Gerência de Configuração de Software

Os sistemas de software estão em constante evolução. A manutenção do software, isto é, modificações em artefatos existentes, chega a consumir 75% do custo total do seu ciclo de vida. Aproximadamente, 20% de todo o esforço de manutenção é usado para consertar erros de implementação e os outros 80% são utilizados na adaptação do software em função de modificações em requisitos funcionais, regras de negócios e na reengenharia da aplicação.

A Gerência de Configuração de Software surgiu da necessidade de controlar estas modificações, por meio de métodos e ferramentas, com o intuito de maximizar a produtividade e minimizar os erros cometidos durante a evolução.

É uma disciplina que controla e notifica as inúmeras correções, extensões e adaptações aplicadas durante o ciclo de vida do software de forma a assegurar um processo de desenvolvimento e evolução sistemático e rastreável, sendo indispensável quando equipes manipulam, muitas vezes em conjunto, artefatos comuns.

Apesar de existir um forte apelo para o uso da Gerência de Configuração de Software durante a etapa de manutenção, a sua aplicação não se restringe somente a essa etapa do ciclo de vida do software.

O uso dos sistemas de Gerência de Configuração é fundamental para prover controle sobre os artefatos produzidos e modificados por diferentes recursos desde o planejamento e levantamento de requisitos até a construção e entrega do produto. O motivo da sua importância está geralmente associado aos problemas identificados quando a Gerência de Configuração não é utilizada no desenvolvimento de software. Abaixo, vamos analisar alguns destes problemas.

Imagine que uma organização desconhece o que seja Gerência de Configuração e que em um determinado projeto um desenvolvedor esteja modificando os artefatos C1, C2 e C3 em um diretório compartilhado na rede. Simultaneamente, um segundo desenvolvedor modifica os artefatos C4, C5 e também o artefato C3, como exemplifica a Figura 1.

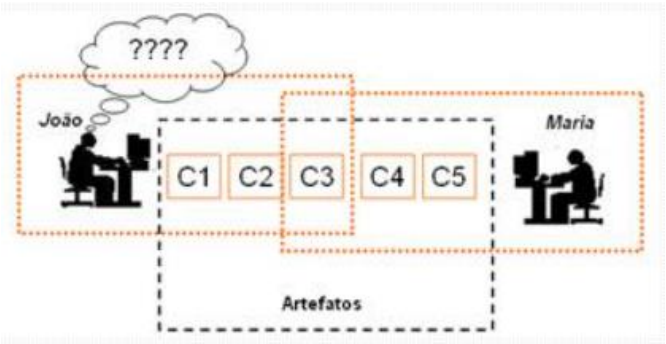

Figura 1. Espaço de Trabalho compartilhado por vários desenvolvedores

Neste cenário, o segundo desenvolvedor não notifica o primeiro desenvolvedor sobre o impacto que a modificação do artefato C3 pode causar no código. Consequentemente, o primeiro desenvolvedor, que está usando o mesmo espaço de trabalho, não conseguirá identificar, de forma rápida, o motivo que levou sua implementação a falhar. Este problema acontece pela falta de notificação e pelo compartilhamento de artefatos de software por diversos desenvolvedores.

Imagine que agora foi acordado entre os desenvolvedores que o ideal seria centralizar os artefatos em um repositório e que cada desenvolvedor implementaria suas modificações em um espaço de trabalho privado. Após cada modificação, o artefato seria devolvido ao repositório. Considerando este cenário, frequentemente ocorreriam sobreposições ou perdas de modificações implementadas nos artefatos comuns nas organizações sem a prática da Gerência de Configuração.

Um desenvolvedor poderia implementar sua modificação em uma versão desatualizada do artefato e sobrepor a versão mais atual disponibilizada por outro. Este problema ocorre devido à atualização simultânea, quando dois desenvolvedores compartilham o mesmo repositório e não existe controle ou restrição quanto ao acesso a este repositório (ver Figura 2).

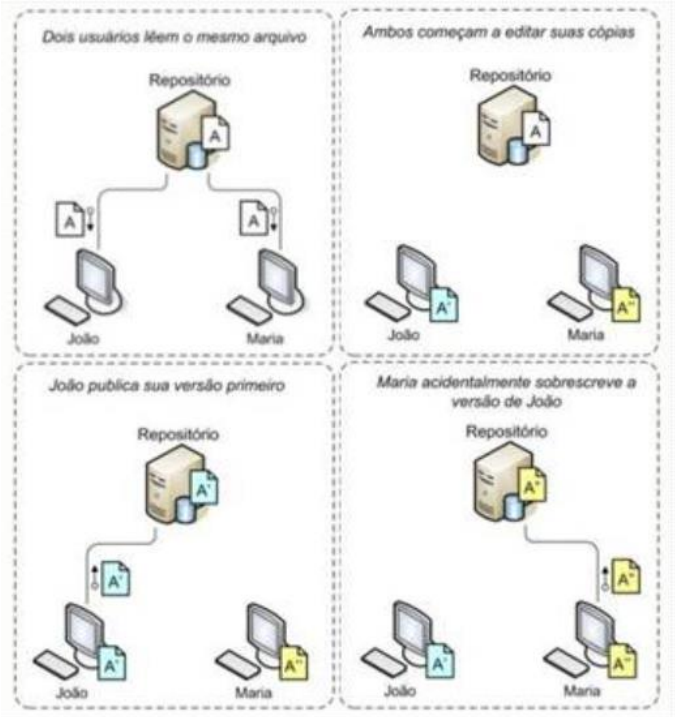

Figura 2. Repositório centralizado compartilhado por vários desenvolvedores

Sob a perspectiva de desenvolvimento, a Gerência de Configuração de Software abrange três sistemas principais: controle de modificações, controle de versões e controle de gerenciamento de construção.

O sistema de controle de versões permite que os artefatos sob Gerência de Configuração evoluam de forma distribuída, concorrente e disciplinada, evitando perdas ou sobreposições durante o desenvolvimento e a manutenção do artefato. Podemos citar como exemplos de ferramentas de mercado: CVS, Subversion, IBM Rational ClearCase e Microsoft Visual Source Safe.

O sistema de controle de modificações armazena todas as informações geradas durante o andamento das solicitações de modificação e relata essas informações aos participantes interessados e autorizados. Podemos citar como exemplos de ferramentas de mercado: Bugzilla, Jira, Trac e IBM Rational ClearQuest.

O sistema de gerenciamento de construção automatiza o processo de transformação dos diversos artefatos do software que compõem um projeto em um sistema executável propriamente dito. Este processo é nomeado construção do software que, por exemplo, testa e empacota a aplicação java como um arquivo jar. Este processo ocorre de forma aderente às normas, procedimentos, políticas e padrões definidos para o projeto. Podemos citar como exemplos de ferramentas de mercado: Maven e Apache Ant.

As vantagens da utilização da Gerência de Configuração de Software são inúmeras. Dentre elas, podemos listar:

ganho de produtividade e eficiência;

diminuição do retrabalho e dos erros;

aumento da disciplina no processo de desenvolvimento;

aumento da memória organizacional;

acesso às informações qualitativas e quantitativas referentes ao processo de desenvolvimento, como por exemplo, medida de esforço para efetuar uma alteração e frequência de modificações por componente;

possibilidade de estabelecer uma trilha de auditoria indicando por que, quando e por quem um artefato foi alterado;

auxílio à gerência de projetos e

garantia de ambiente estável no qual o produto deve ser desenvolvido.

Neste artigo, veremos alguns conceitos da área de Gerência de Configuração de Software e como esta área se relaciona com o processo de desenvolvimento de software. Também serão apresentadas algumas estratégias de organização do trabalho.

Terminologia

O sistema de controle de versões permite que os artefatos sejam obtidos, por meio de uma operação conhecida como check-out, modificados dentro do espaço de trabalho do desenvolvedor e, depois, retornados ao repositório, por meio de uma operação conhecida como check-in, como exemplifica a Figura 3. O repositório é o local de armazenamento dos artefatos que estão sob controle da Gerência de Configuração de Software. Estes artefatos recebem o nome de itens de configuração. A cada operação de check-in realizada, a versão do item de configuração é incrementada de uma unidade. Quando o item é adicionado pela primeira vez no repositório, este item passa a ter a versão igual a 1. Para cada item de configuração armazenado, são anexadas informações como: datas da criação ou alteração, comentários e versões.

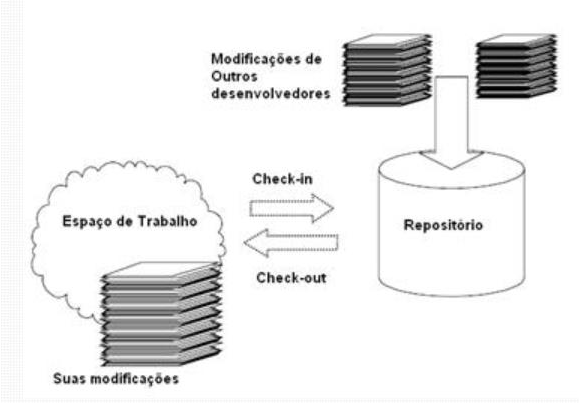

Figura 3. Operações Check-in e Check-out

Neste cenário, não há perdas ou sobreposições porque políticas de trabalho foram estabelecidas, restringindo ou controlando as modificações no repositório. As ferramentas de controle de versões normalmente suportam a definição de diferentes políticas de trabalho. Dentre essas políticas, podemos citar a política pessimista, que enfatiza o uso de check-out reservado, fazendo bloqueio e inibindo o paralelismo do desenvolvimento sobre o mesmo artefato. O uso desta política pode ser exemplificado na Figura 4.

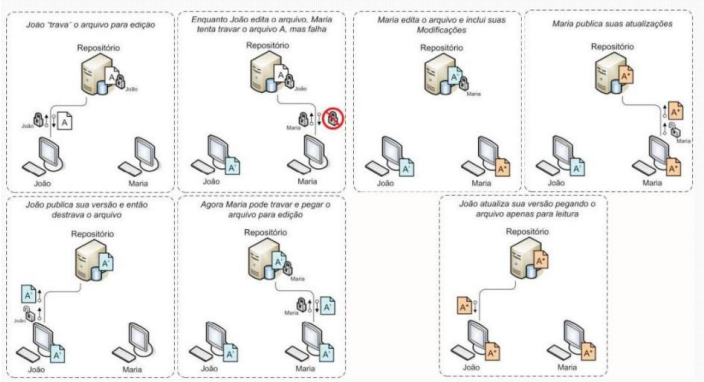

Figura 4. Política Pessimista

Outra política amplamente utilizada é a otimista. Neste cenário, se um artefato for alterado simultaneamente por dois desenvolvedores, a política assume que a quantidade de conflitos será naturalmente baixa e que será mais fácil tratar cada conflito individualmente, caso eles venham a ocorrer. A política otimista usa um mecanismo conhecido como junção (merge), que une as modificações efetuadas em paralelo sobre um mesmo artefato e produz uma nova versão deste artefato que contém a soma das modificações. Os conflitos ocorrem quando a mesma região ou linha do arquivo é modificada. A junção é automática, na maioria dos casos, mas quando ocorre um conflito, ela deve ser feita de forma manual. A Figura 5 ilustra o processo.

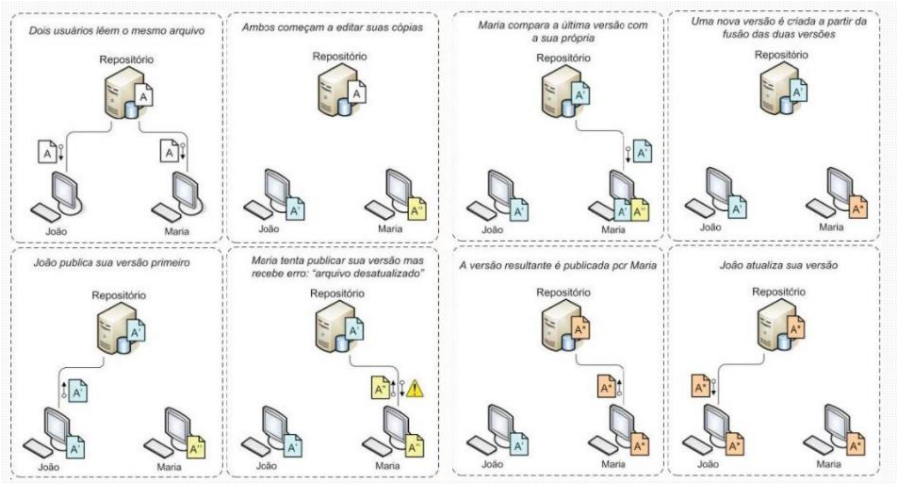

Figura 5. Política Otimista

Existem situações onde uma determinada política é mais indicada do que as demais. Nos casos onde a junção tende a ser complexa, quando, por exemplo, os artefatos não são textuais e a ferramenta não dá apoio automatizado para junções, é mais indicado trabalhar usando políticas pessimistas. Contudo, na grande maioria das situações referentes ao desenvolvimento de software, as políticas otimistas atendem satisfatoriamente.

Além desses recursos fundamentais presentes no sistema de controle de versão, outros recursos mais elaborados também são encontrados com frequência. Em determinados momentos do ciclo de vida de desenvolvimento e manutenção do software, os itens de configuração são agrupados e verificados, constituindo configurações do software voltadas para propósitos específicos. Neste momento, cria-se marcos no versionamento de artefatos que são denominados baselines ou releases.

A diferença entre baselines e releases é sutil. As baselines representam conjuntos de itens de configuração formalmente aprovados que servem de base para as etapas seguintes de desenvolvimento. Mas, quando uma entrega formal é feita ao cliente, no final de uma iteração, por exemplo, denominamos esta entrega de release. Baselines e releases são identificadas no repositório, na grande maioria das vezes, pelo uso de etiquetas (tags). A Figura 6 representa a baseline criada ao final da fase de codificação, considerando quatro itens de configuração. Cada item de configuração apresenta seu respectivo histórico de versões no repositório.

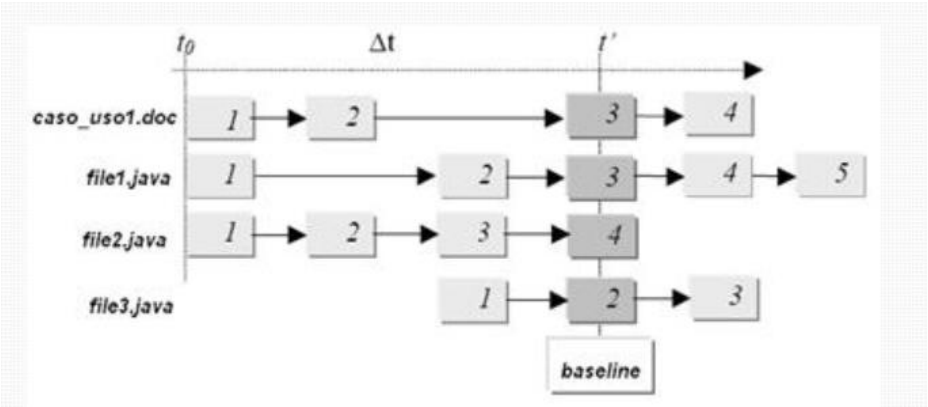

Figura 6. Definição de Baselines

A Gerência de Configuração de Software também permite que a implementação de novas funcionalidades por uma equipe seja realizada em paralelo, mas de forma isolada e independente das modificações de outros desenvolvedores. O isolamento é obtido com o uso de ramo (branch). As linhas de desenvolvimento (codelines) são designadas para cada projeto e são compartilhadas por vários desenvolvedores. A primeira linha de desenvolvimento definida no projeto é, por convenção, nomeada mainline. O ramo é criado no repositório e representa uma linha secundária de desenvolvimento que pode ser unida novamente à linha principal (mainline) por meio da operação de junção (merge). Atualmente, a necessidade de atender, ao mesmo tempo, as múltiplas demandas do projeto torna o uso de ramos um diferencial (ver Figura 7).

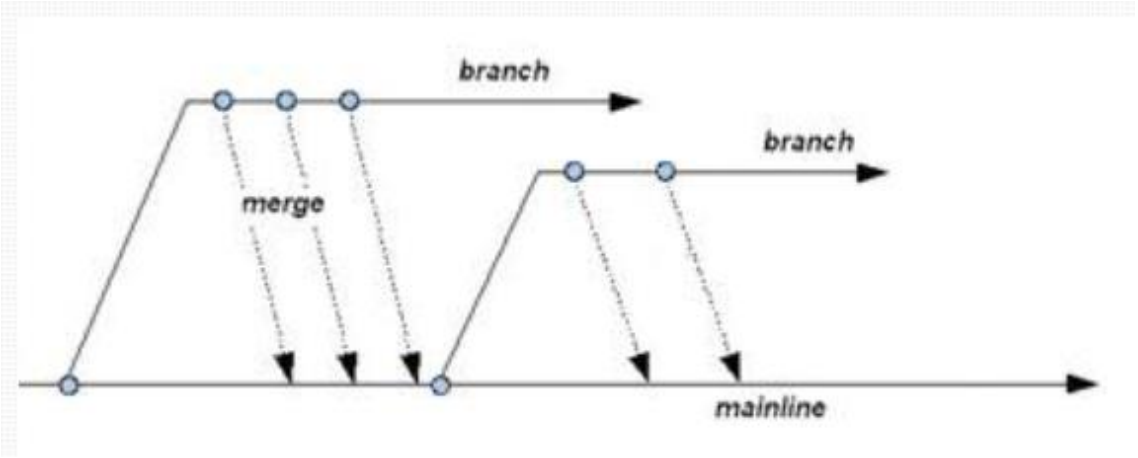

Figura 7. Definição de ramos

A Figura 8 exemplifica a criação de um ramo e a operação de junção. A junção é efetuada para cada artefato do ramo e todas as modificações efetuadas desde o ancestral em comum são levadas em consideração. Imagine que um artefato foi modificado na mainline e no ramo. Digamos que, quando o ramo foi criado, este artefato estava na mainline com a versão 1. Mas quando ocorreu a junção do ramo, ele estava na mainline com a versão 2. O processo de junção soma as modificações efetuadas em paralelo, criando uma nova versão do artefato. Portanto, ao final, a versão 3 do artefato contém as modificações ß efetuadas na mainline desde a versão 1 deste artefato e também as modificações a realizadas no ramo.

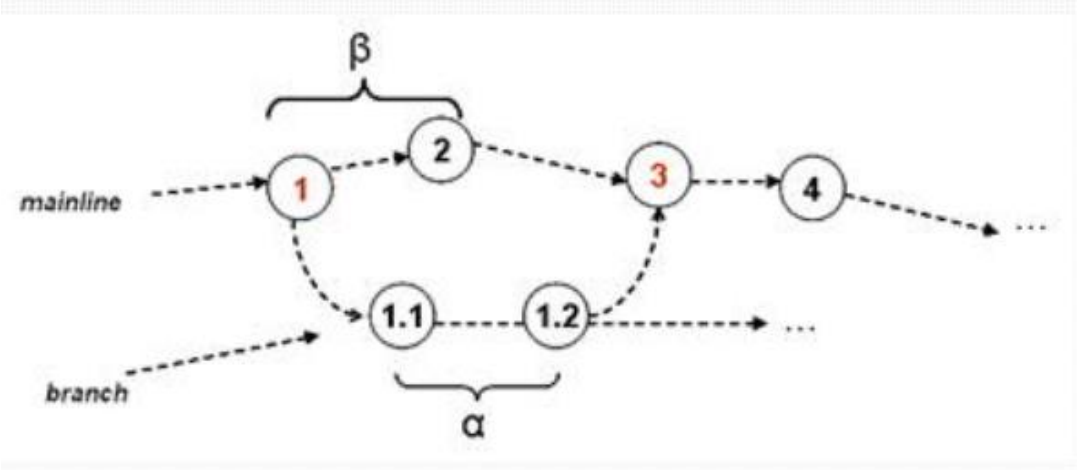

Figura 8. Junção

É importante ressaltar que o isolamento longo dificulta a operação de junção, portanto, é necessário realizar junções periodicamente.

Estratégias de Organização

Estratégias podem ser adotadas para organizar o trabalho dos desenvolvedores no projeto.

De forma genérica, criar um ramo é um meio de organizar o trabalho, pois isola o desenvolvimento de outras modificações e possibilita que o trabalho seja executado sem que uma modificação específica cause impacto nas demais alterações do software.

No entanto, o isolamento que o ramo promove, tem um custo associado. Mesmo com ótimas ferramentas, a junção pode ser uma atividade difícil, em função dos conflitos que ocorrem. Portanto, deve existir um balanceamento entre custo versus benefício, avaliando duas alternativas:

usar isolamento com ramos e conciliar os eventuais conflitos que podem surgir ao fazer a junção deste ramo com a mainline; ou

não usar isolamento e fazer todas as modificações na mainline. Esta última alternativa pode minimizar os problemas de integração, mas maximizar a concorrência e dificultar a entrega de uma release com um subconjunto de requisitos completo e correto em um marco do projeto.

Estratégias de ramificação podem ser utilizadas, ao longo do desenvolvimento, em situações específicas. Algumas podem ser combinadas, outras são mutuamente exclusivas. Dentre as estratégias de organização, vamos citar neste artigo:

[Manutenção caótica] Estratégia onde não existe isolamento e, portanto, a impossibilidade de separar o que é manutenção corretiva da evolutiva (ver Figura 9). Na mainline ocorre a evolução e a correção do software. Não existem ramificações.

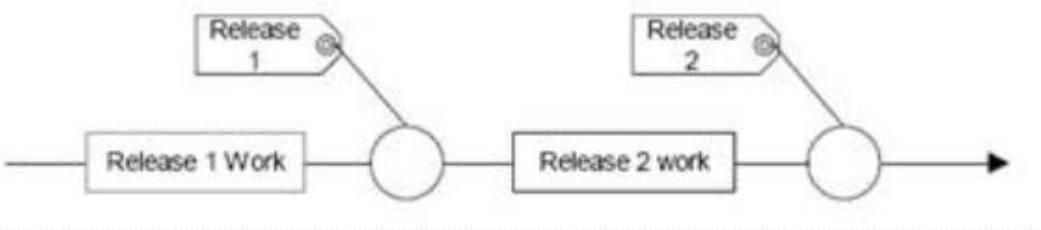

Figura 9. Manutenção caótica

Gerência de Configuração e Desenvolvimento de Software

Por ser uma área fortemente calcada em controle, a Gerência de Configuração é referenciada em diversas normas, processos, procedimentos, políticas e padrões, como ISO 12207, CMMI e MPS.Br.

Do ponto de vista gerencial, o processo de Gerência de Configuração de Software é dividido em cinco funções (ver Nota 1): identificação da configuração, controle da configuração, acompanhamento da situação da configuração, auditoria da configuração e gerenciamento de entrega.

Nota 1:

A função de identificação da configuração tem por objetivo:

a seleção de quais artefatos serão itens de configuração;

a definição de uma nomenclatura, que possibilite a identificação inequívoca dos itens de configuração, baselines e releases; e

a descrição dos itens, tanto física quanto funcionalmente.

A seleção de itens de configuração é feita no início da fase de planejamento e leva em conta:

se o artefato é crítico para o projeto;

a dependência entre artefatos;

o impacto que uma modificação do item tem no produto;

se o artefato pode ser modificado por dois ou mais grupos;

se é frequentemente alterado devido a sua complexidade e

se é gerado manualmente, automaticamente ou ambos.

A função de controle da configuração é designada para controlar e acompanhar a evolução dos itens de configuração selecionados na função de identificação. Ferramentas como JIRA, Bugzilla, dentre outras, apoiam, em conjunto com as ferramentas de controle de versões, as atividades desta função.

A função de acompanhamento da situação da configuração visa:

armazenar as informações geradas pelas demais funções; e

permitir que essas informações possam ser acessadas em função de necessidades específicas, por exemplo, para a melhoria do processo, para a estimativa de custos futuros e para a geração de relatórios gerenciais.

Estas informações podem ser obtidas, no decorrer do projeto, a partir dos sistemas de controle de versões e modificações.

A função de auditoria da configuração ocorre geralmente quando uma release deve ser criada. Suas atividades compreendem:

auditoria funcional, que abrange a revisão dos planos, dados, metodologia e resultados dos testes, assegurando que a release cumpre corretamente o que foi especificado; e

auditoria física, com o objetivo de certificar que a release é completa em relação ao que foi acertado em cláusulas contratuais. A auditoria pode ser feita com base nos relatórios obtidos na função anterior.

Já a função de gerenciamento de liberação e entrega descreve o processo formal de:

construção e liberação de uma release do produto; e

entrega, com informações de como implantar o software no ambiente final de execução. Ferramentas, como Ant, permitem que roteiros de construção sejam escritos e executados no apoio a esta fase.

Para auxiliar e garantir a execução das atividades relacionadas com as funções da Gerência de Configuração de Software, uma organização pode definir uma equipe de Gerência de Configuração, normalmente única no contexto organizacional.

A Gerência de Configuração é um processo auxiliar de controle e acompanhamento das atividades do desenvolvimento de software (ler Nota 2).

Para exemplificar, vamos considerar um ciclo de vida iterativo e incremental que propõe inicialmente a identificação do escopo do projeto e a aprovação deste escopo pelo cliente. Posteriormente, nesta abordagem, o software é construído em ciclos sucessivos denominados iterações.

A cada iteração, os requisitos são priorizados, detalhados, aprovados e o software é modelado, construído e testado, como ilustra a Figura 11. Ao final, uma release do produto é entregue para homologação e aprovação do cliente.

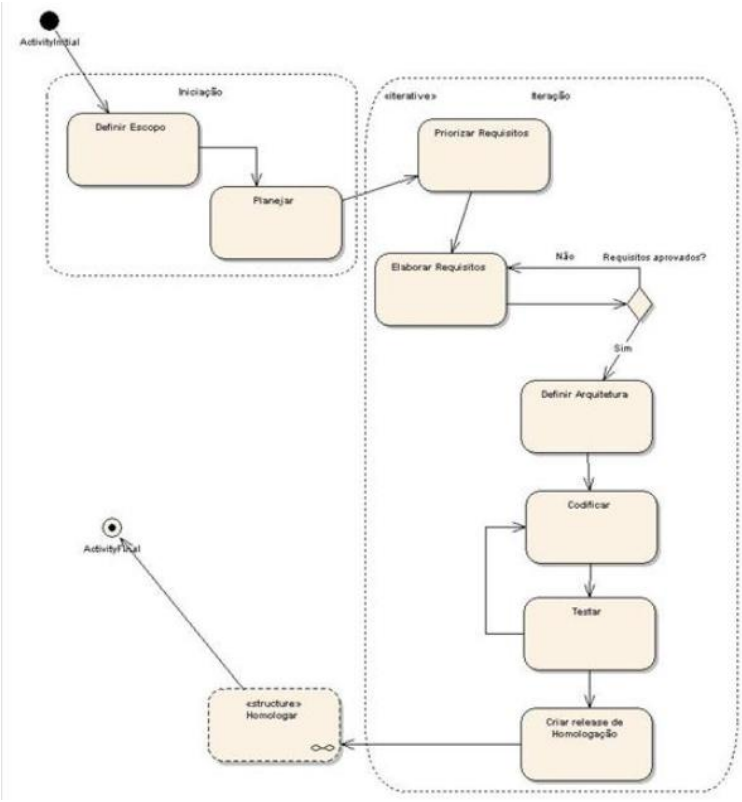

Figura 11. Processo de Desenvolvimento

Na atividade homologar é importante verificar a release em um ambiente isolado do desenvolvimento, realizando testes funcionais. Neste momento, à medida que erros são identificados no teste, solicitações de modificações são registradas na atividade modificar liberação. A Figura 12 ilustra a fase de homologação.

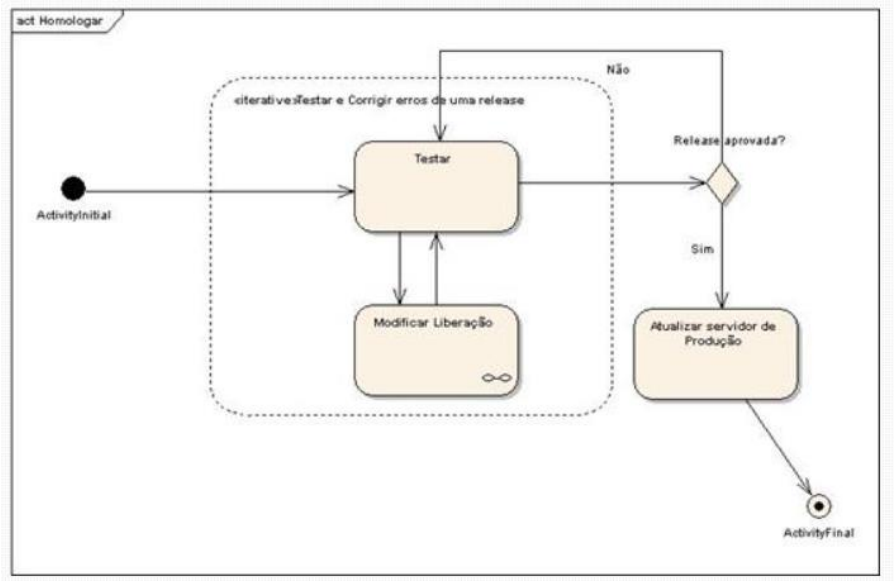

Figura 12. Fase de Homologação

Para que itens de configuração de uma baseline ou release possam evoluir de forma controlada, a função de controle da configuração estabelece as atividades:

solicitação de modificação, que pode ser corretiva, evolutiva, adaptativa ou preventiva;

classificação da modificação, que estabelece a prioridade da solicitação em relação às demais solicitações efetuadas anteriormente;

análise de impacto, que visa relatar os impactos em esforço, cronograma e custo;

avaliação da modificação, que estabelece se a modificação será implementada, rejeitada ou postergada, em função do laudo da análise de impacto da modificação;

implementação da modificação, caso a solicitação tenha sido aprovada pela avaliação da modificação;

verificação da modificação com relação à proposta de implementação levantada na análise de impacto;

gerência da baseline e release. Uma nova baseline ou release pode ser criada ao final, agrupando uma ou mais solicitações resolvidas. A Figura 13 ilustra esta função que está relacionada com a atividade modificar liberação executada na fase de homologação do produto.

Nota 2: Uso do processo de gerência de configuração

No caso de artefatos ainda em desenvolvimento, recomenda-se um processo mais ad-hoc, que não faça com que a dinâmica do desenvolvimento seja perdida. Neste caso, apenas as operações check-out e check-in são necessárias. Mas, recomenda-se que um resumo do que foi implementado seja exposto no comentário a cada operação de check-in realizada, juntamente com o item de configuração e sua respectiva versão no repositório.

No caso de baselines ou releases, isto é, versões estáveis do produto, é ideal ser mais formal, pois qualquer mudança nos itens de configuração pode causar impacto em custo e prazo de entrega do produto. O formalismo aplicado, no entanto, pode variar em função da flexibilidade que determinados processos de desenvolvimento de software necessitam.

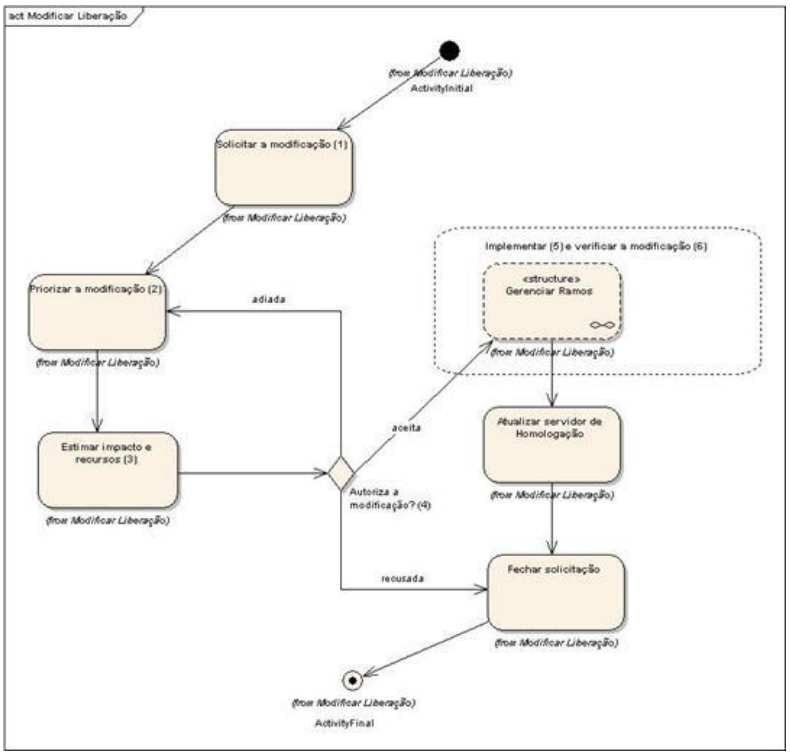

Figura 13. Modificar Liberação

Analisando as atividades da Figura 13, vemos que para implementar e verificar a modificação podemos criar um ramo separado. Portanto, podemos criar um ramo para corrigir os erros encontrados, adotando a estratégia de organização em série, como vimos na seção Estratégias de Organização. Desta forma, a implementação de novas funcionalidades do produto pode continuar na mainline, considerando as próximas iterações do desenvolvimento, enquanto a correção dos erros é feita em um ramo à parte.

Neste cenário, os erros são corrigidos no ambiente de desenvolvimento, e após todas as verificações, uma release é criada. Ao final, atualiza-se o ambiente de homologação com a nova release. Quando a release for aprovada pelo cliente, uma versão do produto pode ser posta em produção. A Figura 14 detalha este processo.

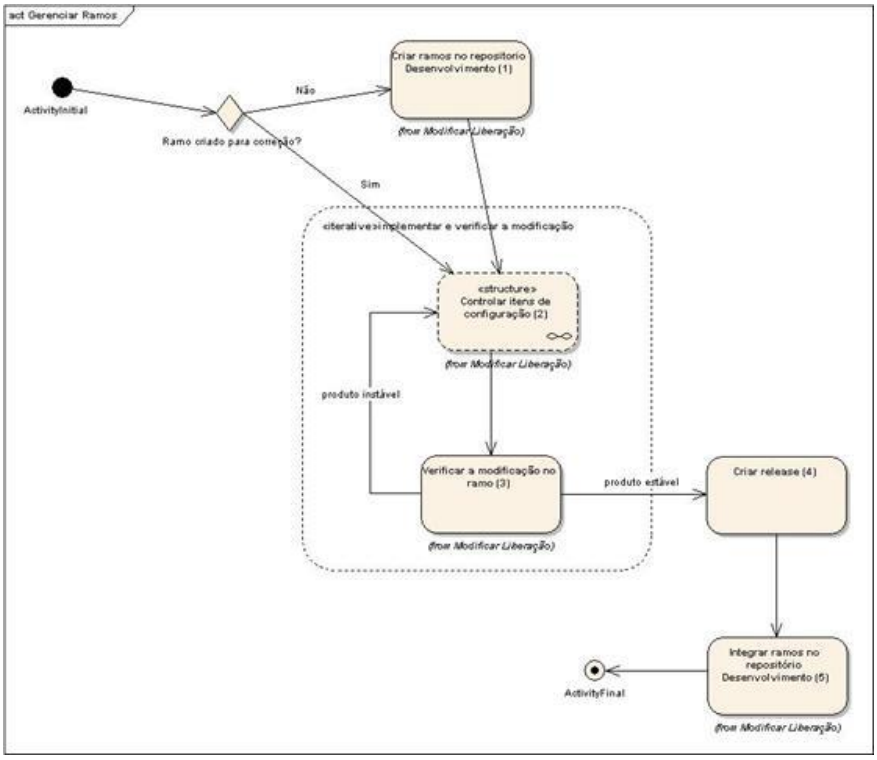

Figura 14. Gerenciar ramos

A atividade gerenciar ramos na Figura 14 mostra como proceder a implementação e verificação das correções dos erros encontrados na fase de homologação. Esta atividade apresenta os seguintes passos:

criar ramo no repositório,

controlar itens de configuração no ramo durante a atividade de implementação e teste,

verificar se os erros foram corrigidos corretamente,

criar release com um conjunto de solicitações atendidas,

integrar na mainline as correções feitas no ramo em função das solicitações.

Neste contexto, controlar itens de configuração significa: efetuar check-out atualizando o espaço de trabalho com os itens contidos no ramo criado no repositório; alterar os itens e efetuar check-in, ao terminar a modificação, devolvendo os itens do espaço de trabalho para o repositório.