# GERÊNCIA DE CONFIGURAÇÃO E EVOLUÇÃO

AULA 3

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, mantendo controle de versão e de alteração para evitar retrabalho ou perdas de dados. Esse controle é fundamental, principalmente quando o mercado está em crescimento e as demandas por novidades são grandes. Nesses casos, o software precisa ser evoluído para acompanhar as tendências de mercado.

Além disso, não podemos esquecer que trabalhamos em equipes, nas quais o desenvolvimento do software ocorre de forma colaborativa e o código é propriedade coletiva da equipe de desenvolvimento. Quanto mais integrado, padronizado e controlado estiver o código, melhor e mais produtivo será o desenvolvimento.

Nesta etapa, vamos explorar mais em detalhes como ocorrem as mudanças nos softwares, os principais tipos de mudanças que ocorrem e como fazer uma gestão efetiva da evolução dos softwares.

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

- Mudanças de software;
- Requisição e gerenciamento de mudanças;
- Tipos de mudança;
- Impacto das mudanças;
- Gestão de mudanças.

Vamos explorar os tópicos e entender como o processo de gerência de configuração apoia para controlar a gestão de mudança nos projetos de software.

## TEMA 1 – MUDANÇAS DE SOFWARE

Já percebemos que construir um software é um trabalho que se encerra quando entregamos o software para o usuário começar a utilizá-lo no seu dia a dia, não é mesmo? A vida do software continua em constante evolução enquanto ele estiver sendo utilizado. Dessa forma, as mudanças são inerentes a todo software, seja por necessidades de correção, por necessidades de adaptação

ou mesmo por necessidades de evolução do software, por conta da concorrência e tendências de mercado.

Dessa forma, uma coisa podemos tratar como certa, o software vai evoluir e para isso precisará passar por mudanças. Mas o risco de uma mudança precisa ser gerenciado, para que não gere impactos ou até mesmo prejuízo para o negócio de uma empresa.

Você consegue imaginar o tamanho do prejuízo para um site como o da americanas.com ou para a amazon.com se o sistema estiver inoperante por meia hora que seja? Pois é, alguns milhões de reais. E ninguém quer isso, não é mesmo?

A necessidade de gerenciar bem as mudanças fica cada vez mais evidente. Dessa forma, vamos compreender um pouco mais sobre o que é uma mudança em software. Para isso, vamos voltar em conteúdo anterior, relembrando os tipos de mudança que podem ocorrer ao longo do ciclo de vida de um software.

O escopo de uma mudança pode envolver necessidade de alteração por evolução nos requisitos, também chamada de mudanças evolutivas. Ou por envolver correções do funcionamento de algum requisito, também chamada de mudanças corretiva.

Vamos detalhar cada um desses tipos de mudança a seguir.

## 1.1 Mudanças corretivas

Como discutimos anteriormente, um dos tipos mais comuns de mudança é a mudança corretiva, que envolve a correção de defeitos ou de mal entendimento dos requisitos. Parece improvável que mesmo após um tempo do software em produção ainda sejam identificados defeitos. Mas não é incomum que isso ocorra, pois pode acontecer de algum cenário de teste não ter sido testado anteriormente, ou uma situação tão específica demorar a surgir e a ser testada.

Sabemos também que entender a necessidade do cliente e traduzir os requisitos de negócio em requisitos de software é algo complexo, por isso, muitas vezes, os requisitos são mal-entendidos e mal construídos. Por conta disso, são necessárias mudanças corretivas para atender às necessidades do negócio.

Mudanças corretivas, além de gerar alterações no código, geram alterações na documentação técnica do projeto, que precisa ser mantida

atualizada e com controle de versão, para que seja útil para entender como o software foi construído e funciona.

O controle de versão é importante para manter um histórico das alterações realizadas nos produtos de trabalho, de forma a possibilitar a rastreabilidade de todas as modificações realizadas. Lembrando que o rastreio do que foi feito envolve todos os itens de configuração que foram gerados ao longo do desenvolvimento do projeto.

Uma mudança corretiva ainda envolve a correção de defeitos que não foram identificados anteriormente. Não é incomum identificar defeitos mesmo após o software estar sendo utilizado durante algum tempo, por conta dos motivos que já discutimos anteriormente.

Dessa forma, mudanças corretivas envolvem correção de defeitos e correção de requisitos mal-entendidos ou mal construídos, que não atendem à necessidade do negócio.

## 1.2 Mudanças adaptativas

Uma mudança adaptativa ocorre por conta da necessidade de negócio ou por exigência do mercado, que muitas vezes exige que as empresas se adaptem para se manterem competitivas.

Esta situação de mudança de requisito por conta de necessidade de negócio ou por exigência de mercado ocorre pela dinâmica natural dos negócios, que evoluem rapidamente exigindo que os softwares estejam preparados para atender às novas necessidades e às tendências do futuro.

Uma mudança adaptativa pode ocorrer, por exemplo, por questões legais, quando a empresa precisa se adaptar a alguma situação específica.

Por exemplo, quando a Lei Geral de Proteção de Dados (LGPD) entrou em vigor em agosto de 2020, as empresas tiveram que adaptar seus softwares para estarem aderentes aos requisitos exigidos pela lei. Portanto, foi preciso analisar a lei e fazer mudanças adaptativas nos softwares para garantir a segurança dos dados pessoais dos colaboradores, clientes e fornecedores. Na maioria dos casos, não foi necessário desenvolver novos requisitos, mas sim adequar os requisitos já existentes para inserir as questões de segurança exigidas pela lei.

## 1.3 Mudanças evolutivas

Por outro lado, as mudanças evolutivas envolvem a evolução natural do escopo de um software, podendo incluir novos requisitos que surjam com a evolução natural do mercado ou do negócio. Esse movimento é muito comum, por conta do dinamismo do mercado e das necessidades dos clientes.

Por exemplo, quando o mercado financeiro lançou o pagamento com PIX, muitos *websites* tiveram que se adaptar e mudar o escopo do software, para inserir a nova forma de pagamento. Esse caso foi tipicamente uma mudança evolutiva, que surgiu por conta de uma nova necessidade criada pelo mercado. O *website* que não inseriu PIX como forma de pagamento, provavelmente perdeu algumas vendas para outros *websites* que já tinham essa opção para ofertar para seus clientes.

A necessidade de novos requisitos também é algo constante no mundo do desenvolvimento de software, pois tudo é muito dinâmico e as necessidades evoluem com muita velocidade, até mesmo por conta da tecnologia que permite novas e mais eficientes soluções.

Além da mudança adaptativa, como vimos no item anterior, uma mudança evolutiva pode envolver, inclusive, mudanças legais necessárias por imposição dos governos. Esse tipo de mudança é obrigatório para as empresas, sendo passíveis de multa. Por isso, precisam ser implementadas no tempo exigido pelas questões legais, par evitar um custo desnecessário e muitas vezes não previsto para o negócio. A diferença é que uma mudança adaptativa não envolve inserção de novos requisitos, apenas adaptação de requisitos já existentes, ao passo que em uma mudança evolutiva novos requisitos são inseridos no escopo do software.

Dessa forma, uma mudança evolutiva sempre insere novos requisitos no software, para atender às necessidades que surjam, seja para se manter competitivo no mercado seja para expandir os negócios.

Em todo tipo de mudança que envolve alteração ou inclusão de novos requisitos, é preciso atualizar a documentação do software e depois atualizar o código em si.

Manter a documentação atualizada com a evolução do código é um sinal de qualidade do projeto de software, pois garante uma manutenção mais

organizada do software ao longo do tempo. A documentação ajuda a entender como o software foi projetado e construído.

O planejamento realizado para o desenvolvimento do projeto também deve ser revisto com a mudança de escopo, pois é preciso acompanhar o custo e o prazo do desenvolvimento da mudança, para que tudo seja controlado, além de garantir a qualidade do que está sendo construído para atender à necessidade de evolução do software.

As metodologias ágeis, como o *SCRUM*, são exemplos de metodologias que aceitam muito bem as mudanças de requisitos, pois utilizam conceitos de construir o software por partes, que são projetadas e construídas aos poucos. Dessa forma, é possível absorver melhor as mudanças, pois o retrabalho tende a ser menor.

Mas independe de se utilizar uma metodologia ágil ou uma metodologia tradicional para desenvolver o software, é preciso pensar em estratégias para absorver melhor as mudanças, minimizando os riscos para o projeto, seja com alterações indesejadas, seja com falhas no planejamento ou ainda com problemas de falta de qualidade no software. E, é fundamental entender – não existe software sem mudança –, portanto, preparar-se para controlar as mudanças de forma adequada é a melhor solução.

## TEMA 2 – REQUISIÇÃO E GERENCIAMENTO DE MUDANÇAS

Se as mudanças em software são inevitáveis, por conta da dinâmica do mercado e dos negócios, é preciso definir um processo organizado e que cubra as atividades de requisição ordenada e gerenciamento de mudanças.

Nesse sentido, o MPS.Br, que foi discutido em conteúdo anterior, pode auxiliar na definição de um processo completo e que garanta o controle adequado para inserir as mudanças minimizando riscos de impacto no negócio.

O processo de gerenciamento da mudança deve estar organizado desde o momento da requisição da mudança. Ou seja, é preciso que a requisição de mudança esteja bem detalhada, com a especificação do que deve ser feito, de forma clara e completa. Dessa forma, é possível compreender o que precisa ser feito, estimar de maneira assertiva o tempo para o desenvolvimento da mudança, estimar o custo envolvido com a mudança, além dos riscos que podem ser inseridos no software e, principalmente, no negócio da empresa.

Uma vez que a mudança for entendida e planejada, é possível fazer seu gerenciamento adequado, até sua inserção no software que já está em funcionamento.

Dessa maneira, vamos analisar como o MPS.Br sugere tratar essa parte do processo de mudança, para gerenciar as mudanças necessárias a um software.

## 2.1 Processo de gerência de projetos e gerência de requisitos, de acordo com o MPS.Br

Já estudamos que o MPS.Br é organizado em sete níveis de maturidade, todos com processos definidos a serem seguidos, e sendo cumulativos, ou seja, para atingir um nível superior é preciso atender aos processos e seus resultados esperados, do nível anterior, e assim por diante.

Para cuidar adequadamente de um processo de gestão de mudança, é preciso, além do processo de Gerência de Configuração, executar novamente os processos de Gerência de Requisitos e Gerência de Projetos. Faz sentido para você?

Vamos entender o motivo de envolver a Gerência de Requisitos e a Gerência de Projetos, para isso, vamos relembrar o propósito desses processos que fazem parte do primeiro nível de maturidade do MPS.Br, o nível G:

- **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**. Quando inserirmos uma mudança no escopo do projeto, é preciso planejar as tarefas necessárias para o desenvolvimento dessa mudança, identificando o profissional ou profissionais envolvidos no planejamento técnico e construção dela, o custo envolvido na mudança e o tempo necessário para que ela seja analisada, projetada, construída e testada. Dessa maneira, é fundamental voltar ao processo de Gerência de Projetos para, de maneira organizada e padronizada, executar as tarefas que foram realizadas quando o projeto foi planejado pela primeira vez. Fica claro, então, que o processo de Gerência de Projetos precisa ser

executado sempre que houver necessidade de algum novo planejamento para o projeto;

- **Gerência de Requisitos (GRE)**: **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. Seu foco está em gerenciar o escopo definido para o projeto**. Assim como foi feito com o processo de Gerência de Projetos, as tarefas relacionadas com a Gerência de Requisitos também precisam ser executadas novamente, para inserir os novos requisitos que serão criados por contada mudança solicitada para o software. Os requisitos precisam estar claros e completos, e é preciso fazer uma análise de impacto desses novos requisitos com os requisitos já existentes no projeto, para garantir que não há inconsistência entre eles que possam gerar mal-entendidos no momento da construção do código.

Essa é a maneira natural e adequada de inserir os novos requisitos ao projeto, fazendo uma análise crítica dos impactos e minimizando os riscos.

E é preciso lembrar, que dependendo da complexidade da mudança, pode ser necessário, inclusive, voltar aos processos mais técnicos, os relacionados com a engenharia de software em si para analisar se não será necessária nenhuma mudança também no projeto técnico definido para a solução construída.

## 2.2 Processo de Gerência da Configuração, de acordo com o MPS.Br

Além dos processos de Gerência de Requisitos e Gerência de Projetos, o processo de Gerência de Configuração é fundamental para controlar as versões dos componentes criadas por conta de uma mudança, garantindo uma explicação adequada, por meio de comentários, nos produtos de trabalho alterados. Essa disciplina de documentar o que será feita nos produtos de trabalho ajuda a identificar em qual versão uma determinada alteração foi inserida no software, facilitando a recuperação de versão anterior, caso seja necessário.

Vamos relembrar, então, o processo de Gerência de Configuração, para deixar bem clara a importância desse processo para a gestão adequada de qualquer mudança.

A Gerência de Configuração é um processo do segundo nível de maturidade do MPS.Br, o nível F, e tem como propósito:

- **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**. Portanto, como estaremos inserindo novos produtos de trabalho ao projeto, é fundamental manter a integridade desses novos componentes. Tudo o que for inserido no projeto, seja como documentação, seja como código, precisa ser inserido como controle de versão e gerenciado na mesma forma como é feito com os demais componentes criados para o software. Lembrando que a integração dos novos componentes com os componentes já existentes deve envolver também o processo de Integração de Produtos.

Conforme discutido em conteúdo anterior, o processo de gerência de configuração é organizado em três subsistemas, sendo eles:

- 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 estarem completamente prontas;
- O sistema de controle de versões permite que os itens de configuração sejam identificados e evoluídos de forma controlada. Essa característica é necessária para que as diversas solicitações de modificação que possam existir para os requisitos do software possam ser 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.

Os três subsistemas precisam ser executados também no caso das mudanças, pois garantem a padronização e a organização do software como um todo. Toda mudança vai passar a fazer parte do código principal do software, logo, precisa ser incorporada de maneira para segura possível. O pensamento sempre deve ser que uma mudança é uma parte do software que não foi identificada no momento de planejamento inicial, mas que precisa ser inserida de maneira organizada e padronizada no software que já está pronto, ou que está em construção.

Vamos agora analisar os resultados esperados, para compreender quais estão relacionados também com a gestão de mudança.

Já estudamos que, conforme o MPS.Br (2016), o processo de Gerência de Configuração possui sete resultados esperados, sendo eles:

- **GCO1** – é o resultado esperado que estabelece e mantém um sistema de gerência de Configuração, 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 muito provável que esse resultado esperado não seja afetado no caso de uma mudança, pois ele é responsável pela definição de como o processo de gerência de configuração será definido e executado ao longo do projeto, logo, se ele foi bem definido, continuará sendo executado independente de uma mudança, da mesma forma como já estava sendo executado anteriormente**;
- **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. **Esse resultado esperado pode ser acionado para verificar se, a partir dos critérios estabelecidos, os novos componentes gerados por conta da mudança deverão ser tratados também como itens de configuração**;

- **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. **Esse resultado esperado obrigatoriamente deve ser executado, uma vez que novos componentes serão criados e outros poderão ser alterados por conta da mudança. Todos eles, que estiverem sob gerência de configuração, deverão ser gerenciados adequadamente, com o controle formal exigido para componentes alterados ou inseridos em um software que já está sem fases mais avançadas do ciclo de vida de desenvolvimento de software. Os novos componentes ou alterações de componentes deverão fazer parte de uma baseline, quando estiverem prontos para evoluir nas fases do ciclo de vida. Portanto, esse resultado esperado é bastante sensível quando tratamos de mudança de requisitos em um software**;

- **GCO4** – é o resultado que cuida da situação dos itens de configuração e das baselines sendo registradas e disponibilizadas ao longo do tempo, ou seja, cuida do histórico de cada item de configuração. As ações de gerenciamento de configuração como a inclusão e alteração de itens no repositório de componentes, e a geração e 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. **Esse resultado esperado também é acionado em caso de mudança de requisitos, exatamente por conta da sua natureza, pois é o resultado esperado responsável por manter o histórico tanto dos itens de configuração quanto das baselines**;

- **GCO5** – é o resultado responsável por controlar as modificações nos itens de configuração. A partir do momento que os itens de configuração passam a fazer parte de uma baseline, toda e qualquer modificação sobre esses itens de configuração deve passar por um processo formal de controle de modificações. Esse processo formal de controle de modificações visa analisar o impacto das modificações e notificar aos afetados, evitando retrabalho e efeitos colaterais indesejáveis. **Esse resultado esperado também é acionado em caso de mudança de**

**requisitos, exatamente por conta da sua natureza, pois é o resultado esperado responsável pelo monitoramento e controle das modificações dos itens de configuração**;

- **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. **Esse resultado também será executado em caso de mudança, pois todo itens de configuração criados ou alterados deverão ser armazenados, manipulados e liberados da mesma forma como ocorre com os demais itens de configuração**;
- **GCO7** – esse resultado prevê que auditorias de itens de configuração são 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 contém todos os componentes necessários e corretos para compor uma versão do software que será evoluída para a próxima fase do ciclo de vida de desenvolvimento de software. **Esse resultado também deverá ser executado em caso de mudança, uma vez que os componentes necessários para implementar uma mudança foram criados, integrados ao software e disponibilizados para uso, eles também deverão passar pelas auditorias planejadas para garantir a qualidade e assertividade das baselines**.

Dessa forma, é possível compreender que tanto no desenvolvimento inicial quanto nas mudanças necessárias ao software ao longo do seu ciclo de vida, alguns processos previstos pelo MPS.Br precisam ser acionados e executados, de forma a garantir que o software continuará íntegro e funcionando de maneira adequada, sem inserir riscos para o negócio da empresa. O fundamental é que o software se mantenha estável e disponível para uso a maior parte do tempo, aumentando a confiança dos usuários e a lucratividade dos negócios.

## TEMA 3 – TIPOS DE MUDANÇA

Além das mudanças adaptativas, evolutivas e corretivas, ainda é possível classificar as mudanças em planejada ou emergencial.

Classificar uma mudança ajuda a entendê-la melhor, para planejar de maneira adequada sua construção. Além disso, é possível utilizar base histórica de projetos anteriores para avaliar se soluções já utilizadas em outros casos de implantação de mudança podem ser repetidas nas mudanças atuais. Ou seja, é uma forma de prever o resultado de uma mudança e minimizar riscos e de inserção de problemas no software.

É pelo processo de gerenciamento de mudanças que todas as implementações e alterações na infraestrutura de TI ou nos softwares serão analisadas e planejadas para que se tenha o menor risco e impacto. Esse processo tem como objetivo gerenciar todas as mudanças que possam causar impactos na entrega de serviços que mantém a estabilidade do funcionamento dos softwares da empresa.

O processo normal de gerência de mudanças deve ser planejado, analisado criticamente e aprovado, antes de ser implementado e implantado. Mas, deve ser previsto um processo para os casos de fluxo emergencial de tratamento das mudanças para que mudanças urgentes ou mesmo correções causadas por outras mudanças possam ser realizadas de forma mais ágil, vale destacar que esse fluxo emergencial é de caráter extraordinário e que se deve ao máximo respeitar o fluxo normal. As mudanças emergenciais devem ser a exceção e não a regra em uma empresa, minimizando o risco para os negócios.

Vamos compreender um pouco mais o que são as mudanças planejadas e as mudanças emergenciais:

- **Mudança planejada**: é a mudança que segue o fluxo normal de avaliação, aprovação e autorização. O processo de gestão de mudança planejada é um processo institucionalizado para avaliar as mudanças do ponto de vista técnico e gerencial, a fim de minimizar os impactos para o negócio. É importante que sejam definidas as prioridades das mudanças conforme as necessidades do negócio, para garantir que as mudanças mais importantes sejam implementadas e implantadas antes de outras menos importantes. As mudanças devem ser planejadas e agendadas no seu tempo correto, para trazer o benefício esperado;

- **Mudança emergencial**: é uma mudança que pretende reparar um erro em um software que está causando um impacto negativo para a empresa, e que por isso precisa ser corrigido rapidamente. Essa mudança, mesmo sendo emergencial, precisa ser projetada e testada, ou então poderá causar um impacto maior que o inicial. Devem ser tratadas apenas como mudanças urgentes ou emergenciais aquelas que implicam indisponibilidade atual ou imediata de um serviço ou de um sistema de software. Muitas vezes, é preciso usar uma solução secundária apenas para voltar à estabilidade do ambiente, e depois analisar a causa raiz do problema para projetar a solução definitiva, que deve sim ser planejada.

Toda mudança, seja ela planejada ou emergencial, precisa ter um procedimento padrão para retorno da versão anterior do software, como forma de contingência. Não é incomum que uma mudança mal projetada ou planejada gere grandes problemas para o software. Por isso, a gerência de configuração é fundamental nos casos de alteração de componentes, pois pode ser necessário retornar rapidamente a uma versão anterior do software. É fundamental ter o histórico de todas as modificações nos componentes do software, para identificar onde o problema foi inserido, corrigi-lo e voltar com a mudança, para atender à melhoria a que ela se propunha.

Como já falamos anteriormente, um software sempre sofrerá mudanças ao longo de sua vida, pois a inconstância das demandas do mercado e dos clientes exigem que os softwares estejam o tempo todo em adequação. Por isso, é preciso definir um processo de gestão de mudança que seja bom o suficiente para monitorar todo o ciclo de vida da mudança, até sua implantação em produção, inserindo as validações e aprovações necessárias para realizar uma mudança com segurança.

## TEMA 4 – IMPACTO DAS MUDANÇAS

Já discutimos que não existe software sem mudança, pois a mudança é algo inerente de qualquer negócio. Além dessa premissa, é preciso entender também que toda mudança gera algum impacto. Este impacto precisa ser conhecido e gerenciado, para não gerar prejuízo além do aceitável para o negócio.

É por isso que toda mudança precisa ser bem entendida e bem planejada. A análise crítica de uma mudança deve envolver o benefício que ela trará para o negócio e o custo necessário para implementar e implantar a mudança.

Para realizar uma análise técnica criteriosa sobre a mudança, é preciso identificar todos os pontos que podem ser impactados pela alteração, para que esses pontos sejam verificados e gerenciados. A pior coisa que pode ocorrer com uma mudança mal planejada é o surgimento de defeitos não previstos, e, muitas vezes, maiores que o problema que a mudança em si vai resolver.

Para apoiar na identificação de tudo o que pode ser impactado por uma mudança em um software, é utilizada uma técnica chamada Matriz de Rastreabilidade.

Mas vamos entender mais em detalhes o que realmente é rastreabilidade.

## 4.1 O que é rastreabilidade?

O MPS.Br (2016) reforça que a análise de impacto descreve quais itens de configuração serão afetados pela modificação e quais devem ser as correções propostas. Pensando no contexto gerencial, a análise de impacto indica uma estimativa do esforço necessário para realizar a modificação em termos de custo e tempo. Para auxiliar na realização dessa análise, aconselha-se o uso de algum mecanismo que indique a rastreabilidade entre os itens de configuração, como uma matriz de rastreabilidade.

Mas, enfim, o que é rastreabilidade? Bom, a rastreabilidade é um conceito relacionado com os requisitos de um software, que começa a ser identificado no processo de Gerência de Requisitos, na visão do MPS.Br.

Segundo o site Rede Requisitos (2017), a rastreabilidade dos requisitos envolve a identificação dos relacionamentos entre as fontes dos requisitos, entre os requisitos propriamente ditos, e entre requisitos e os outros produtos de trabalho gerados ao longo da construção do software. Além disso, a rastreabilidade pode envolver documentos do projeto, modelos criados na fase de projeto técnico, entre outros.

Portanto, rastreabilidade é a ação de relacionar os produtos de trabalho de um software, partindo os requisitos, que é a unidade mais básica e inicial da construção do software.

Dessa forma, sempre que houver alteração em um requisito, é possível identificar requisitos relacionados aos que sofrerão a alteração, além de outros

produtos de trabalho que também serão impactados. Essa identificação ajuda no momento de planejar a mudança, identificando tudo o que deverá ser analisado, pois são produtos de trabalho candidatos a serem alterados, para tornar a mudança completa e adequada à necessidade do negócio. A identificação dos impactos ajuda a projetar a solução técnica e a planejar a estimativa de custo e prazo necessários para implementar a mudança.

Logo, a rastreabilidade deve ser criada no momento de gestão de requisitos e ser mantida ao longo de toda a construção do software, inclusive durante a implantação de mudanças, quando o software já está em uso intenso pelos usuários. Dessa forma, a rastreabilidade tem vida e precisa ser mantida e gerenciada enquanto o software estiver ativo e em uso.

## 4.2 Exemplo de rastreabilidade

O MPS.Br, apesar de não definir uma solução única de como a empresa deve tratar a rastreabilidade de seus requisitos, o modelo sugere que a rastreabilidade por ser organizada por meio de uma matriz, que é comumente chamada de Matriz de Rastreabilidade, no meio da engenharia de software.

A matriz de rastreabilidade é uma ferramenta simples, mas eficaz para relacionar os requisitos com eles mesmos e com os demais produtos de trabalho de um software. Ela pode ser construída conforme os requisitos vão são sendo identificados e compreendidos, podendo ser gerenciada manualmente ou de forma automatizada, utilizando ferramentas próprias para esse sim. Porém, algumas vezes, essas ferramentas podem ser muito trabalhosas ou até mesmo muito caras. É preciso analisar a melhor solução para a implementação de uma matriz de rastreabilidade, solução que seja viável para o projeto em questão e adequada para a necessidade de controle desejado.

Vamos analisar um exemplo de matriz de rastreabilidade:

Tabela 1 – Exemplo de Matriz de Rastreabilidade

| Requisitos | Requisitos | UC | Classe | Entidade - BD | Código | Caso de Teste |
|---|---|---|---|---|---|---|
| RN01 | RN03 | UC01 | CLIENTE | CLIENTE | Cliente.jsf | CT01 |
| RN02 | | UC02 | PRODUTOS | PRODUTOS | Produto.jsf | CT02 |
| RN03 | | UC01, UC03 | VENDAS | VENDAS | Venda.jsf | CT03 |

| | | | | | | |
|---|---|---|---|---|---|---|
| RN04 | RN05 | UC04, UC05 | CLIENTE | CLIENTE | Cliente.jsf | CT04 |
| RN05 | | UC05 | VENDAS | VENDAS | Venda.jsf | CT05 |

Nesse exemplo, podemos identificar o relacionamento entre os requisitos, identificando quando um deles tem ligação direta com outro, provavelmente, com passagem de parâmetro ou ligação de processamento de dados entre eles. Identificamos como cada requisito é analisado logicamente, gerando um detalhamento sobre ser funcionamento por meio de um caso de uso (UC). Cada caso de uso é implementado por um conjunto de classes e entidade do banco de dados, que faz parte do processamento das informações relacionadas a cada funcionalidade do software.

É possível também identificar o caso de teste que foi criado para garantir a qualidade e o funcionamento adequado de cada requisito. O código também pode ser identificado na matriz, conforme apresentado no exemplo. Mas uma outra solução muito utilizada por vários projetos é utilizar o comentário da ferramenta de versionamento de código, para identificar a qual caso de uso o componente está relacionado. Dessa forma, todo método terá um comentário para identificar a funcionalidade que ele está implementado. É possível, então, identificar, por meio de pesquisa diretamente no código, qual o componente ou componentes estão relacionados com os requisitos que vão sofrer alterações por conta de uma mudança de escopo necessária ao negócio.

Vejam que a criação e a manutenção de uma matriz de rastreabilidade facilitam a análise crítica de uma mudança, por permitir, de forma clara e direta, identificar os produtos de trabalho relacionados com o requisito que sofrerá a mudança. Essa medida simples, mas eficaz, ajuda a não esquecer partes do software que podem ser impactadas por uma alteração, gerando indisponibilidade do software e prejuízo para o negócio.

## TEMA 5 – GESTÃO DE MUDANÇAS

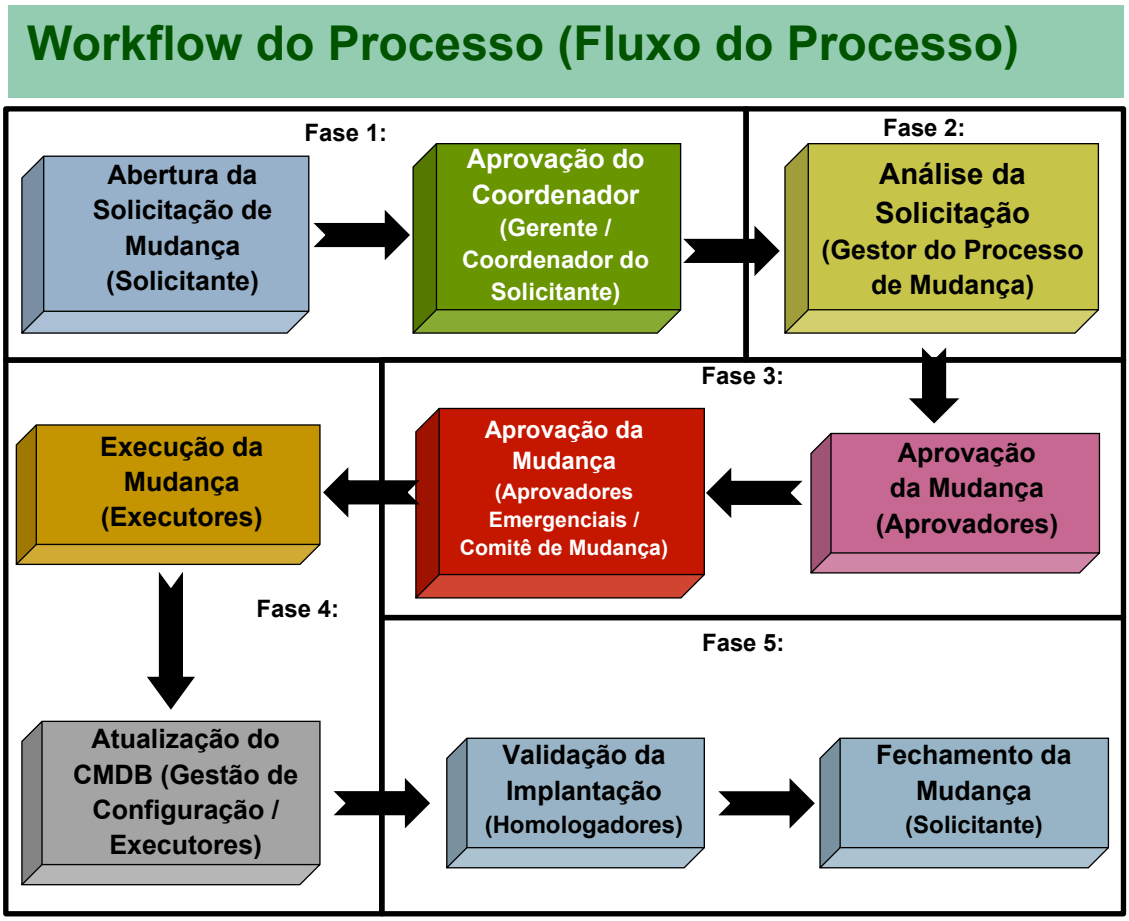
A gestão efetiva de mudanças envolve vários conceitos, para garantir um processo completo e de qualidade. A gestão de mudança pode também ser chamada de GMUD em várias empresas. De forma resumida, uma GMUD é o conjunto de procedimentos e ações necessárias para detectar, implementar e controlar as mudanças necessárias em um software e se certificar de que estão

sendo executadas corretamente e não trazem risco não gerenciado para o negócio.

Cada empresa pode estruturar o seu processo de acordo com sua necessidade e expectativas, mas o processo precisa cobrir as tarefas desde a abertura de uma solicitação de mudança até a implantação e validação dela, com todo o controle e monitoramento necessário para minimizar riscos para o negócio.

Vamos analisar um exemplo de processo:

Figura 1 – Exemplo de processo de gestão de mudança




Fonte: <https://www.webartigos.com/artigos/gerencia-de-mudancas-gmud-controlando-as-intervencoes-no-ambiente-de-producao/12779>. Acesso em: 15 jun. 2022.

Esse processo foi estruturado em cinco fases, de forma a organizar as tarefas ao longo da vida da mudança, desde o surgimento da sua necessidade até a finalização da mudança, com o armazenamento para consulta futura e base de comparação para novas mudanças.

O primeiro passo do processo é a tarefa para abrir a solicitação de mudança, em que deve ser registrado qual a alteração a ser executada. A seguir,

temos o primeiro ponto de controle, com a aprovação do gestor do solicitante. Continuando o processo, o gestor de mudança avalia a solicitação, para verificar se é realmente necessária e o impacto que ela poderá trazer para o software já em funcionamento. A fase 3 envolve mais dois passos relacionados com a aprovação da mudança, uma com os aprovadores principais, caso seja uma mudança planejada, e outra com os aprovadores emergenciais ou com o comitê de mudança, para os casos de mudanças emergenciais. Os critérios de avaliação e aprovação são diferentes, dependendo da criticidade da mudança. O passo 4 é a execução da mudança em si e o registro na ferramenta de gerência de configuração, em que CMDB (*Configuration Management Data Base*) que, em português, significa Banco de Dados utilizando pela Gerência de Configuração para armazenar o histórico de evolução dos itens de configuração. E o passo 5 é composto por tarefas relacionadas com a validação da mudança pelos homologadores, que tem a responsabilidade de verificar se a mudança foi realizada como deveria e se o software como um todo continua funcionando de maneira adequada.

Cada empresa pode definir o processo de gestão de mudança da forma como achar melhor, mas precisa se preocupar com o registro da solicitação de mudança, que deve ser avaliada e aprovada por mais de uma área, para garantir que todos os aspectos da mudança foram analisados. Os modelos como CMMI e MPS.Br ajudam a garantir um processo completo, mas lembrando, eles não dizem como o processo deve funcionar, e sim o que o processo precisa ter para seguir as boas práticas de mercado.

Algumas empresas optam por criar um comitê permanente para avaliar as mudanças planejadas. Esse comitê pode se reunir também de maneira esporádica, caso haja uma mudança emergencial para ser avaliada, e ele costuma se chamar CAB – Change Advisory Board, ou, em português, comitê consultivo de mudança. Vamos entender um pouco mais sobre o papel e as responsabilidades desse comitê.

## 5.1 CAB – Change Advisory Board

O CAB pode se reunir periodicamente ou de maneira esporádica, dependendo da estabilidade dos requisitos, da complexidade das mudanças ou da criticidade dos softwares. Independentemente da periodicidade, o CAB pode ser entendido como um comitê de governança, que reúne algumas pessoas-

chave como analistas de sistemas, analista de segurança da informação, analista de requisitos, arquiteto de sistemas, entre outras, com o objetivo de analisar a mudança a ser realizada. É preciso formalizar a mudança com, no mínimo, as seguintes informações: descrição da atividade, data/hora início da implantação de mudança, data/hora fim da implantação da mudança, possível impacto para o software e para o negócio, risco, plano de retorno, caso a implantação da mudança não funcione conforme o planejado e serviços que ficarão indisponíveis durante implantação da mudança.

No CAB, as informações sobre as mudanças são analisadas com o objetivo de fomentar a reflexão e a discussão de todos os presentes, para que possam verificar os projetos em paralelo e quais são os riscos existentes com cada uma das mudanças em avaliação. Pode ser necessário avaliar mais de uma mudança ao mesmo tempo, dependendo da estabilidade dos requisitos e da quantidade de software existentes na empresa.

Caso exista mais de uma mudança, é preciso verificar se podem estar planejadas no mesmo horário e se precisam readequar a janela da atividade para não existir conflitos e impactos de uma mudança na outra. É nesse momento que as dúvidas sobre as mudanças e seus impactos são tiradas, que os comentários sobre os riscos são feitos e que os membros do CAB avaliam se a mudança pode ser aprovada ou não, com o consenso de todos os presentes.

O CAB pode envolver uma análise preliminar técnica, sobre a viabilidade da mudança. Se tecnicamente a mudança for viável, é feita uma análise gerencial, que verifica o custo X benefício trazido pela implementação e implantação da mudança.

O processo de aprovação de uma mudança deve ser diferenciado para mudanças emergenciais, pois estas, muitas vezes, não podem aguardar todo o processo ser executado para colocar uma mudança em funcionamento, pois o prejuízo para o negócio pode ser grande. Nesses casos, é possível implementar uma solução paliativa, para resolver o problema de imediato, enquanto é feita uma análise mais criteriosa sobre a solução definitiva para o problema. Mas é preciso se atentar para que tudo não se torne emergencial por falta de planejamento. Uma mudança emergencial deve ser uma exceção na empresa, e não a rotina.

A implantação de uma mudança precisa ser avisada a todos os envolvidos, de forma a deixar a comunicação clara e disponível para todos sobre

o dia e o horário da sua implantação, pois os sistemas param de funcionar no período em que estiver ocorrendo a implantação.

Outros conceitos importantes podem ser inseridos no processo para garantir uma melhor análise da mudança a ser executada e um melhor teste do software como um todo, minimizando impactos e reduzindo riscos para o negócio. Vamos discutir sobre alguns tipos de testes que podem ser executados em um software.

## 5.2 Tipos de testes de software

Um tipo de teste bastante utilizado não apenas em componentes relacionados com mudanças de software, mas em qualquer componente ou conjunto de componentes construídos, é um teste relacionado com a identificação da causa raiz de um defeito. Ou seja, nem sempre é fácil identificar um defeito em um software. Dessa forma, pode-se utilizar o método FCA (ou fato-causa-ação), que consiste em identificar o fato, ou problema em questão, sua causa e definir uma ação para resolver o problema. A grande vantagem do método é que utilizando a análise do fato, é possível chegar à causa raiz de qualquer problema o que facilita a definição da melhor ação ou solução para realmente resolver a questão.

Outro tipo de teste que também pode ser utilizado ao longo de todo o desenvolvimento do software, até em mudanças de requisitos, é o teste chamado de s*moke testing* ou, em português, teste de fumaça. O *smoke testing* é um tipo de teste básico que verifica as funcionalidades básicas de um software. É um processo que deve ser executado de forma rápida para determinar se a compilação do software está estável ou não. É uma verificação que ajuda a equipe a decidir se avança ou não para novos testes. Consiste em um conjunto mínimo de testes para validar as principais funcionalidades, tanto originais do software, quanto da mudança implementada. O principal objetivo desse tipo de teste é dar a garantia de que as principais características do software estão funcionando como esperado, ou informar problemas iniciais que possam ser rapidamente corrigidos.

É importante também utilizar um teste de recursividade, que teste não apenas a mudança em si, mas todos os componentes que podem ter sido impactados pela alteração. Para a identificação dos componentes que podem ter

sofrido impacto com a alteração, é utilizada a matriz de rastreabilidade que foi discutida anteriormente nesta etapa.

## FINALIZANDO

Nesta etapa, discutimos sobre a importância de definir um processo adequado para fazer a gestão de mudanças. Discutimos que entender os vários tipos de mudança é fundamental para escolher o melhor processo para ser aplicado.

Discutimos sobre a necessidade de registrar a solicitação de mudança para garantir o entendimento de todos que vão projetar a solução técnica para atender à alteração. Uma mudança pode envolver a execução novamente de vários processos relacionados com o ciclo de vida do desenvolvimento de software, pois é preciso garantir que o planejamento adequado da mudança seja feito exatamente como foi feito inicialmente, quando o projeto foi iniciado.

Vimos a importância de criar e manter uma matriz de rastreabilidade, iniciando no processo de Gerência de Requisitos e sendo mantida ao longo do processo de Gestão de Mudança, de forma a facilitar a identificação dos possíveis produtos de trabalho impactados pela mudança, evitando, assim, instabilidades ou até mesmo indisponibilidade no software.

Conversamos sobre testes importantes de serem realizados para garantir que a mudança implementada não gere nenhum impacto não previsto no funcionamento do software como um todo.

Compreendemos sobre as responsabilidades que envolvem o comitê de mudança ou o CAB, na análise crítica de uma mudança, avaliando não apenas os aspectos técnicos, mas também os aspectos gerenciais. Uma mudança precisa ser viável tecnicamente e gerar um ganho para o negócio. O CAB deve também definir o melhor dia e horário para que a mudança seja implantada, para que impacte o menos possível o resultado dos negócios, isso porque, no período em que ocorre a implantação, os softwares envolvidos na mudança param de funcionar.

Vamos continuar evoluindo e aprofundando nos conceitos que envolvem a Gerência de Configuração, posteriormente.

## REFERÊNCIAS


ANDRADE JUNIOR, J. R. de. **Gerência de configuração**. 1. ed. Pearson, 2014.

MORAIS, I. S. de. **Engenharia de software**. Porto Alegre: Sagah, 2017.

MUNIZ, A.; SANTOS, R. **Jornada Devops**: unindo cultura ágil, Lean e tecnologia para entrega de software com qualidade. 1. ed. Brasport, 2019.

PAULA FILHO, W. de P. **Engenharia de softwares**: produtos. 4. ed. Rio de Janeiro: LTC, 2019.

PFLEEGER, S. L. **Engenharia de software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.

REDE REQUISITOS. Gerência de Requisitos de software e a Matriz de Rastreabilidade. 2017. Disponível em: <http://rederequisitos.com.br/gerencia-de-requisitos-de-software-e-a-matriz-de-rastreabilidade/>. Acesso em: 15 jun. 2022.

SBROCCO, J. H. T. de C. **Metodologias ágeis**: engenharia de software sob medida. 1. ed. São Paulo: Érica, 2012.

[SOFTEX, 2016]. ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – SOFTEX. **MPS.BR** – Guia Geral MPS de Software:2016. 2016. Disponível em: <www.softex.br>. Acesso em: 6 maio 2022.

[SOFTEX, 2011c] ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – SOFTEX. **MPS.BR** – Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS:2011. 2011. Disponível em: <www.softex.br>. Acesso em: 6 maio 2022.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2019.

VETORAZZO, A. de S. **Engenharia de software**. Porto Alegre: Sagah, 2018.