. Essa divisão do escopo em partes menores ajuda no gerenciamento de possíveis mudanças de escopo e na validação do cliente, que passa a ter contato com o *software* em menos tempo, pois ele vai acompanhado a construção a cada iteração. E as disciplinas ocorrem em maior ou menor intensidade exatamente por conta das iterações, que podem necessitar de entendimento dos negócios, entendimento dos requisitos, da análise e do projeto e assim por diante, sempre detalhando melhor a parte do escopo que será trabalhada em cada iteração.

O RUP é um modelo tradicional de desenvolvimento de *software*, que agrega conceitos interessantes de iterações e desenvolvimento com a preocupação de entregar o *software* funcionando para o cliente o mais rápido possível.

3.2 Conhecendo a Spice

A Spice foi um projeto desenvolvido em 1993 como parte da definição de uma norma que organizasse um ciclo de vida para o desenvolvimento de *software*. Essa norma é chamada atualmente de *ISO/IEC 15504* – no Brasil, com a parte da Spice constando da ABNT NBR ISO/IEC 15504-4:2008 –, uma norma de padrão internacional que estabelece um *framework* para a construção de processos de avaliação e melhoria do processo de *software*. Um *framework* é um conjunto de processos definidos, incluindo ferramentas e padrões a serem seguidos, com o objetivo de organizar e dar mais previsibilidade ao resultado esperado para um *software* em construção (ABNT, 2008).

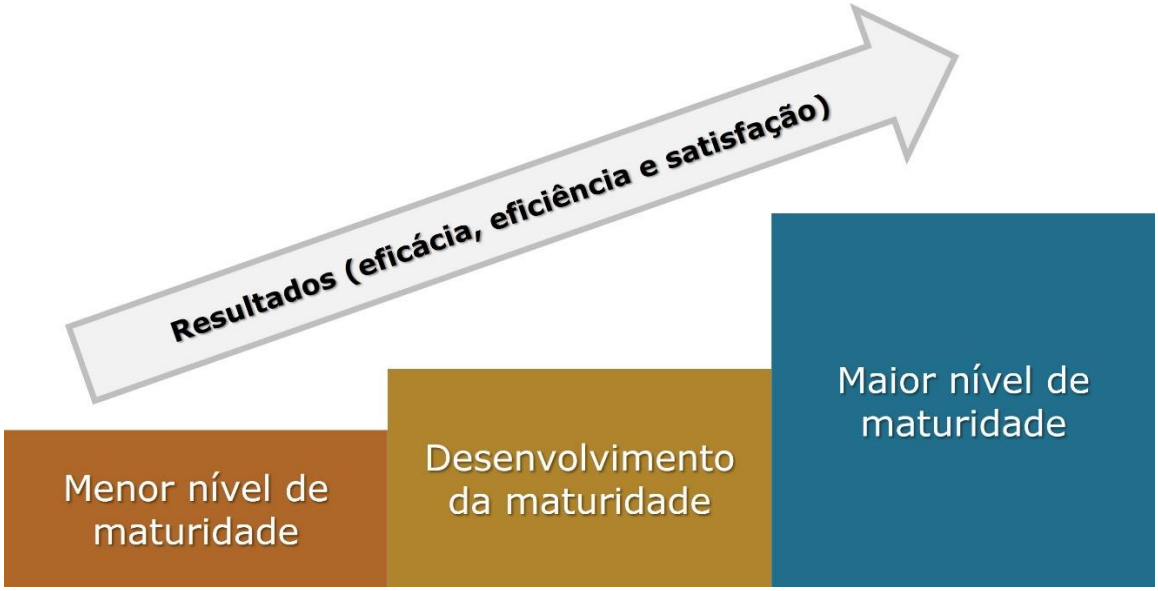
A norma ISO 15504 é originalmente dividida em cinco partes, da seguinte forma:

- **Parte 1**: define conceitos e o vocabulário utilizado ao longo da norma.
- **Parte 2**: define os requisitos mínimos para a realização de uma avaliação de aderência à norma, buscando consistência e repetibilidade.
- **Parte 3**: é um guia que apoia a interpretação dos requisitos para a realização de uma avaliação de aderência à norma.

- **Parte 4**: é um guia que apoia a determinação da capacidade do processo e das melhorias necessárias para buscar a excelência do processo.
- **Parte 5**: é um modelo de referência de processos, que explica melhor o funcionamento da norma, ajudando a interpretá-la como um todo.

Como em todo modelo que busca aumentar o nível de maturidade de um processo, o foco pode ser ilustrado pela Figura 4.

Figura 4 – Aumento da maturidade dos processos



fonte: https://advisera.com/27001academy/pt-br/blog/2015/04/14/atingindo-a-melhoria-continua-atraves-do-uso-de-modelos-de-maturidade/

Dessa forma, a ISO 15504 atua identificando o nível de maturidade de um processo e apoiando o desenvolvimento dessa maturidade até se alcançar um grau aceitável de eficácia, eficiência e satisfação, quando o processo está em um nível maior de maturidade. Se fizermos uma analogia, podemos lembrar de quando aprendemos a dirigir um carro. Não era fácil pensar em todos os comandos que tínhamos que executar para fazer o carro se movimentar. Porém, com o tempo e a prática em dirigir, os movimentos passam a ser até instintivos, quase nem pensamos mais no que estamos fazendo. Pois bem, isso é a evolução da maturidade em um determinado processo – nesse caso, no processo de dirigir um carro.

A ISO 15504 está organizada, então, em níveis de maturidade que ajudam a avaliar a maturidade dos processos, por meio da sua aderência aos requisitos de cada nível, no mesmo processo que vimos no CMMI e na MPS.BR. Inclusive,

tanto o CMMI quanto a MPS.BR utilizam a norma ISO 15504 como base para definir seus modelos (ABNT, 2008). Vamos entender, então, os níveis de maturidade propostos pela Spice:

- **Nível 0: incompleto** – nenhum processo implementado ou pouca ou nenhuma evidência de atingimento sistemático do propósito do processo. Aqui se trabalha com o esforço pessoal dos profissionais, não se tendo nenhum apoio processual para o desenvolvimento do *software*.
- **Nível 1: executado** – o processo atinge seu resultado esperado, ou seja, existe um processo ainda muito dependente de pessoas, pois não é um processo institucional, mas já há uma evolução, pois existe algum processo a ser seguido pelos projetos.
- **Nível 2: gerenciado** – o processo é implementado de uma forma gerenciada, existindo um planejamento e um monitoramento, o que pode gerar ajustes para atender aos resultados esperados para o processo. Os pacotes do produto de *software* são apropriadamente estabelecidos, controlados e mantidos. Aqui se vê um pouco mais de controle do que está sendo feito no desenvolvimento do *software*.
- **Nível 3: estabelecido** – o processo é implementado com algo já definido, padronizado e conhecido pelos profissionais que vão construir o projeto. É o nível em que os processos são capazes de atingir os resultados esperados, não dependendo apenas da senioridade dos profissionais, mas se direcionando o que esses profissionais devem fazer ao longo do projeto.
- **Nível 4: previsível**: é o nível de maturidade que permite que o processo opere dentro de limites definidos para atingir seus resultados esperados. Aqui entra o conceito do controle estatístico de processos, já discutido quando falamos sobre a MPS.BR.
- **Nível 5: otimizado** – é o nível de maturidade da melhoria contínua, quando os processos são continuamente melhorados para atingir objetivos relevantes atuais e futuros, para as empresas.

O foco dos modelos é, portanto, alavancar, por meio de boas práticas, processos que ajudem a organizar o desenvolvimento de *software*, buscando entregar *softwares* melhores para os clientes.

TEMA 4 – PLANEJAMENTO DAS MUDANÇAS E DA CONFIGURAÇÃO E CONTROLE DA CONSTRUÇÃO, DAS MODIFICAÇÕES E DO VERSIONAMENTO DO PROJETO DE *SOFTWARE*

Já estudamos que tanto o CMMI quanto a MPS.BR possuem processos que cobrem as necessidades básicas para definir de forma completa a gerência de configuração, no contexto do desenvolvimento de *software*. Vamos nos aprofundar, então, em como a MPR.Br define a gerência de configuração e quais os resultados esperados para cobrir as necessidades desse processo.

De acordo com a MPS.BR (Softex, 2016), o propósito do processo de gerência de configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos. Além disso, o modelo reforça que a gerência de configuração é a disciplina responsável por controlar a evolução de sistemas de *software*, mesmo após este entrar em produção.

A gerência de configuração se inicia com a definição dos produtos de trabalho que deverão estar sob controle de versão. Esses produtos de trabalho selecionados passam então a ser chamados de *itens de configuração*. Um ponto importante a se entender é que o processo de gerência de configuração está contemplado em todos os demais processos, uma vez que produtos de trabalho gerados dos demais processos devem estar sob controle de versão. Os níveis de controle podem variar de acordo com a importância ou criticidade dos produtos de trabalho, mas devem ser adequados a cada caso específico.

A MPS.BR (Softex, 2016) ainda define sistemas que fazem parte do processo de gerência de configuração, tais como:

- O sistema de gerenciamento de construção do *software* automatiza o processo de transformação dos diversos componentes que compõem um projeto em um sistema executável, que será utilizado pelos usuários, organizando o gerenciamento de liberação e entrega das partes de *software*, que são construídas de maneira incremental, para as fases seguintes do ciclo de vida do desenvolvimento de *software*, até elas estarem completamente prontas.
- O sistema de controle de versões permite que os itens de configuração sejam identificados e evoluam de forma controlada. Essa característica é

necessária para que as diversas solicitações de modificação que possam existir para os requisitos do *software* sejam assim tratadas em paralelo, sem gerar riscos de perdas de código ou inserção de defeitos no *software*.

- O sistema de controle de modificações tem a função de executar sistematicamente o controle da configuração, armazenando todas as informações geradas durante o andamento das solicitações de modificação e relatando essas informações aos envolvidos, assim como estabelecido pela função de contabilização da situação da configuração. Esse sistema é também responsável pelo controle do histórico de tudo o que ocorre com cada um dos itens de configuração do *software*.

Vamos, agora, conhecer os resultados esperados definidos pela MPS.BR para se implementar o processo de gerência de configuração, relembrando que o modelo não define como fazê-lo, apenas o que deve ser implementado (Softex, 2016).

4.1 Resultados esperados para a gerência de configuração

Relembrando, um resultado esperado é uma ação que precisa ser executada, para que parte do processo seja cumprida. O cumprimento do conjunto de resultados esperados definidos faz com que o processo inteiro seja atendido.

Conforme a MPS.BR (Softex, 2016), o processo de gerência de configuração possui sete resultados esperados, sendo eles:

1. **GCO1** – é o resultado esperado que estabelece e mantém um sistema de gerência de configuração e que deve funcionar de maneira sistemática ao longo do desenvolvimento dos projetos, englobando o controle de versões, o controle de modificações e o gerenciamento de construção do produto. É fundamental se pensar em um sistema automatizado, que confira produtividade e facilite a vida da equipe de desenvolvimento, além de definir procedimentos de cópia e recuperação dos dados (o que, em inglês, é chamado de *backup* e *restore*, respectivamente).
2. **GCO2** – é o resultado esperado que identifica os itens de configuração com base em critérios estabelecidos de acordo com as necessidades dos projetos e da empresa. Para cada item de configuração identificado,

devem ser definidos: um identificador único; o nível de controle pretendido e o momento de se aplicar esse controle; um responsável.

3. **GCO3** – é o resultado esperado que executa a gerência de configuração em si, colocando sob *baseline* (linha de base) os itens de configuração que terão um controle formal. Durante a execução dos projetos, o sistema de gerência de configuração usualmente atua em um baixo nível de controle, permitindo maior produtividade da equipe de desenvolvimento. Isso porque esse é o momento em que a construção do *software* está ocorrendo e os componentes estão sendo constantemente alterados. Mas, quando esses itens de configuração passam a servir como insumo para as demais fases do ciclo de vida do desenvolvimento de *software*, o nível de controle pode ter que ser aumentado, evitando que modificações sejam feitas sem a devida aprovação e notificação aos interessados, minimizando o retrabalho e evitando até mesmo a perda de informações. Uma linha de base, ou *baseline*, é um conjunto de itens de configuração que estão prontos para evoluir para a próxima etapa do ciclo de vida, por isso mesmo precisam ser controlados de maneira unificada, para garantir que se produza uma versão do *software* coerente.
4. **GCO4** – é o resultado que cuida da situação dos itens de configuração e das *baselines* registradas e disponibilizadas ao longo do tempo, ou seja, do histórico de cada item de configuração. As ações de gerenciamento de configuração, como a inclusão e a alteração de itens no repositório de componentes, e a geração e a liberação de *baselines* precisam ser registradas e disponibilizadas em um nível de detalhe suficiente para que o conteúdo e a situação de cada item de configuração sejam conhecidos e que versões anteriores possam ser recuperadas, caso seja necessário.
5. **GCO5** – é o resultado responsável por controlar as modificações nos itens de configuração. A partir do momento em que os itens de configuração passam a fazer parte de uma *baseline*, toda e qualquer modificação desses itens de configuração deve passar por um processo formal de controle de modificações, que visa analisar o impacto das modificações e notificá-lo aos afetados, evitando retrabalho e efeitos colaterais indesejáveis.
6. **GCO6** – é o resultado responsável por controlar o armazenamento, o manuseio e a liberação de itens de configuração e *baselines*. Todos os

produtos de trabalho que forem itens de configuração, tanto de projetos quanto de processos, são armazenados no sistema de gerência de configuração seguindo as especificações definidas para cada tipo de item de configuração. Apesar de a MPS.BR não exigir o emprego de uma ferramenta específica, um *software* de controle de versão e de alterações facilita o atingimento desse resultado acompanhando cada item de configuração.

7. **GCO7** – esse resultado prevê que auditorias de itens de configuração sejam realizadas objetivamente, para assegurar que as *baselines* e os itens de configuração estejam íntegros, completos e consistentes. O objetivo de realizar uma auditoria de itens de configuração é garantir que a *baseline* contenha todos os componentes necessários e corretos para se compor uma versão do *software* que evoluirá para a próxima fase do ciclo de vida de desenvolvimento de *software*.

Portanto, fazendo um resumo do processo de gerência de configuração proposto pela MPS.BR, temos que o primeiro resultado esperado direciona para se definir como será realizada a gerência de configuração nos projetos. Em seguida, são identificados os produtos de trabalho que serão tratados como itens de configuração. Depois, o processo definido para os itens de configuração é executado. A execução se preocupa com registrar e controlar as modificações dos itens de configuração ao longo do tempo, além de manter o armazenamento e o manuseio desses itens. E, por último, a MPS.BR direciona para a realização de uma auditoria que vai garantir que todo o processo definido seja executado de maneira adequada.

Após tudo o que discutimos até o momento, acreditamos que tenha ficado clara a importância da gerência de configuração para se estruturar e manter um *software* padronizado, organizado e com critérios de qualidade. Tudo isso com o objetivo de facilitar a vida da equipe de desenvolvimento, garantindo a construção e a manutenção do *software* de forma mais organizada, produtiva e facilitada, buscando a excelência na construção de *software*.

Vamos, agora, compreender o que é o comitê de mudança e como ele funciona no contexto da gerência de configuração.

TEMA 5 – PAPÉIS E RESPONSABILIDADES DO COMITÊ DE MUDANÇA

O conceito de comitê de mudança está muito relacionado com a necessidade de controlar de maneira mais efetiva os componentes construídos e mantidos por um projeto de *software*. Estudamos, anteriormente, a importância de se definir e seguir um processo de gerência de configuração com o objetivo de garantir o controle, evitar o retrabalho e monitorar as alterações promovidas em um *software* ao longo do tempo, de forma a não gerar defeitos indesejados ou perda de código. Para isso é que, com o intuito de conferir formalidade e análise crítica ao momento de alterar um *software* já pronto e em funcionamento, muitas empresas usam o comitê de mudança, que é formado por um conjunto de pessoas que possuem a responsabilidade de avaliar as mudanças e seus impactos e autorizar ou não a sua implantação no *software*. O comitê, inclusive, é responsável por analisar o melhor momento para se implementar uma mudança no *software*, quando o seu impacto será o menor possível, como nos momentos de uso menos intenso do *software*, por exemplo.

A constituição do comitê de mudança se faz necessária quando um *software* já está em produção, pois, nesses casos, há o risco de o negócio ser impactado por uma manutenção mal planejada, que pode gerar prejuízos enormes para as empresas. Geralmente, o comitê de mudança é formado por profissionais de mais de uma área envolvida com o *software*, por exemplo: área de negócios, área de desenvolvimento, área de riscos, área de segurança da informação, área de suporte, entre outras, cada uma dessas áreas analisando, conforme seu escopo de atuação, a mudança e seu impacto na estabilidade dos negócios. O objetivo do comitê de mudança não é impedir a evolução do *software*, mas sim que a isso se proceda com tranquilidade e estabilidade, por conta das necessidades do mercado, da legislação e dos negócios.

FINALIZANDO

Nesta etapa, discutimos vários processos e normas que apoiam a implantação de um processo de gerência de configuração robusto, que suporte o desenvolvimento de um *software*. É essencial se construir um *software* pensando em qualidade e, para isso, a gerência de configuração é um grande aliado, pois organiza todos os produtos de trabalho controlando as versões que são geradas ao longo do desenvolvimento do *software*, as mudanças ou

alterações e também o trabalho de quem as fez, em cada uma das versões dos produtos de trabalho gerados durante o desenvolvimento do *software*.

Precisamos reforçar a importância de se ter uma ferramenta que automatize o processo de gerência de configuração, pois isso evita trabalho manual e erros humanos, além de facilitar a vida da equipe de desenvolvimento de *software*. O processo definido deve ser útil e fácil de ser seguido, para que a cultura do processo seja rapidamente absorvida e seguida pela equipe de trabalho. Todo processo a ser implantado em uma empresa passa pela necessidade de se criar uma nova cultura, afinal as pessoas vão passar a trabalhar seguindo o direcionamento de um processo, o que, muitas vezes, pode não ser aceito imediatamente. É preciso, então, criar uma cultura de boas práticas, mostrando o ganho que será alcançado com o uso do processo.

Conhecemos ainda, nesta etapa, a ISO 15504, que é uma norma que padroniza o conceito de nível de maturidade que vai sendo alcançado conforme os processos vão sendo utilizados, avaliados e melhorados (ABNT, 2008). Vimos também os modelos de melhoria de processo de *software* CMMI e MPS.BR e como eles criam uma estrutura de processo que auxilia na organização de um processo próprio para a empresa que busca criar *softwares* melhores e de maneira previsível e organizada. E, por último, conhecemos o RUP, que é uma metodologia para desenvolvimento de *software* baseada em fases e disciplinas, que são divididas em iterações com o intuito de gerenciar melhor um escopo que normalmente é grande e complexo, quando falamos do contexto de desenvolvimento de *software*.

Vamos continuar evoluindo e nos aprofundando nos conceitos que envolvem a gerência de configuração.

REFERÊNCIAS

ABNT – Associação Brasileira de Normas Técnicas. **ABNT NBR ISO/IEC 15504-4:2008**: tecnologia da informação – avaliação de processo – parte 4: orientação no uso para melhoria do processo e determinação da potencialidade do processo. Rio de Janeiro, 25 fev. 2008.

_____. **ABNTNBR ISO/IEC/IEEE 12207:2021**: engenharia de sistemas e software – processos de ciclo de vida de software. Rio de Janeiro, 24 ago. 2021.

SOFTEX – Associação para Promoção da Excelência do Software Brasileiro. Melhoria de Processo do Software Brasileiro. **MPS.BR**: guia geral MPS de software. Brasília; Manaus, jan. 2016. Disponível em: <https://www.softex.br/wp-content/uploads/2018/11/MPS.BR_Guia_Geral_Software_2016-com-ISBN.pdf>. Acesso em: 12 maio 2022.