# 1. INTRODUÇÃO A GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE

Vimos que na Engenharia de Software temos quatro atividades básicas para o processo de desenvolvimento de software que são: especificação, implementação, validação e evolução. Na atividade de evolução, o sistema deve evoluir para atender as necessidades mutáveis do cliente, ou seja, os sistemas sempre mudam durante o seu desenvolvimento e uso. Nesta unidade, vamos falar especificamente sobre Gerência de Configuração de Software.

De acordo com Pressman e Maxim (2016, p. 623) as mudanças são "inevitáveis quando o software de computador é construído e podem causar confusão quando os membros da uma equipe de software estão trabalhando em um projeto". E se estas mudanças não forem bem analisadas antes de serem registradas e implementadas, surgem as confusões e informações desencontradas. E durante a criação de um software, mudanças acontecem, e, por isso, para coordenar a confusão e as mudanças, precisamos gerenciá-las eficazmente.

O Gerenciamento de Configuração de Software (SCM - *Software Configuration Management)* ou GCS, é um conjunto de atividades destinadas a gerenciar as mudanças, identificando os artefatos que precisam ser alterados, estabelecendo relações entre eles, definindo mecanismos para gerenciar esses artefatos, controlando as alterações impostas e auditando e relatando todas as mudanças feitas no sistema.

> Gerência de Configuração é um termo amplamente usado, muitas vezes como sinônimo de controle de versão. Definição formal: Gerência de Configuração se refere ao processo pelo qual todos os artefatos relevantes ao seu projeto, e as relações entre eles, são armazenados, recuperados, modificados e identificados de maneira única. Sua estratégia de gerência de configuração determinará como você gerencia todas as mudanças que ocorrem dentro do seu projeto. (HUMBLE e FARLEY, 2014, p. 31):

Durante o desenvolvimento e o uso, os sistemas sempre mudam. Erros aparecem ou são descobertos e precisam ser corrigidos. Mudanças ocorrem nos requisitos do sistema, sejam a pedido do cliente ou imposição de regras de negócio, e com isso, é necessário implementar essas mudanças em uma nova versão do sistema. E ao realizar essas mudanças no software e cria-se uma nova versão de um sistema. Ou seja, o sistema pode ser pensado como um conjunto de versões, sendo que cada uma dessas versões necessita ser mantida e gerenciada.

Essas versões implementam as propostas de mudanças como novas funcionalidades, correções de defeitos e adaptações de novas tecnologias. E em um projeto, podemos ter várias versões em desenvolvimento e em uso ao mesmo tempo. Por isso, é necessário ter procedimentos de gerenciamento de configuração efetivos, para que não seja desperdiçado tempo e esforço em versão errada de um sistema e entregá-la ao cliente errado ou esquecer onde está armazenado o código-fonte do sistema para uma versão específica do sistema ou componente (PRESSMAN e MAXIM, 2016).

Em projetos individuais é fácil uma pessoa esquecer quais as mudanças que realizou e o gerenciamento de configuração é útil para controlar elas. Agora imagine em projetos em equipe, com vários desenvolvedores trabalhando ao mesmo tempo em um sistema ou para desenvolvedores que trabalham com equipes distribuídas, com membros em diferentes locais pelo mundo. Utilizar um sistema de gerenciamento de configuração de software (SCM) é importante para garantir que as equipes tenham acesso a informações sobre o sistema que está em desenvolvimento e que não interfiram no trabalho dos membros da equipe ou de outras equipes.

Como as mudanças podem ocorrer a qualquer momento em um sistema, temos quatro atividades no gerenciamento de configurações de software (figura 1).

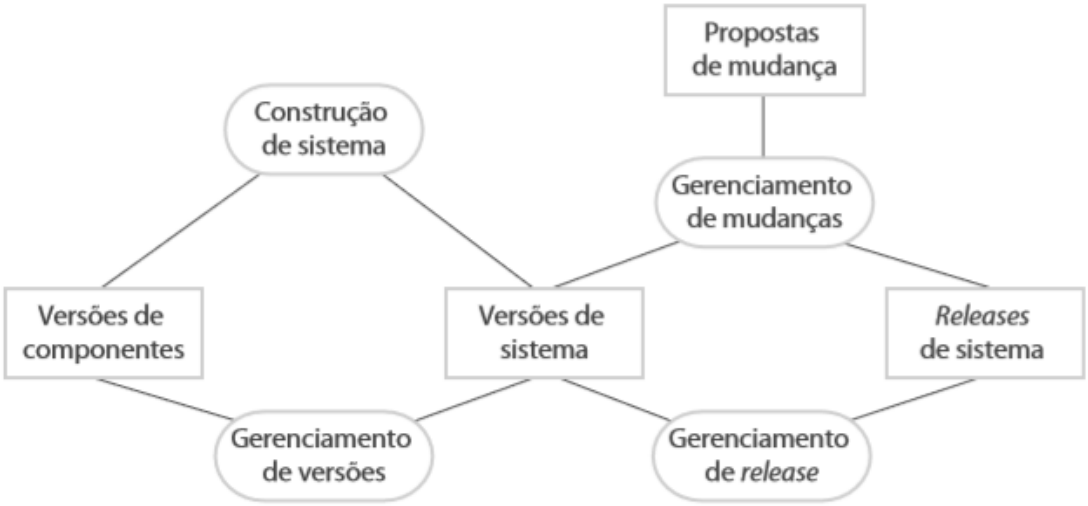


Figura 1 – Atividades de Gerenciamento de Configuração de Software (Sommerville, 2011, p. 476).

Temos que:

**1. Gerenciamento de mudanças**: é a atividade que procura manter o acompanhamento das solicitações de mudanças no sistema (dos clientes e desenvolvedores). Ajuda na definição dos custos, impacto das mudanças e na decisão de quando e se as mudanças devem ser implementadas.

**2. Gerenciamento de versões:** é a atividade que procura manter o acompanhamento das várias versões do sistema e de seus componentes. Assegura que as mudanças possam ser realizadas por diferentes desenvolvedores ao mesmo tempo e que uns não interfiram na tarefa dos outros.

**3. Construção do sistema**: é a atividade onde ocorre o processo de montagem de componentes de sistema, como: dados e bibliotecas. Depois, a compilação e ligação destes, para criar um sistema executável.

**4. Gerenciamento de releases**: é a atividade que envolve a preparação de software para o release (lançamento) externo. Procura manter o acompanhamento das versões de sistema que foram liberadas para uso do cliente.

Vimos que o gerenciamento de configuração é um conjunto de atividades desenvolvidas para controlar as mudanças que podem ocorrer ao longo do ciclo de vida de um sistema. Entretanto, em diferentes empresas, podemos ter os mesmos conceitos, mas com termos diferentes.

Vamos conhecer algumas terminologias usadas no gerenciamento de configuração de software:

| **Item** | **Descrição** |
|---|---|
| Item de Configuração | Qualquer coisa associada ou produzida para um projeto de software (projeto, código, dados de teste, documentos etc) e que tenha sido inserido em um controle de configuração. Em muitos projetos, existem diferentes versões de um item de configuração. Os Itens de configuração devem ter um nome único para serem rastreados. |
| Controle de Configuração | Item onde o processo de versões de sistemas e componentes sejam registradas, controladas e mantidas para que as mudanças sejam gerenciadas e todas as versões de componentes sejam identificadas e armazenadas por todo o ciclo de vida do sistema. |
| Versão | É uma instância de um item de configuração que difere de alguma forma, de outras instâncias desse item. As versões sempre têm um identificador único, o qual é geralmente composto pelo nome do item de configuração + um número de versão. |
| Baseline (linha de base) | Uma baseline é uma coleção de versões de componentes que compõem um sistema. As baselines são controladas, o que significa que as versões dos componentes que constituem o sistema não podem ser alteradas. |

| | |
|---|---|
| Codeline (linhas de desenvolvimento) | Uma codeline é um conjunto de versões de um componente de software e outros itens de configuração dos quais esse componente depende. |
| Mainline (linha principal) | Trata-se de uma sequência de baselines que representam diferentes versões de um sistema |
| Release (liberação ou lançamento) | Uma versão de um sistema que foi liberada para os clientes (ou outros usuários em uma organização) para uso. |
| Espaço de Trabalho | É uma área de trabalho privada em que o software pode ser modificado sem afetar outros desenvolvedores que possam estar usando ou modificando o softw. |
| Branching (ramificação) | Trata-se da criação de uma nova codeline de uma versão em uma codeline existente. A nova codeline e uma codeline existente podem, então, ser desenvolvidas independentemente. |
| Merging (fusão ou mesclagem) | Trata-se da criação de uma nova versão de um componente de software, fundindo versões separadas em diferentes codelines. Essas codelines podem ter sido criadas por um branch anterior de uma das codelines envolvidas. |
| Construção de Sistema | É a criação de uma versão de sistema executável pela compilação e ligação de versões adequadas dos componentes e bibliotecas que compõem o sistema. |

Quadro 1 - Terminologias usadas no gerenciamento de configuração de software (Fonte: SOMMERVILLE, 2011, p. 476).

O gerenciamento de configuração de software (SCM) tem o objetivo de respondes a algumas perguntas, como:

(i) o que foi mudado e quando foi mudado?

(ii) E porque houve as mudanças?

(iii) E quem autorizou e quem fez as mudanças?

(iv) E podemos reproduzir esta mudança?

Resumindo, o gerenciamento de configuração de software (SCM) é uma atividade de apoio para o controle de versão, o controle de mudança e a auditoria das configurações de um sistema.

## 2. GERENCIAMENTO DE MUDANÇAS DE SOFTWARE

Vimos que mudanças podem ocorrer a qualquer momento em um sistema e que para controlar essas mudanças, temos o Gerenciamento de Configuração de Software que é possui quatro atividades que são: o gerenciamento de mudanças, o gerenciamento de versões**,** a construção do sistema e o gerenciamento de releases. Nesta unidade, vamos falar especificamente sobre a atividade gerenciamento de mudanças.

Para Sommerville (2019, p. 709) o gerenciamento de mudanças é a atividade que “envolve manter o acompanhamento das solicitações dos clientes e desenvolvedores por mudanças no software, definir os custos e o impacto de fazer tais mudanças, bem como decidir se e quando as mudanças devem ser implementada”.

Hoje em dia, a mudança é uma realidade para todo tipo de sistema. Ao longo do ciclo de vida útil de um sistema, as necessidades e os requisitos se alteram, erros precisam ser corrigidos e os sistemas necessitam se adaptar às mudanças em seu ambiente ou a novas tecnologias que surgem. E como controlar essas mudanças e garantir que elas sejam aplicadas ao sistema de forma correta? Com um conjunto de processos de gerenciamento de mudanças apoiado por ferramentas. O objetivo do gerenciamento de mudanças é garantir que a evolução do sistema seja um processo controlado e gerenciado e com a

priorização das mudanças mais urgentes e efetivas (SOMMERVILLE, 2019, p. 709).

O conjunto de processos de gerenciamento de mudanças está relacionado com (i) realização de uma análise de custos (ii) analise dos benefícios das mudanças propostas (iii) a aprovação das mudanças e (iv) o acompanhamento do que foi alterado no sistema.

A seguir o modelo de um processo de gerenciamento de mudanças que mostra as principais atividades de gerenciamento (figura 2).

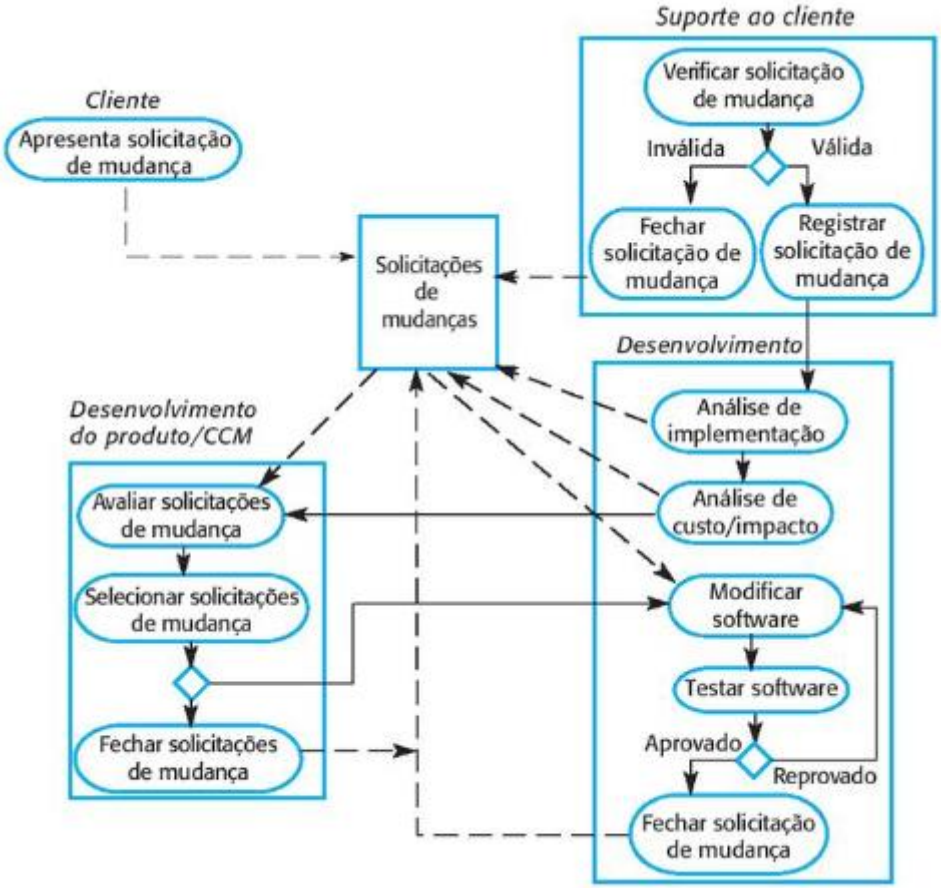


Figura 2 - O processo de gerenciamento de mudanças (Sommerville, 2011, p. 478)

O processo de gerenciamento de mudanças começa quando o cliente apresenta uma solicitação de mudança. Essas solicitações entram na fila de Solicitações de Mudanças, podendo vir do suporte ao cliente, onde é verificada a solicitação de mudanças e verificado se ela é válida ou inválida. Se for válida,

passa ao setor de desenvolvimento, onde serão realizadas a: análise de implementação, análise de custo e impacto, onde será modificado o software e testado.

Para as solicitações de mudança pode ser usado um formulário de solicitação de alteração (CRF, do inglês change request form). Um exemplo de um formulário de solicitação de mudança parcialmente concluído é mostrado no quadro 2.

| **Formulário de solicitação de mudança** | |
|---|---|
| **Projeto:** APP teste | **Número:** 5236 |
| **Solicitante de Mudanças:** Janaina | **Data:** 15/08/22 |
| **Mudança solicitada:** O status dos requerentes (rejeitados, aceitos etc.) deve ser mostrado visualmente na lista de candidatos exibidas na tela. | |
| **Analista de Mudanças:** Pedro | **Data da análise:** 16/08/22 |
| **Componentes afetados:** ApplicantListDisplay, StatusUpdater | |
| **Componentes associados:** StudentDatabase | |
| **Avaliação de Mudança:** Relativamente simples de implementar, alterando a cor de exibição de acordo com status. Uma tabela deve ser adicionada para relacionar status a cores. Não é requerida alteração nos componentes associados. | |
| **Prioridade da Mudança:** Média | |
| **Implementação da Mudança** | |
| **Esforço estimado:** 2 horas | |
| **Decisão**: Aceitar alterar. Mudança deve ser implementada no Release 1.2 | |
| **Data de teste:** 22/08/22 | |
| **Comentários** | |

Quadro 2 - Um formulário de solicitação de mudança parcialmente concluído (Fonte: Sommerville, 2019, p.711).

Esse é um exemplo de um tipo de formulário de mudanças que pode ser usado em um sistema. O desenvolvedor de sistema responsável decide como implementar essa mudança e estima o tempo necessário para a alteração.

Sommerville (2019) descreve alguns fatores importantes que devem ser levados em consideração ao se decidir pela aprovação ou não de uma mudança solicitada. Os fatores são:

1. **As consequências de não fazer a mudança**: deve ser considerado o que acontecerá se a mudança solicitada não for implementada. Se a mudança for associada a um erro/falha do sistema relatada pelo cliente, a gravidade do erro/falha precisa ser levada em consideração.
2. **Os benefícios da mudança:** deve ser analisado se a mudança vai beneficiar muitos usuários do sistema ou é uma mudança especifica.
3. **O número de usuários afetados pela alteração:** deve ser analisado número de usuários que serão afetados pela mudança. Se apenas alguns usuários forem afetados, a mudança pode receber prioridade baixa. Se forem muitos usuários afetados, a mudança pode ser desaconselhada caso ela possa a ter efeitos adversos no sistema.
4. **Os custos de se fazer a mudança**: deve ser analisado se a mudança afetar muitos componentes de sistema aumentando as chances de introdução de novos erros e/ou levar muito tempo para ser implementada, então ela pode ser rejeitada, dados os elevados custos envolvidos.
5. **O ciclo de release de produto**. Deve ser analisada se outra mudança acaba de ser liberada em uma nova versão do software para os clientes, deve-se atrasar a implementação da mudança solicitada até o próximo release planejado

Se a equipe de desenvolvimento mudar os componentes de software, o ideal é manter um registro das mudanças feitas para cada componente.

Resumindo, o gerenciamento de mudanças ajuda a mapear, para cada mudança efetuada no sistema, qual foi o motivo que gerou esta mudança. É uma atividade importante do gerenciamento de configuração de software, pois é a atividade que permite saber os motivos das mudanças e de uma configuração ter sido mudada para outra configuração ou outra versão do sistema.

## 3. GERENCIAMENTO DE VERSÕES

Anteriormente, vimos que mudanças podem ocorrer a qualquer momento em um sistema e que para controlar essas mudanças, temos o Gerenciamento de Configuração de Software que é possui quatro atividades que são: o gerenciamento de mudanças, o gerenciamento de versões**,** a construção do sistema e o gerenciamento de releases. Nesta unidade, vamos falar especificamente sobre a atividade gerenciamento de versões.

O gerenciamento de versão é o processo de manter o controle de diferentes versões de componentes de software ou itens de configuração e dos sistemas em que esses componentes e itens são utilizados. Ele também fornece a garantia de que as mudanças feitas por diferentes desenvolvedores nas versões, não interfiram umas nas outras.

Os sistemas de controle de versão procuram identificar, armazenar e controlar todo o acesso as diferentes versões dos componentes. Existem dois tipos de sistema de controle de versão:

| Tipo de sistema | Descrição |
|---|---|
| Sistemas centralizados | O controle é feito em um único repositório mestre que mantém todas as versões dos componentes de software que estão sendo desenvolvidos. Exemplo: Subversion |
| Sistemas distribuídos | O controle é feito com várias versões de repositório de componentes ao mesmo tempo. Exemplo: Git. |

Quadro 3 - Tipos de sistema de controle de versão (Fonte: SOMMERVILLE, 2019, p. 698).

Vamos pensar em gerenciamento de versões como o processo de gerenciamento das linhas de desenvolvimento (codelines) e das linha de base (baselines).

> Essencialmente, uma codeline é uma sequência de versões de código-fonte com versões posteriores na sequência derivadas de versões anteriores. Normalmente, as codelines são aplicadas para componentes de sistemas, havendo, assim, diferentes versões de cada componente. Uma baseline é uma definição de um sistema específico. A baseline, portanto, especifica a versão de cada componente incluída no sistema, mais uma especificação das bibliotecas usadas, arquivos de configuração etc. (SOMMERVILLE, 2019, p. 697).

A seguir, podemos ver que diferentes baselines usam versões diferentes dos componentes de cada codeline (figura 3).

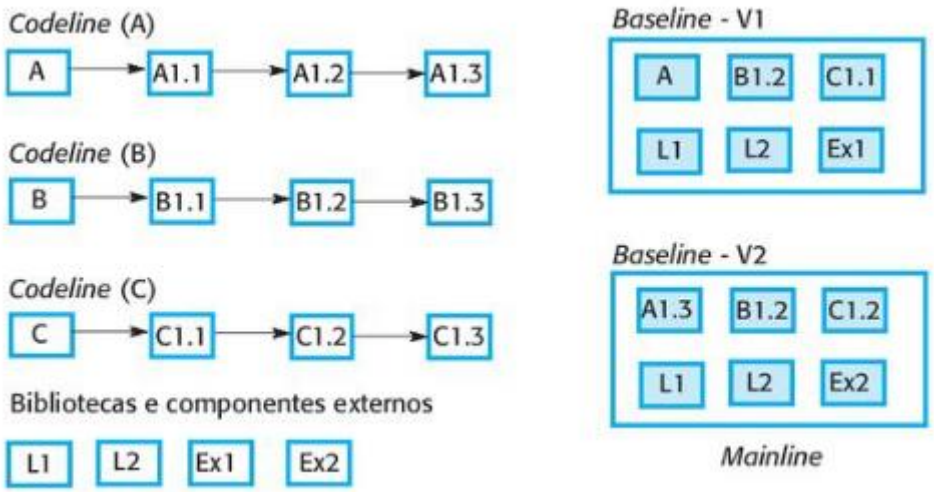


Figura 3 – Diferença entre as Codelines e baselines (Fonte: Sommerville, 2019, p. 698).

No lado direito, temos caixas sombreadas que representam componentes na definição de baseline e que indicam referências a componentes de uma codeline. As baseline formam uma mainline, que é uma sequência de versões de sistema. Para Sommerville (2019, p. 698) as baselines podem ser "especificadas usando-se uma linguagem de configurações que lhe permite definir quais componentes estão incluídos em uma versão de um determinado sistema. É possível especificar explicitamente a versão de um componente

específico (X.1.2, por exemplo) ou, simplesmente, especificar o identificador de componente (X) ".

Alguns recursos são fornecidos pelos sistemas de gerenciamento de versões, como:

| **Recurso** | **Descrição** |
|---|---|
| Identificação de Versão e Release | Recurso onde as versões gerenciadas recebem identificadores e eles se baseiam no nome do item de configuração seguido por um ou mais números. |
| Gerenciamento de Armazenamento | Fornece recursos de armazenamento para reduzir os espaços de armazenamento requeridos pelas várias versões de componentes. |
| Registro de Histórico de Alterações | Recurso onde todas as mudanças feitas no sistema ou em componentes são registradas e listadas, para que em seguida sejam selecionados os componentes que serão incluídos na baseline. |
| Desenvolvimento independente | Recurso onde diferentes desenvolvedores podem trabalhar ao mesmo tempo no mesmo componente. O gerenciamento de versões garante que as mudanças feitas nos componentes por desenvolvedores diferentes não interfiram. |
| Suporte a projetos: | Recurso onde se tem vários projetos que compartilham componentes e com o suporte a projetos é possível fazer check-in e check-out dos arquivos dos projetos. |

Quadro 4 - Recursos fornecidos pelos sistemas de gerenciamento de versões (Fonte: SOMMERVILLE, 2011, p. 482) adaptado.

O recurso gerenciamento de armazenamento é uma das funções mais importante dos sistemas de gerenciamento de versões. Ele ajuda a reduzir o

espaço requerido em disco para manter todas as versões do sistema. Quando uma nova versão é criada, ele armazena um delta (uma lista de diferenças) entre a nova versão e a mais antiga, usada para criar essa nova versão.

O recurso desenvolvimento independente também é uma função importante, pois a maior parte do processo de desenvolvimento de software é uma atividade de equipe e por isso, ocorrem situações em que os membros da equipe trabalham no mesmo componente e ao mesmo tempo. Este recurso ajuda diferentes desenvolvedores a trabalhar ao mesmo tempo no mesmo componente. Por exemplo: temos o desenvolvedor Paulo que está fazendo mudanças no sistema nos componentes A, B e C. Ao mesmo tempo, o desenvolvedor André está trabalhando em mudanças e estas requerem mudanças são nos componentes X, Y e C. Portanto, Paulo e André, estão mudando o componente C ao mesmo tempo. Por isso é importante usar o recurso desenvolvimento independente para evitar que tais mudanças interferiram umas nas outras. Este recurso usa o conceito de repositório público e espaço de trabalho privado. Sommerville (2019, p. 483) explique que os desenvolvedores “realizam check-out de componentes de um repositório público em seu espaço de trabalho privado e podem mudá-los como quiserem em seu espaço de trabalho privado. Quando as mudanças forem concluídas, eles realizam o check-in de componentes para o repositório”.

Quando o componente é modificado, ele é registrado e é atribuído um identificador de versão diferente para as diferentes versões, as quais são armazenadas separadamente. Muitas vezes, existem sobreposições entre as mudanças feitas, e elas interferem umas nas outras. Um desenvolvedor deve verificar sempre se existem conflitos e modificar as mudanças para que estas sejam compatíveis.

O gerenciamento de versões possui algumas vantagens, como:

- **Versionamento**: em alguns casos, pode ser necessário voltar a uma versão de um determinado arquivo por algum erro cometido ou por mudança de escopo, e com o versionamento é possível fazê-lo de forma simples.

- **Segurança**: permite que apenas usuários autorizados acessem os códigos fontes do sistema.
- **Rastreabilidade**: permite saber todas as informações dos componentes, como: quem mudou, quando foi mudado, como e porque ocorreu as mudanças.
- **Confiança** e **Colaboração**: importante para as empresas que tem muitas equipes trabalhando nos componentes ao mesmo tempo.

Resumindo, o gerenciamento de versão é muito importante para o processo de manter o controle de diferentes versões de componentes de sistemas. Ele fornece a garantia de que as mudanças feitas por diferentes desenvolvedores nas versões, não interfiram umas nas outras.

## 4. FERRAMENTAS PARA AUXILIAR NA GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE.

Vimos uma introdução para entender a gerência de configuração de software. Seguimos para conhecer as atividades de gerenciamento de configuração que são: o gerenciamento de mudanças e o gerenciamento de versões. Para finalizar, vamos apresentar as ferramentas para auxiliar na gerência de configuração de Software.

A gerência de configuração de software envolve o controle de grande quantidade de informação e garantia de que mudanças no sistema sejam registradas e controladas. E com essa necessidade de controlar estas informações e modificações, surgiu muitos métodos e ferramentas, com o intuito de maximizar a produtividade e minimizar os erros cometidos durante as mudanças. A seguir exemplos de ferramentas especializadas que existem no mercado para o gerenciamento de configuração de software:

| **Ferramentas** | **Descrição** |
|---|---|
| Concurrent Versions System (CVS) | Ferramenta usada para o controle de versão que permite que se trabalhe com diversas versões de |

| | |
|---|---|
| | arquivos organizados em um diretório e localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando manipulou os arquivos. |
| Subversion (SVN) | Ferramenta usada para o controle de versão open-source que gerencia arquivos e diretórios controlando as alterações realizadas, ajuda na recuperação de versões anteriores e a visualizar o histórico de alterações. |
| Git | Ferramenta para o controle de versão distribuído gratuito e com código fonte aberto. Fácil de usar e de aprender e com desempenho extremamente rápido. |
| Bitbucket | Ferramenta para hospedagem e colaboração e compartilhamento de dados, criação e implantação de códigos e automatização de testes. Usa conceito de cloud computing e infraestruturas on premises. |
| Bugzilla | Ferramenta usada para ajudar a gerenciar o desenvolvimento de software, com estrutura de banco de dados otimizada para maior desempenho e escalabilidade, possui excelente segurança para proteger a confidencialidade e para consulta avançada. |
| GitHub | Ferramenta que é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão que usa o Git. Permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo |

| | |
|---|---|
| Jira | Ferramenta que permite o monitoramento de tarefas e acompanhamento de projetos garantindo o gerenciamento de todas as suas atividades em único lugar. |
| Phabricator | Ferramenta que é uma suite colaborativa de código aberto para desenvolvimento de software e aplicações web de código aberto. |
| Redmine | Ferramenta usada como gerenciador de projetos baseados na web e ferramenta de gerenciamento de bugs. Contém calendário e gráficos de Gantt para ajudar na representação visual dos projetos e seus deadlines. |
| Trac | Ferramenta open source e de interface web para controle de mudanças em projetos de desenvolvimento de software. Ajudar o desenvolvedor a rastrear as mudanças. |
| Mercurial | Ferramenta multiplataforma de controle de versão distribuído para desenvolvedores de software implementado principalmente em Python, porém o utilitário binário diff foi escrito em C. |
| Buildbot | Ferramenta de integração contínua de desenvolvimento de software que automatiza o ciclo de compilação ou teste necessário para validar as mudanças na base de código do projeto. |
| CircleCI | Ferramenta que é uma plataforma de entrega e integração contínuas, que pode ser usada para implementar práticas de DevOps. |

| | |
|---|---|
| **Code Climate** | É uma ferramenta de análise estática de qualidade do seu código, varredura do código procurando por problemas de duplicação, code smells e outros problemas variados. |
| **Codeship** | Ferramenta de serviço de entrega contínua hospedado que se concentra na velocidade, confiabilidade e simplicidade. Pode ser configurado para criar e implantar aplicativos do GitHub para o cenário ou a plataforma de produção de sua escolha. |
| **Drone.io** | Ferramenta open source, desenvolvida em Go e baseada fortemente no uso de containers Docker. |
| **Jenkins** | Ferramenta que é um servidor de automação de código aberto, que ajuda a automatizar as partes do desenvolvimento de software relacionadas à construção, teste e implantação, facilitando a integração e a entrega contínuas. |
| **Travis CI** | Ferramenta que é um serviço de integração contínua hospedado usado para construir e testar projetos de software hospedados no GitHub e Bitbucket. |

Quadro 5 - Ferramentas especializadas para o gerenciamento de configuração de software (Fonte: autora)

São várias ferramentas disponíveis no mercado para auxiliar no gerenciamento de configuração de software. A maioria das ferramentas são programas ou serviços usados por engenheiros de software ou desenvolvedores para auxiliar nas atividades gerenciamento de mudanças e o gerenciamento de versões.