GERÊNCIA DE CONFIGURAÇÃO E EVOLUÇÃO

AULA 1

Profª. Adriana Bastos da Costa

CONVERSA INICIAL

Desenvolver um *software* não é algo trivial, pois envolve uma série de variáveis relacionadas com o entendimento das necessidades e requisitos do cliente, além de saber interpretar esses requisitos e transformá-los em linhas de código, que vão compor o *software* que será utilizado pelos usuários.

Um outro ponto importante que deve ser ressaltado é a importância de evoluir o *software* de maneira controlada. Ou seja, uma vez o *software* construído e em uso, novas necessidades de evolução e alteração podem surgir. O gerenciamento das alterações também é um fator de sucesso para a manutenção do *software*.

É sobre esses conceitos que vamos conversar nesta etapa, discutindo sobre a importância da gerência de configuração para a qualidade do *software*.

Esta etapa está dividia em 5 tópicos, sendo eles:

- Mudanças em *software*;
- Gerenciamento de mudança *versus* gerenciamento de configuração;
- Gerência de configuração e qualidade do *software*;
- Profissional de gerência de configuração;
- Itens de configuração.

Vamos explorar o mundo da gerência de configuração utilizando-nos dos conceitos básicos, mas necessários para compreender o objetivo e o funcionamento do controle dos componentes de um *software*.

TEMA 1 – MUDANÇAS EM *SOFTWARE*

Por ser um processo complexo, construir *software* envolve entender o problema que precisa ser automatizado, interpretar as necessidades e transformar essas necessidades em um *software* que agregue valor para o negócio do cliente.

Segundo o DVMedia (S.d.), os sistemas de *software* estão em constante evolução. A manutenção do *software*, muitas vezes, chega a consumir 75% do custo total gasto durante todo o seu ciclo de vida de desenvolvimento do software. Cerca de 20% de todo o esforço de manutenção é usado para consertar erros de implementação e os outros 80% são utilizados na adaptação

e evolução do *software* por conta de modificações em requisitos funcionais, regras de negócios e na reengenharia da solução inicial do *software*.

Portanto, a gerência de configuração surgiu da necessidade de controlar essas modificações, por meio de processos, métodos e ferramentas, com o intuito de maximizar a produtividade e minimizar os erros cometidos durante a evolução do escopo de um *software*.

Dessa forma, mudanças em *software* é algo extremamente normal no contexto de projetos de *software*. Essa premissa nos leva a entender a importância de gerenciar as mudanças que ocorrem no escopo de um *software*.

Todo *software* possui um escopo, que é a descrição do que deve ser feito em termos de requisitos e funcionalidades.

Gerenciar as mudanças em um *software* envolve documentar o que está sendo alterado e manter um histórico das diversas versões de cada componente que faz parte do *software*, para que seja possível recuperar versões anteriores, caso seja necessário.

Os componentes de um *software* podem ser entendidos como todo produto de trabalho gerado ao longo da construção desse *software*, seja código ou até mesmo documentação gerada para planejar o projeto e para construir solução técnica que será aplicada.

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

Vamos detalhar cada um desses tipos de mudança a seguir.

1.1 Mudanças corretivas

Uma mudança corretiva envolve a correção de defeitos ou de mau entendimento dos requisitos.

Como entender a necessidade do cliente e traduzir os requisitos de negócio em requisitos de *software* é algo complexo, pois muitas vezes os requisitos são mal-entendidos e malconstruídos. Por isso, são necessárias mudanças corretivas para atender às necessidades do negócio.

Mudanças corretivas podem gerar, além de alterações no código, alterações também na documentação técnica do projeto, que precisa ser mantida atualizada e com controle de versão.

O controle de versão significa manter um histórico das alterações realizadas nos produtos de trabalho, de forma a possibilitar o rastreio de todas as modificações realizadas. Lembrando que esse rastreio envolve componentes de código e documentação que foi gerada ao longo do desenvolvimento do projeto.

Uma mudança corretiva ainda envolve a correção de defeitos que não foram identificados anteriormente. Parece estranho, mas em *software*, muitas vezes existem tantos cenários diferentes que nem sempre é possível identificar todas as possibilidades e garantir que todas estejam funcionando adequadamente. É por isso que não é incomum identificar defeitos mesmo após o *software* estar sendo utilizado durante algum tempo.

Dessa forma, mudanças corretivas envolvem correção de defeitos e correção de requisitos mal-entendidos ou malconstruídos, que não atendem a necessidade do negócio. E até mesmo, a mudança de algum requisito por conta da necessidade de negócio ou por exigência do mercado.

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.

1.2 Mudanças evolutivas

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.

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.

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.

Além das mudanças legais, podem ser necessárias mudanças para inserir novos requisitos no *software*, para atender às necessidades que surjam, seja para se manter competitivo no mercado, seja para expandir os negócios.

Quando é necessária uma mudança evolutiva, seja por questões legais, seja por questões de negócio, é preciso atualizar a documentação do *software* e depois atualizar o código em si.

Manter a documentação atualizada junto 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.

O planejamento realizado para o desenvolvimento do projeto também deve ser feito para uma 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*.

TEMA 2 – GERENCIAMENTO DE MUDANÇA *VERSUS* GERENCIAMENTO DE CONFIGURAÇÃO

Segundo o guia geral do MPS.BR (Softex, 2016), “o propósito do processo de gerenciamento 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”.

Ressaltando que os produtos de trabalho são todos os componentes gerados ao longo do desenvolvimento de um projeto, sejam componentes de código ou de documentação.

O gerenciamento de mudança envolve o controle de versão e de alteração dos componentes de trabalho do projeto. Toda evolução deve ser registrada e controlada a partir de uma ferramenta que apoia o versionamento de componentes, automatizando esse processo de controle.

É possível fazer o gerenciamento de versão e o controle de mudanças de forma manual, por meio de um processo definido que organize as tarefas necessárias para gerar um controle das versões dos produtos de trabalho do software. Mas é muito mais fácil e produtivo utilizar uma ferramenta própria para essa função.

Toda alteração em um *software* pressupõe a alteração em algum requisito ou funcionalidade definida para o *software*. Dessa forma, a gerência de configuração e o controle de alterações estão diretamente relacionados com o controle do escopo de um projeto de *software*.

Vamos entender o que é o gerenciamento de mudanças e o gerenciamento de configuração no contexto de projeto de *software*.

2.1 Gerenciamento de configuração

Vamos entender o que é o gerenciamento de configuração no contexto de projeto de *software*. Para entender o conceito, vamos seguir o que é explicado pelo MPS.Br, que é um modelo brasileiro para a melhoria de *software*.

Traduzindo em outras palavras, o MPS.Br é um modelo de qualidade que tem como foco a melhoria do processo de desenvolvimento de *software*, definindo boas práticas que envolvem todo o processo de engenharia de software, indo desde o entendimento dos requisitos passando pela análise, projeto técnico e construção do *software*, até os testes, para garantir a qualidade do que será entregue ao cliente e disponibilizado em produção para utilização do usuário final.

Segundo o Softex (2016),

> durante o processo de desenvolvimento, o sistema de Gerência de Configuração é fundamental para prover controle sobre os produtos de trabalho produzidos e modificados por diferentes engenheiros de software. Além disso, esse sistema possibilita um acompanhamento minucioso do andamento das tarefas de desenvolvimento.

De acordo com o DVMedia (S.d.), a gerência de configuração é uma disciplina que controla e gerencia as inúmeras correções, evoluções e adaptações aplicadas durante o ciclo de vida do *software* de forma a assegurar um processo de desenvolvimento e evolução controlado, sistemático e rastreável, sendo indispensável quando equipes manipulam, muitas vezes em conjunto, os produtos de trabalho do *software*.

Analisando o ciclo de vida de desenvolvimento de *software*, a Gerência de Configuração entra como um processo de suporte, servindo de base para os demais processos que fazem parte do ciclo de desenvolvimento de *software*, conforme detalhado na figura a seguir:

Figura 1 – Ciclo de vida de desenvolvimento de *software*

O ciclo de vida de desenvolvimento de software possui fases bem definidas, tais como:

- Iniciação do projeto – é a fase inicial, quando o projeto está sendo vendido e o escopo macro está sendo entendido. Nessa fase, é quando o contrato para o desenvolvimento do *software* é definido e assinado;
- Desenvolvimento – é a fase que se inicia após a venda do projeto, ou seja, com o início em si do projeto. Esta fase possui ainda subfases responsáveis por entendimento e análise dos requisitos, que é a fase de entendimento lógico sobre o *software*; *design e projeto técnico*, que é a fase do entendimento físico do projeto, quando a solução técnica para o projeto é planejada e construída; *codificação*, é a fase onde os projetos lógico e físico são implementados por meio de uma linguagem de programação, gerando linhas de código; e por último, a *fase de testes*, em que são executados processos para garantir que o *software* está funcionando como previsto e entregando os requisitos corretos, conforme necessidade dos usuários;
- Operação – manutenção e suporte – é a fase que ocorre após o projeto estar concluído, aprovado e colocado em produção para seu uso pelos usuários finais. Esta é a fase em que ocorrem as mudanças evolutivas e corretivas do *software*;
- Gerência de projeto – é uma fase de apoio, que ocorre ao longo de todas as fases de iniciação, desenvolvimento e operação do projeto. Todo projeto de *software* precisa ser planejado antes de ser executado, para garantir que

todas as necessidades foram previstas e estarão disponíveis no momento correto, para possibilitar um desenvolvimento contínuo, sem interrupções ao longo do ciclo de vida do desenvolvimento do *software*.

- Garantia da qualidade – é uma fase de apoio, responsável por definir e aplicar as ações de qualidade necessárias para que o *software* seja construído da melhor forma possível, pensando em boas práticas de programação e de gestão de projetos de *software*.
- Gerência de configuração – é uma fase de apoio, responsável pelos processos relacionados com a gerência de configuração que deve ser aplicada durante todo o ciclo de vida do *software* e não apenas durante o ciclo de vida do seu desenvolvimento. A gerência de configuração, conforme apresentada pela Figura 1, faz o controle de todos os produtos de trabalho gerados para o *software*, incluindo os produtos de trabalhos gerados pelas demais fases de apoio e os produtos de trabalho gerados na fase de operação do *software*.

2.2 Gerenciamento de mudança

O gerenciamento ou controle de mudança faz parte da gerência de configuração, como uma de suas responsabilidades.

A gerência de mudança é responsável por fornecer um serviço complementar ao serviço oferecido pelo sistema de controle de versão onde o foco desse tipo de ferramenta está nos procedimentos pelos quais as mudanças de um ou mais itens de configuração são propostas, avaliadas, aceitas, desenvolvidas e aplicadas no *software*. Pode envolver mudança em qualquer produto de trabalho gerado para o *software*, seja documentação ou código.

O controle de mudanças executa as seguintes tarefas:

- Identificar as mudanças nos itens de configuração;
- Rastrear as mudanças nos itens de configuração;
- Analisar as mudanças nos itens de configuração;
- Controlar as mudanças nos itens de configuração;

Mas, afinal, o que é um item de configuração? Para compreender exatamente tudo o que envolve o controle de mudanças, é preciso também conhecer o conceito de itens de configuração.

Para isso, precisamos entender que a gerência de configuração é um processo que tem seu início com a identificação das partes que constituem o *software*.

Essas partes, denominadas *itens de configuração*, representam a agregação de componentes de código do software e da documentação criada como parte integrante do *software*, tratados pela gerência de configuração como itens de configuração. Em função da granularidade utilizada em um contexto de *software*, um item de configuração pode ser formado por um conjunto de produtos de trabalho, bem como um único produto de trabalho pode ser formado por vários itens de configuração.

O que define como um item de configuração é constituído pela complexidade e pela necessidade do projeto, lembrando que o objetivo principal é gerenciar e controlar as alterações e as versões geradas nos itens de configuração ao longo da vida do *software*. Dessa forma, é possível tratar item de configuração como um sinônimo de produto de trabalho gerado para o *software*, quando for definido que um produto de trabalho é um item de configuração que estará sob a gerência de configuração, fazendo parte do processo definido para garantir o controle de versão, o controle das mudanças e a integração contínua do código criado ou alterado.

Fica claro então que a gerência de mudança e a gerência de configuração possuem objetivos diferentes, apesar de estarem intrinsecamente relacionadas. Uma das responsabilidades da gerência de configuração é, portanto, gerenciar as mudanças necessárias ao *software*, para mantê-lo sempre útil e agregando valor ao negócio do cliente.

TEMA 3 – GERÊNCIA DE CONFIGURAÇÃO E QUALIDADE DE *SOFTWARE*

A gerência de configuração de *software* deve ser aplicada durante o desenvolvimento do *software*, mas também depois de sua entrada e uso em produção. O ciclo de vida de um *software* envolve também a etapa de manutenção, que ocorre pela necessidade de evolução e correção no *software* identificadas durante seu uso massivo pelos diferentes usuários.

Por conta da evolução constante das necessidades e exigências do mercado, temos a certeza de que, enquanto o *software* estiver sendo utilizado, haverá necessidade de manutenção no mesmo.

Dessa forma, enquanto o *software* existir, o uso dos processos e sistemas de gerência de configuração serão fundamentais para prover controle sobre os artefatos produzidos e modificados por diferentes profissionais desde o planejamento e levantamento de requisitos até a construção e entrega do produto, e mesmo depois disso, enquanto o *software* for utilizado.

O motivo da sua importância está geralmente associado aos problemas identificados quando a gerência de configuração não é utilizada no desenvolvimento de *software*, que afetam principalmente retrabalho e falta de qualidade na construção e manutenção do *software*.

É por isso que a gerência de configuração é considerada uma disciplina que também agrega qualidade ao desenvolvimento do *software*. Vamos entender um pouco mais sobre o relacionamento entre a gerência de configuração e a qualidade no desenvolvimento do *software*.

3.1 Qualidade no desenvolvimento de *software*

Quando falamos em qualidade no desenvolvimento de *software*, estamos preocupados com a construção e o controle do processo utilizado para construir um *software*, uma vez que já vimos que construí-lo não é algo trivial, pois envolve compreender as necessidades do cliente e transformar os requisitos de negócio em requisitos técnicos que poderão ser transformados em linhas de código.

Um processo de desenvolvimento de *software* deve conter passos bem definidos que envolvam também o processo de gerência de configuração, que é um processo de suporte dentro do ciclo de vida do desenvolvimento de *software*.

A gerência de configuração possui três pilares fundamentais para seu bom funcionamento, sendo eles o controle de versão, o controle de mudança e a integração contínua.

Figura 2 – Pilares da Gerência de Configuração

Os três pilares influenciam diretamente a qualidade do *software* que é gerado. Vamos detalhar cada um deles:

- **Controle de versão**: tem como objetivo apoiar as atividades de controle de mudança e integração contínua, envolvendo as seguintes tarefas: identificação, armazenamento e gerenciamento dos itens de configuração e de suas versões durante todo o ciclo de vida do *software*; manter o histórico de todas as alterações efetuadas nos itens de configuração, de forma a permitir recuperar uma versão anterior, caso seja necessário; possibilitar a criação de rótulos e ramificações no projeto, de forma a organizar o ambiente de desenvolvimento, facilitando encontrar o que se deseja e manter uma manutenção mais facilitada; oferecer a possibilidade de recuperar uma configuração específica mais antiga do *software*, sempre que necessário;
- **Integração contínua:** tem como objetivo garantir que as mudanças ao longo do projeto sejam desenvolvidas, testadas e integradas tão logo quanto possível depois de serem introduzidas no código do *software*. Além disso, a integração contínua também permite integrar com frequência as alterações no código realizadas por toda a equipe de desenvolvedores, de maneira organizada e contínua, permitindo acompanhar a evolução da construção do *software*; propiciar a utilização de um processo automatizado, facilitando a

vida da equipe de desenvolvimento; possibilitar que cada integração seja verificada visando detectar erros de integração o quanto antes, minimizando o esforço com o retrabalho e a identificação de erros; permitir a notificação de toda a equipe quando um problema na integração é identificado, melhorando a correção dos problemas encontrados e gerando um código com mais qualidade e padronizado;

- **Controle de mudanças**: tem como objetivo fornecer um serviço complementar ao oferecido pelo sistema de controle de versão, permitindo controlar, por meio de comentários, todas as mudanças necessárias em um *software*. Tem como foco controlar os procedimentos pelos quais as mudanças de um ou mais itens de configuração são propostas, avaliadas, aceitas e aplicadas. Esse controle permite também identificar todas as mudanças e o objetivo de cada uma, através da disciplina da equipe do projeto em documentar o que está sendo feito. É possível até criar uma rastreabilidade entre documentação e código, amarrando o impacto das alterações em todos os produtos de trabalho gerado para o *software*.

Dessa forma, é possível compreender a importância da gerência de configuração para 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.

TEMA 4 – PROFISSIONAL DE GERÊNCIA DE CONFIGURAÇÃO

Uma grande vantagem da área de TI é que existem infinitas possibilidades de atuação para um profissional formado na área.

É muito comum os profissionais pensarem que a área de TI se restringe a profissionais especializados em desenvolvimento de *software*. Mas isso é um grande engano, pois o desenvolvimento de *software* é apenas pequena parte das possibilidades oferecidas pela área da tecnologia da informação, ou TI, como é mais conhecida.

É possível atuar como administrador de banco de dados, especialista em negócios, analista de segurança da informação, designer de experiência do usuário (UX – *user experience*), gerente de projetos, cientista de dados, entre outras possibilidades.

O analista de configuração e o analista de DevOps são mais duas atuações possíveis para o profissional de TI, cada um com responsabilidades e papéis próprios e até mesmo complementares dentro de uma equipe de desenvolvimento de *software*.

Vamos entender o que fazem esses papéis dentro de um projeto de *software*.

4.1 Analista de configuração

Dependendo da empresa e da sua estrutura de trabalho, o profissional responsável para execução do processo de gerência de configuração pode ser chamado de analista ou de gerente. Mas, independente no nível hierárquico, o papel desse profissional é garantir que o processo de gerência de configuração seja cumprido pelos times de projetos de *software*.

Dessa forma, esse profissional é o responsável pelo processo de gerência de configuração do projeto, disciplina que define o conjunto de atividades necessárias para a gestão de *baselines*, itens de configuração e requisições de mudança do projeto.

O gerente ou analista de configuração e mudança disponibiliza o ambiente e a infraestrutura geral de gerência de configuração para a equipe de desenvolvimento do produto.

A função do analista ou gerente de configuração é oferecer suporte à atividade de desenvolvimento de produtos de *software* para que os desenvolvedores e integradores tenham espaços de trabalho adequados para criar e testar seus trabalhos durante as fases de construção de um *software*.

O gerente ou analista de configuração também deve assegurar que o ambiente de gerência de configuração facilite a revisão do produto e as atividades de controle de mudanças e defeitos, além de facilitar a evolução do produto, mesmo após sua entrada no ambiente de produção.

O papel do gerente ou analista de configuração é importante durante todo o ciclo de vida de um *software*, até que este esteja obsoleto e seja substituído por outra solução, que também passará pela gerência de configuração, e assim continuamente.

4.2 Analista de DevOps

Por conta das necessidades impostas ao desenvolvimento de *software*, necessidades estas que envolver gerar e disponibilizar *softwares* cada vez mais rapidamente e com uma qualidade melhor, foi necessário inserir um novo conceito no ciclo de vida de desenvolvimento de *software*, chamado de DevOps.

A palavra DevOps é a combinação dos termos *desenvolvimento* e *operações*, unindo e dando o mesmo nível de importância para as fases de desenvolvimento e manutenção de um *software*. Dessa forma, DevOps inclui segurança, maneiras colaborativas de trabalhar, análise de dados e capacidade dos servidores, além de outras práticas e conceitos.

A metodologia DevOps envolve definição de um processo com passos bem definidos, mas também pode ser entendido como a combinação de uma forma de pensar o desenvolvimento de *software*, uma cultura que envolve disciplina das equipes com práticas e ferramentas que aumentam a capacidade de uma empresa de construir e distribuir *softwares* em alta velocidade, aumentando a competitividade dos negócios e a qualidade do que é construído.

O conceito ou metodologia descreve abordagens e estratégias que ajudam a acelerar de maneira controlada, os processos necessários para levar uma ideia de *software* do desenvolvimento à implantação em um ambiente de produção no qual ela seja capaz de gerar valor para o usuário final. Essas ideias podem ser um novo recurso de *software*, uma solicitação de aprimoramento ou uma correção de *bug*, entre outras alterações ou mudanças necessárias para o *software*.

A base principal do DevOps é a comunicação frequente entre as equipes de desenvolvimento e operações, trabalho colaborativo e empatia com os demais membros das equipes, para que o trabalho flua sempre pensando no objetivo maior que é entregar um *software* de qualidade e que funcione de maneira adequada para os cientes no ambiente final de uso. Também é necessário pensar em um provisionamento flexível e escalabilidade, poiso *software* pode precisar ser evoluído continuamente, aumentando sua capacidade de processamento e espaço de memória. Os desenvolvedores, que normalmente criam códigos em um ambiente de desenvolvimento padrão, trabalham de forma colaborativa com a equipe de suporte e manutenção, para acelerar a compilação de programas de *software*, a realização

de testes e o lançamento de soluções, sempre preocupados com a confiabilidade e a estabilidade do *software*.

Ou seja, com a implementação de um modelo de DevOps, as equipes de desenvolvimento e operações não ficam mais separadas. Dependendo da organização da empresa, essas duas equipes podem até ser combinadas em uma só. Em alguns modelos de DevOps, as equipes de controle de qualidade e segurança também podem se juntar com as equipes de desenvolvimento e de operações e todo o ciclo de vida dos aplicativos. Esse formato, unindo também as equipes de qualidade e segurança passa a se chamar DevSecOps.

Essas equipes usam práticas para automatizar processos, além de tecnologia e ferramentas que os ajudam a operar e desenvolver aplicativos de modo rápido e confiável.

O analista de DevOps é o profissional que faz a ponte entre a equipe de desenvolvimento e as demais equipes envolvidas na construção do *software*, como a equipe de suporte, de qualidade e de segurança.

O principal objetivo do papel do analista de DevOps é garantir que o processo está sendo seguido e que esse processo é o melhor para o resultado esperado para o *software*.

Se o processo precisa ser alterado por alguma questão, também é o analista de DevOps que atua no entendimento das alterações e na implantação dessas alterações para evoluir o processo, sempre que necessário. Ao evoluir o processo de gerência de configuração, é preciso garantir que todos os envolvidos sejam informados das mudanças e sejam treinados no novo processo. As alterações no processo, publicação, divulgação e treinamento também faz parte do papel do analista de DevOps, uma vez que ele é o responsável por garantir que todo o processo definido seja seguido. Esse é o caminho para melhorar a qualidade do *software* gerado, diminuindo o retrabalho e o esforço para a criação e manutenção dos produtos de trabalho gerados em um projeto de *software*.

TEMA 5 – ITENS DE CONFIGURAÇÃO

Como vimos anteriormente, a gerência de configuração usualmente se inicia na identificação das partes que constituem o *software*. Essas partes são denominadas *itens de configuração*.

Logo, todo produto de trabalho que deve ter sua evolução controlada por meio do controle de versões deve estar sob gerência de configuração.

Por exemplo, vamos imaginar que em um projeto de *software* temos os seguintes produtos de trabalho:

- Cronograma (documento que contém todas as tarefas previstas em um projeto para entregar o objetivo previsto em contrato);
- Plano de Projeto (documento previsto pelo PMBOK para planejar a execução de um projeto);
- Diagrama de Classe (diagrama da UML utilizado para descrever as classes e o relacionamento entre elas, em um *software* construído seguindo os preceitos da orientação a objetos);
- Estórias de Usuários (documento utilizado em metodologias ágeis para descrever os requisitos que serão implementados no *software*);
- Modelo de Dados (documento que contém o modelo projetado para o banco de dados que será utilizado para persistir as informações necessárias para a solução de *software* definida);
- Código (envolve todas as classes e métodos implementados para atender ao escopo definido para o *software*).

A equipe do projeto, pensando nas boas práticas de construção de *software* e na necessidade do projeto pode definir pela seguinte estratégia:

- Cronograma – não é um item de configuração, por ser um produto de trabalho que evolui ao longo do desenvolvimento do projeto, não sendo necessário controlar versões, uma vez que a última versão sempre será a versão válida para acompanhar o status de desenvolvimento do projeto. A evolução das tarefas previstas para o desenvolvimento do projeto é algo contínuo;
- Plano de projeto – é um item de configuração por ser um documento que mostra o planejamento do projeto, e por isso mesmo, deve ter suas versões controladas e também as alterações necessárias para refletir uma nova estratégia para o projeto;
- Diagrama de classe – é um item de configuração porque faz parte da definição técnica da solução do projeto, e deve ter as versões e alterações controladas e monitoradas. As alterações nas classes afetam diretamente a construção do *software* e o código em si;

- Estórias de usuários – é um item de configuração porque faz parte do entendimento das necessidades e funcionalidades que o *software* deve implementar, por isso é fundamental ter as versões e alterações controladas e monitoradas, para garantir que o que será construído é exatamente a necessidade do usuário final. As alterações nas estórias de usuário afetam diretamente a construção do *software* e o código em si;
- Modelo de dados – é um item de configuração porque faz parte da definição técnica da solução do projeto, e deve ter as versões e alterações controladas e monitoradas. As alterações no modelo dedados afetam diretamente a construção do *software* e o código em si. O modelo de dados deve refletir as funcionalidades descritas nas estórias de usuários, portanto, a evolução destes produtos de trabalho deve ocorrer em conjunto;
- Código – é o produto de trabalho criado e alterado de maneira colaborativa pela equipe de desenvolvimento, por isso mesmo seu controle deve ser minucioso, para evitar perdas de alterações e mudanças desnecessárias ou que insiram algum defeito no *software*. É um produto que sempre deve ser tratado como um item de configuração e estar sob gerência de configuração.

A regra de quais produtos de trabalho devem ser tratados como itens de configuração é própria de cada projeto e deve ser definida pela equipe de desenvolvimento no início do projeto. Todos os produtos de trabalho definidos como itens de configuração serão controlados e estarão sob o processo de gerência de configuração.

5.1 Gerenciando os itens de configuração

Discutimos então que todos os produtos de trabalho que forem definidos como itens de configuração deverão estar sob gerência de configuração, tendo, portanto, controle de versão, controle de alteração e passando por integração contínua.

A integração contínua é o pilar que requer a utilização de um repositório único para armazenar o código criado e alterado pela equipe de desenvolvimento. Esse pilar também requer a disciplina de todo o time para,

periodicamente, atualizar o repositório único do projeto com o código desenvolvido individualmente na máquina própria de cada desenvolvedor.

A periodicidade de atualização do repositório único, que deve estar no servidor de desenvolvimento do projeto, deve ser definida pelo time de desenvolvimento em conjunto com o time de operação, baseado na complexidade do projeto, tamanho da equipe e velocidade de produção dos desenvolvedores. A ideia é que, em periodicidade adequada à realidade do projeto, todos os desenvolvedores façam o *upload* de código trabalhado durante um período de tempo par ao repositório único do projeto. Esse repositório deve estar sob processo de *backup (*cópia*) e restore (*recuperação*),* para dar mais segurança ao processo de construção como um todo. Além disso, com a integração contínua é possível monitorar a qualidade do código construído e fazer os ajustes e correções necessárias para manter um código padronizado e seguindo as boas práticas de programação definidas para o projeto.

A integração contínua pressupõe a compilação de todo o código construído, de forma a gerar um executável que será testado e entregue para o cliente.

O gerenciamento dos itens de configuração envolve, além do controle de versão e de alterações, o agrupamento e a verificação da integridade desses itens. O agrupamento de itens de configuração é chamado de *linha de base* ou *baseline*, em inglês. Esse agrupamento de itens de configuração representa um conjunto de produtos de trabalho formalmente aprovados que servem de base para a evolução para as etapas seguintes do ciclo de vida do desenvolvimento de software.

Dessa forma, com a utilização de processos formais de controle de modificações sobre as *baselines*, o processo gerência de configuração atinge o seu propósito principal, que é manter a integridade dos produtos de trabalho.

5.2 Trabalhando com linhas de base (*Baselines*)

O grande objetivo de uma *baseline* é criar um conjunto de itens de configuração que foram testados e estão prontos para serem evoluídos para a próxima fase do ciclo de vida de desenvolvimento do *software*. Ou seja, a *baseline* controla que todos os produtos de trabalho estejam na versão correta para serem promovidos para a próxima etapa de desenvolvimento. A ideia aqui é evitar problema do tipo “mas na minha máquina funciona, não sei por que no

ambiente de homologação está dando erro", ou ainda "mas eu alterei essa funcionalidade, só que a alteração não está sendo refletida nesta versão que foi liberada para a equipe de testes".

O processo de gerência de configuração deve prever tarefas para a avaliação e a revisão dos itens de configuração, por meio do monitoramento da *baseline* que é selecionada para ser liberada para a próxima fase do ciclo de vida de desenvolvimento.

Segundo o MPS.Br (Softex, 2016), as atividades de gerência de configuração devem compreender: executar uma auditoria funcional da *baseline* gerada, via revisão dos planos, dados, metodologia e resultados dos testes, assegurando que ela cumpra corretamente o que foi especificado; e deve também executar uma auditoria física da *baseline*, com o objetivo de certificar que a *baseline* gerada está completa em relação ao escopo que foi construído e que todos os itens de configuração estão nas versões corretas para serem evoluídos para a próxima etapa do ciclo de vida.

Por fim, os itens de configuração que foram uma *baseline* são submetidos a um processo de liberação (*release*), que representa a notificação formal e distribuição de uma versão aprovada do *software*.

Em determinados momentos do ciclo de vida de desenvolvimento e manutenção do *software*, os itens de configuração podem ainda ser agrupados e verificados, constituindo versões do *software* voltadas para propósitos específicos, dependendo da necessidade do projeto. Nesses casos, cria-se marcos no versionamento dos itens de configuração que são denominados *baselines* ou *releases* (DVMedia, S.d.).

A diferença entre *baselines* e *releases* é bastante sutil, mas precisa estar clara para todos nós.

Dessa forma, uma *baseline* representa um conjunto de itens de configuração formalmente aprovados que servem de base para as etapas seguintes do ciclo de vida do desenvolvimento e manutenção do *software*. Mas, quando uma entrega formal é feita ao cliente, no final de uma iteração ou *sprint*, por exemplo, denominamos esta entrega de *release*.

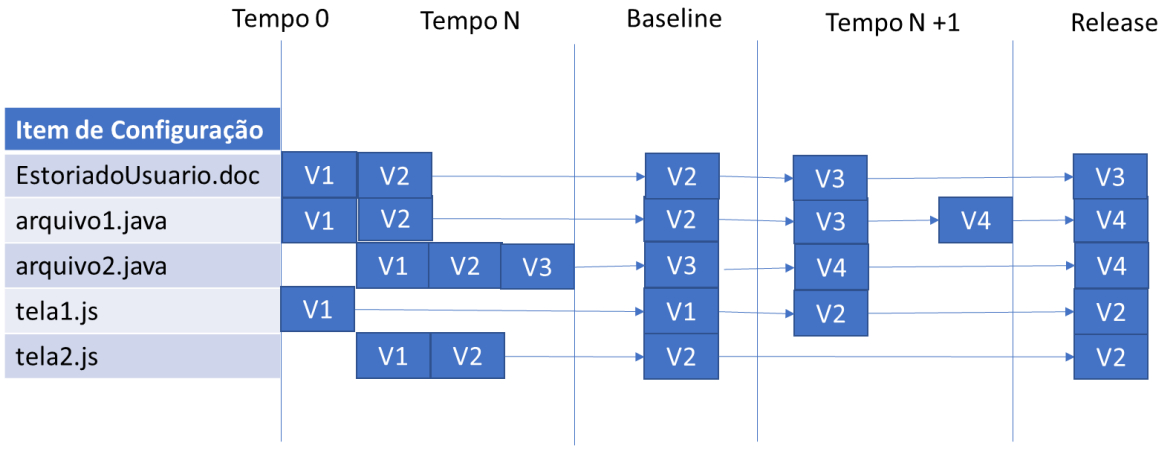
Portanto, a principal diferença é que as *baselines* são utilizadas para controlar as versões fechadas dos itens de configuração para etapas do ciclo de vida que dizem respeito ainda ao envolvimento da equipe de desenvolvimento do projeto, como liberar uma versão, por exemplo, para os testes da equipe de

testes. Enquanto uma *release* é uma *baseline* que passou por todas as etapas que envolvem a equipe de desenvolvimento e está liberada para ser entregue ao cliente, para ser evoluída para o ambiente de homologação ou até mesmo para o ambiente de produção.

Baselines e *releases* são identificadas no repositório único do projeto pelo uso de etiquetas ou *tags,* em inglês, marcando que aquele agrupamento de itens de configuração foi liberado ou para a próxima fase do ciclo de vida ou para o cliente ou usuário final.

Vamos analisar toda essa explicação por meio de um estudo de caso demonstrado na tabela a seguir:

Figura 3 – Quadro de evolução das versões dos itens de configuração em um projeto de *software*



Detalhando o entendimento sobre o quadro acima, podemos entender a evolução dos itens de configuração ao longo do ciclo de vida de desenvolvimento de um *software*.

Os itens de configuração estão representados na lista que contém: EstoriadoUsuario.doc, arquivo1.java, arquivo2.java, tela1.js e tela2.js. Ou seja, 1 arquivo de documentação, contendo as estórias de usuários que serão implementadas; 2 arquivos de código na linguagem Java; e 2 telas construídas em Javascript. Esse conjunto de itens de configuração forma os produtos de trabalho gerados no projeto, todos sob gerência de configuração.

Pois bem, o tempo 0, que é o momento inicial do projeto, todos os itens de configuração estão na versão 1. Conforme o tempo vai passando e alterações são necessárias nos arquivos, novas versões vão sendo geradas. Cada versão corresponde a uma alteração realizadas no arquivo. Quando uma versão está

pronta para ser evoluída para a etapa seguinte do ciclo de vida, é gerada uma *baseline* com a última versão de cada item de configuração. Após ter passado pelas fases que envolvem a equipe de desenvolvimento, ainda em momento de desenvolvimento do *software*, as últimas versões de cada item de configuração poderão ser liberadas para os testes do usuário, logo, essas versões serão empacotadas em uma *release.* A *release* será disponibilizada para o usuário final, no ambiente de homologação ou de produção.

Dessa forma, é possível, de maneira organizada, construir e liberar *software* garantindo qualidade e reduzindo o retrabalho com defeitos indesejados por conta de mal gerenciamento dos itens de configuração de um projeto de *software*.

FINALIZANDO

Nesta etapa, discutimos sobre vários conceitos iniciais, porém de extrema importância para compreender o que é a gerência de configuração em um projeto de *software*.

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

Ter uma ferramenta que automatize o processo de gerência de configuração é essencial para facilitar a vida da equipe de desenvolvimento de *software*. Mas também é essencial ter um processo definido e seguido por todos os integrantes da equipe do projeto. Dessa forma, a cultura de utilizar um repositório único e alimentá-lo continuamente passa a fazer parte do dia a dia da equipe.

Discutimos também sobre o que são itens de configuração, como eles geram uma *baseline* e até mesmo uma *release*. Além de termos compreendido a importância de utilizar a gerência de configuração coo um aliado para a melhoria da qualidade no processo de desenvolvimento de *software*.

O conceito de DevOps surgiu da necessidade de integrar as equipes de desenvolvimento e de operação ou suporte, de forma a garantir que o *software* desenvolvido possa ser facilmente instalado, evoluído e utilizado em ambientes

que tenham a capacidade de suportá-lo, permitindo um uso adequado e estável. A capacidade de um ambiente deve envolver a disponibilização de memória e processamento adequados par a utilização estável e contínua do *software*.

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

REFERÊNCIAS

ANDRADE JUNIOR, J. R. de. **Gerência de configuração**. São Paulo: Pearson, 2014.

DVMEDIA. Gerência de configuração de software. DVMedia, S.d. Disponível em: <https://www.devmedia.com.br/gerencia-de-configuracao-de-software/9145>. Acesso em: 3 maio 2022.

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. São Paulo: Brasport 2019.

PAULA FILHO, W. 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.

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

SOFTEX – Associação para Promoção da Excelência do Software Brasileiro MPS.BR – Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS:2011, **Softex**, jun. 2011**.**

_____. MPS.BR – Guia Geral MPS de Software: 2016. **Softex**, jan. 2016.

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

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