Unidade IV

7 MANUTENÇÃO DE *SOFTWARE*

7.1 Definição de manutenção de *software*

As manutenções de *software* são correções, evoluções ou adaptações técnicas que são realizadas no *software* já construído e em produção, para adequá-lo às mudanças de requisitos dos usuários, às novas funcionalidades solicitadas, para a atualização de regras de negócio e para a adaptação do *software* ao mercado ou às necessidades de atualizações tecnológicas que são demandadas durante o ciclo de vida de um *software*. Este é uma entidade viva, ou seja, precisa sofrer alterações e evoluções para continuar a ser utilizado de maneira adequada pelos usuários.

As manutenções de *software* ocorrem não apenas para corrigir defeitos em um *software*; são bem mais amplas que isso. Surgem em razão de diversos fatores, mas se destacam pelas mudanças e alterações relacionadas a:

- nova ordem estratégica na organização;
- demandas para acompanhar os concorrentes;
- adequação para atender a novas leis e regras;
- novas características sugeridas por clientes;
- evoluções da tecnologia.

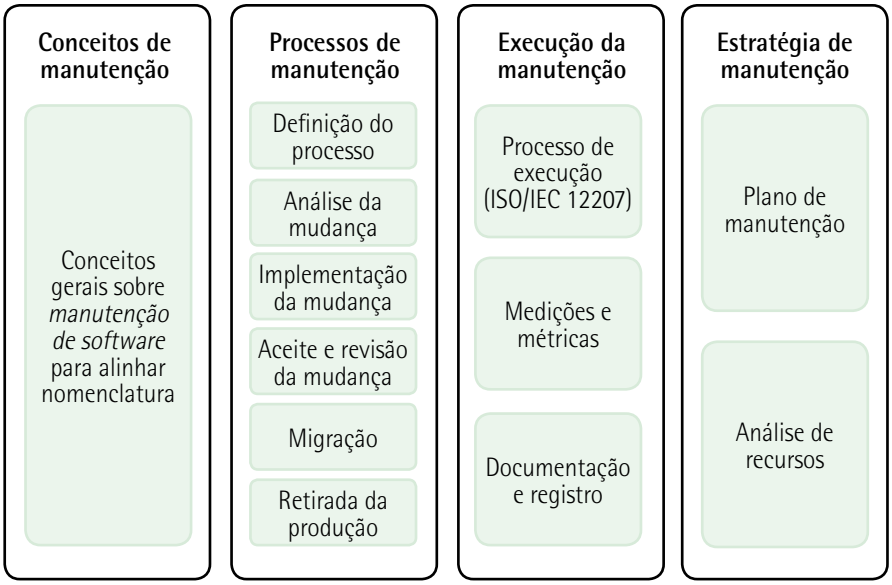

Figura 52 – Estrutura da norma ISO/IEC 14764

Segundo Boehm (1988), a fase de manutenção é a mais problemática do ciclo de vida de um *software* e representa mais de 60% de todo o esforço em Tecnologia da Informação para uma organização. A maioria desses problemas está relacionada à maneira pela qual o *software* foi construído, normalmente, sem documentação ou documentação desatualizada, sem padrões de codificação, sem boas práticas de programação e, muitas vezes, sem controle adequado das evoluções, gerando um grande esforço das equipes para encontrar e corrigir defeitos ou para criar novas funcionalidades, fazendo a atividade de manutenção ser evitada pela maioria dos desenvolvedores. Com isso, acaba ficando nas mãos de desenvolvedores menos experientes, agravando o problema.

A Norma ISO/IEC 12207, vista na unidade I, descreve um conjunto de atividades para a realização da manutenção de um *software*, apresentadas como um processo fundamental da área de Engenharia, mas a norma ISO/IEC 14764 (2006), que é apresentada neste tópico e cuja estrutura está representada na Figura 52, é específica para descrever o ciclo de manutenção do *software*, definindo os conceitos envolvidos em manutenção e estabelecendo um processo claro para tratar e apoiar os desenvolvedores nessa complexa etapa do ciclo de um *software*.

Saiba mais

Detalhes sobre a norma ISO/IEC 14764 podem ser obtidos em: <http://www.iso.org/>.

7.1.1 Conceituação sobre manutenção de *software*

A Norma ISO/IEC 14764 (2006) considera as seguintes definições para a unificação da terminologia utilizada no contexto de manutenção de *software*:

- manutenção: modificação de um produto de *software* após a entrega para corrigir defeitos, melhorar o desempenho ou outros atributos, para adaptar o produto a um ambiente de mudança, incluir novas funcionalidades ou recursos não previstos anteriormente;

- evolução: a evolução de *software* que gera uma nova versão do produto com um conjunto relevante de alterações solicitadas após a colocação do *software* em produção;

- alteração: mudanças realizadas no código do *software*; engloba as correções de defeitos e os novos requisitos propostos pelo usuário;

- melhoramento: mudança realizada no *software* com o objetivo de melhorar o desempenho, a segurança ou adaptá-lo a novos ambientes;

- linha de base: uma versão aprovada formalmente pelo cliente em um determinado momento do ciclo de vida do *software*;

- pedido de mudança: termo utilizado para caracterizar a solicitação de uma mudança; após a correção, pode ser classificado como uma correção ou uma melhoria, conforme ilustrado na Figura 53;

- plano de manutenção: documento que descreve as práticas específicas, os recursos e as atividades relevantes para a manutenção; esse documento é elaborado pelo responsável pela manutenção;

- processo de manutenção: contém um conjunto de tarefas para a realização da manutenção; é disparado quando há um pedido de mudança para um produto de *software*.

7.1.2 Tipos de manutenção

Ainda segundo a norma ISO/IEC 14764, as manutenções são classificadas em quatro tipos distintos e ilustradas na Figura 53:

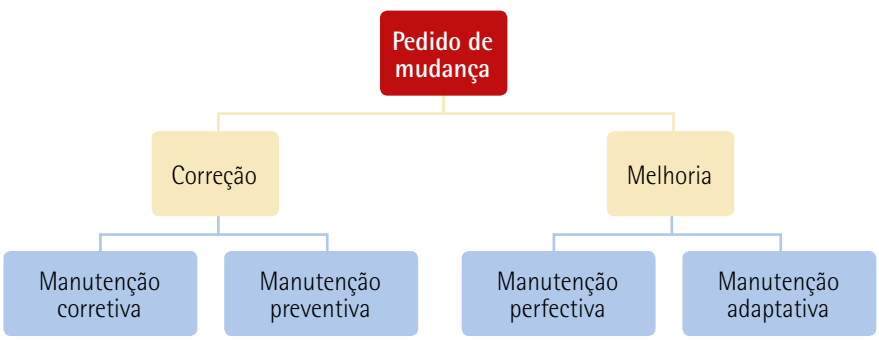

Figura 53 – Classificação de um pedido de mudança

- Manutenção corretiva: são as ações efetuadas para corrigir defeitos encontrados em um *software* em produção normalmente identificados pelos usuários finais. Na maioria dos casos, essas

manutenções possuem equipe dedicada, são atividades rotineiras para a solução de defeitos e a atuação é reativa e emergencial. Esse tipo de manutenção tem características de proatividade e é planejado.

- Manutenção perfectiva: são as ações de manutenção para incluir novas funcionalidades ou alterações que têm por objetivo satisfazer novas necessidades dos usuários. Surgem principalmente a partir da evolução do processo de negócio e para igualar ou superar concorrentes no mercado.

- Manutenção adaptativa: são manutenções feitas para adequar o *software* a novas tecnologias, metodologias, modelos de gestão ou à nova legislação. Muitas vezes são tratadas de forma não planejada e reativa.

- Manutenção preventiva: são manutenções proativas e planejadas para melhorar algum aspecto deficiente no *software*, como desempenho, segurança, defeitos ainda não identificados pelos usuários, usabilidade, dentre outros.

Lembrete

A manutenção perfectiva é mais conhecida no mercado como **manutenção evolutiva**.

A classificação da manutenção em tipos permite o melhor direcionamento dos esforços e a alocação dos recursos humanos necessários para realizar as atividades de acordo com a demanda. De posse dessas informações, o plano de manutenção pode ser elaborado e aprovado pelos envolvidos, formalizando o processo de manutenção. Como pode ser observado na Figura 54, que apresenta a distribuição média dos esforços gastos por uma organização por tipo de manutenção, os esforços com as manutenções perfectivas ou evolutivas é bastante superior às manutenções corretivas e adaptativas.

Observação

As manutenções perfectivas e adaptativas, normalmente, são tratadas como um projeto de *software*, e não como atividade rotineira.

Figura 54 – Distribuição de esforço por tipo de manutenção

7.2 Técnicas e padrões de manutenção de *software*

Um dos principais requisitos, já destacados por McCall, Richards e Walters (1977), para a manutenção de um *software* é a preocupação constante com a manutenibilidade deste, desde a sua concepção e durante todo o seu ciclo de vida.

A manutenibilidade indica quão fácil é fazer uma alteração, uma correção ou uma melhoria em um determinado *software* e é qualificada em função da preocupação com a clareza e as boas práticas da codificação, o uso de padrões no projeto e a boa documentação elaborada durante o desenvolvimento.

Embora o grau de manutenibilidade seja qualitativo, algumas métricas podem ser utilizadas pelas organizações visando avaliar a facilidade de manutenção e criar mecanismos para correção, tais como:

- Em quanto tempo um defeito é encontrado?
- Quanto tempo leva para corrigir o defeito?
- Quanto tempo é necessário para validar a correção do defeito?
- Quanto tempo decorre entre a identificação do defeito e a implantação da correção?

Lembrete

Métricas quantitativas são criadas para que, a partir de uma avaliação inicial, sejam feitas medidas constantes a fim de que o processo possa ser melhorado dentro de um período de tempo.

7.2.1 As atividades de manutenção

A execução da atividade de manutenção é iniciada por uma solicitação de mudança de um usuário do sistema e submetida à avaliação de um grupo responsável por aprovar a mudança. Uma vez aprovada, esta é encaminhada à gerência da manutenção, que irá classificar e priorizar o seu desenvolvimento, e, após isso, a mudança será construída. O ciclo de desenvolvimento de uma manutenção segue os mesmos passos que o da construção de sistema novo.

Figura 55 – Atividades para a realização de uma manutenção

Porém, existem exceções para aquelas manutenções chamadas de emergenciais, como correção de defeitos em produção, alterações críticas no ambiente ou regras de negócio essenciais para o bom funcionamento da aplicação. Esse fluxo é apresentado na Figura 55.

7.2.2 Papéis e responsabilidades na manutenção

Em uma manutenção, a determinação de papéis e responsabilidades claros permite a execução de um fluxo de trabalho estruturado, reduz substancialmente os conflitos e melhora o processo de comunicação. Os principais papéis em uma manutenção estão expostos a seguir.

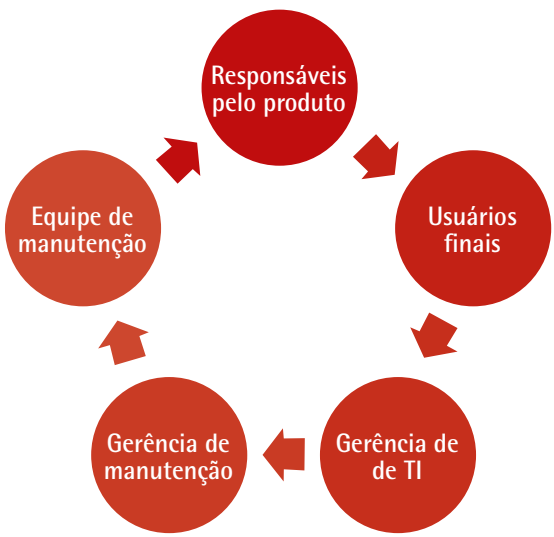

Figura 56 – Atividades para a realização de uma manutenção

- Usuários: são os usuários finais do produto de *software*. Normalmente, são eles que identificam os defeitos que exigem correções imediatas ou apontamento de melhorias necessárias na aplicação. Também são responsáveis pelo aceite final das alterações executadas.

- Responsáveis pelo produto: são aqueles que solicitaram o *software* original. São responsáveis pela evolução de negócio da aplicação tratada pelas manutenções evolutivas e por prover os recursos financeiros para a realização das manutenções do sistema. Também devem dar o aceite nas melhorias e participar do processo de análise da mudança, podendo acatar ou não as solicitações feitas pelos usuários.

- Gerência de TI: são aqueles que cuidam da aplicação e são responsáveis pelo ambiente técnico e pela evolução tecnológica do *software* (manutenções adaptativas). Devem sugerir melhorias técnicas aos responsáveis pelo produto, manter o *software* operando de maneira adequada e participar da análise dos pedidos de mudança.

- Gerência de manutenção: é o responsável técnico por cuidar das manutenções aprovadas. Analisa, classifica e prioriza as manutenções em conjunto com o responsável pelo produto a fim de determinar o que e quando as solicitações de mudança serão atendidas.

- Equipe de manutenção: realiza as mudanças aprovadas e priorizadas pela gerência. Normalmente atendem as solicitações de alteração emergenciais e pequenas melhorias. As mudanças maiores, na maioria das vezes, são tratadas como um projeto à parte.

7.2.3 Manutenção de sistemas legados

Um *software* legado é uma aplicação desenvolvida no passado e que ainda está em uso numa organização. Trata-se de aplicações que sofreram muitas mudanças e adaptações, possuem tecnologia obsoleta, cuja documentação é praticamente inexistente e não há mais pessoas que participaram do desenvolvimento original. Em razão dessas características, tais *softwares* são bastante sensíveis a novas mudanças e se gasta mais tempo e dinheiro para executar as alterações. Portanto, qualquer mudança deve ser cuidadosamente analisada, e testes rigorosos devem ser realizados para evitar surpresas desagradáveis durante o desenvolvimento ou na fase de aceite pelos usuários.

7.2.4 Engenharia reversa

Segundo Chikofsky e Cross II (1990), engenharia reversa é um processo de análise e entendimento de um produto de *software*, com o objetivo de recapturar e decifrar os seus requisitos a partir da sua codificação existente para melhorar a qualidade do produto, reduzir custos de manutenção e atualizar tecnicamente o *software*.

A engenharia reversa é uma forma de manutenção muito utilizada para criar novos produtos de *software* a fim de substituir os sistemas legados, em razão das dificuldades que as organizações têm para manter esses sistemas legados funcionando adequadamente.

7.2.5 Técnicas de manutenção

As manutenções em um *software* podem ser classificadas em dois tipos básicos: a manutenção não estruturada e a manutenção estruturada.

A manutenção não estruturada é aquela na qual a análise de um pedido de manutenção é feita por uma verificação direta no código-fonte, uma vez que este é a única documentação disponível no produto existente. Trata-se de uma atividade demorada, delicada e não é possível verificar os requisitos não funcionais da aplicação. Cada correção de defeito é um desafio. Na maioria das vezes, essa manutenção é executada por pessoal com pouca experiência e com pouco conhecimento do *software*, o que gera desmotivação e insatisfação da equipe e dos usuários.

A manutenção estruturada é aquela que é executada em um *software* que tem uma estrutura documentada e que foi construído seguindo as melhores práticas de desenvolvimento de *software*. Nesse tipo de manutenção, a análise e a correção de defeito são mais rápidas e menos custosas para desenvolvedores e usuários, tornando os resultados melhores e com maior qualidade.

7.2.6 Custos de manutenção

Os custos com a manutenção de uma aplicação são bem superiores aos custos de desenvolvimento de um *software* novo, pois o tempo de vida do *software* em manutenção é bem maior do que o tempo de criação. Além disso, à medida que o *software* vai sofrendo modificações por diversas pessoas ao longo da sua vida útil, as alterações vão se tornando cada vez mais complexas e demoradas. Como exemplo, podemos citar os sistemas legados, vistos anteriormente.

De acordo com Chikofsky (1990), os seguintes fatores são relevantes e afetam diretamente os custos de manutenção de um produto de *software* durante o processo de manutenção:

- Estabilidade da equipe: se a equipe de manutenção for fixa e tiver baixa rotatividade, os custos com a manutenção serão menores e a produtividade será maior que numa situação de trocas constantes.

- Responsabilidade contratual: um *software* novo é desenvolvido por uma equipe por meio de um projeto. Não há garantia, para essa equipe, de que a manutenção será de sua responsabilidade, portanto não há motivação para criar um *software* de fácil manutenção. A saída é fazer a revisão e a inspeção do *software* enquanto é produzido.

- Habilidade da equipe: as atividades de manutenção não são consideradas nobres pelos desenvolvedores, e a equipe designada, com frequência, tem menos experiência e pouco conhecimento do negócio.

7.3 Processos de manutenção

Um processo de manutenção define o conjunto de atividades necessárias para a realização de uma modificação em um produto de *software* para que este atenda as necessidades dos usuários.

A Norma ISO/IEC 14764 (2006) define um processo de manutenção que descreve as tarefas e atividades necessárias para modificar um produto de *software*, mantendo sua integridade. Vale lembrar que o processo de manutenção não determina como fazer o desenvolvimento da manutenção, mas

descreve as atividades de identificação da mudança, análise e aceitação das mudanças, eventual migração ou retirada do *software* de produção.

As atividades que compõem o processo de desenvolvimento são:

- definição do processo de manutenção;
- análise das mudanças;
- desenvolvimento da mudança;
- aceitação e revisão da mudança;
- migração;
- retirada do *software* de produção.

Essas atividades são apresentadas na Figura 57, exceto a atividade de desenvolvimento da mudança, que trata do processo de desenvolvimento de *software* que é direcionado para a norma ISO/IEC 12207. Na prática, significa que a organização pode utilizar a sua própria metodologia para implementar as manutenções identificadas.

Figura 57 – Atividades para a realização de uma manutenção

7.3.1 Definição do processo de manutenção

A definição do processo de manutenção envolve a criação do procedimento a ser adotado pela equipe de desenvolvimento para a execução de uma manutenção. O procedimento deve descrever o que deve ser feito desde o pedido de manutenção até o seu aceite por parte dos usuários envolvidos. Além disso, deve determinar os papéis e as responsabilidades dos envolvidos no processo de manutenção.

Conforme ilustrado na Figura 58, a partir de um pedido de mudança, a equipe de manutenção deve identificar o tipo de manutenção, por exemplo: uma correção ou uma nova funcionalidade. Se for uma nova funcionalidade, a equipe de manutenção deverá fazer a análise, a classificação e a priorização da mudança. Se for uma correção de defeito, deverá primeiro avaliar a gravidade. Se for uma ocorrência

em produção, a mudança deverá ser executada imediatamente, caso contrário passará pelo processo de avaliação. Porém, esse procedimento é apenas um exemplo, podendo a organização estabelecer o processo que melhor se adapta a sua situação cotidiana.

7.3.2 Análise das mudanças

A análise de mudanças envolve o conjunto de atividades para entender o que precisa ser realizado, classificar o tipo de manutenção envolvida e o tempo necessário para a implementação da mudança. Deve ser realizado por um grupo de manutenção que envolva o pessoal de desenvolvimento e os usuários, a fim de determinar corretamente o tipo e a prioridade da alteração solicitada.

Essas atividades envolvem:

- analisar o pedido de mudança, identificar o seu tipo e definir claramente os requisitos da manutenção;
- identificar o que precisa ser realizado no código e realizar a documentação do sistema;
- criar as opções de alterações possíveis e estimar pessoal e tempo de alteração necessário para a opção escolhida;
- submeter a solução à aprovação dos usuários.

A análise da mudança é diretamente afetada pelo fato de a classificação da manutenção ser estruturada ou não estruturada. Como visto anteriormente, as manutenções não estruturadas trazem complicações à análise e maiores riscos à mudança que precisa ser realizada.

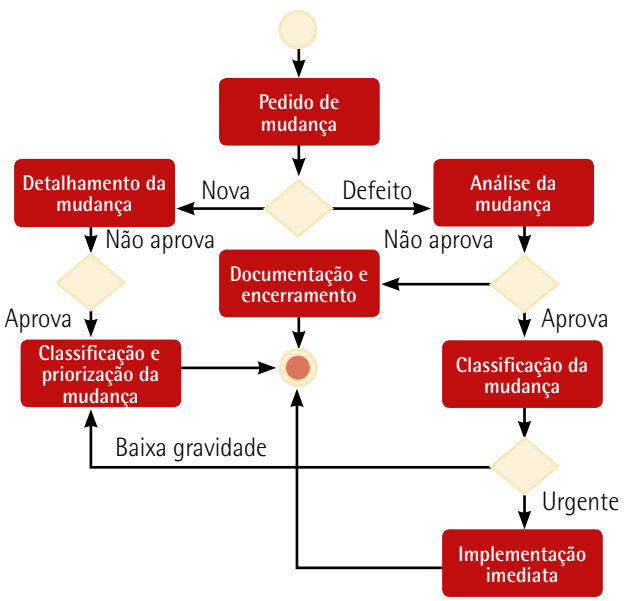

Figura 58 – Exemplo de um fluxo de processo de manutenção

7.3.3 Aceitação e revisão da mudança

Envolvem as atividades para garantir que a mudança esteja corretamente implementada e atenda aos requisitos solicitados. Além de incluir as atividades de garantia da qualidade, deve prever o aceite da mudança efetuada pelo usuário. Algumas atividades envolvidas são:

- revisões;
- inspeções;
- testes;
- testes de regressão;
- processo de aceitação.

Após o aceite da manutenção, a documentação deve ser atualizada e arquivada, além de ser necessária a geração da nova linha de base da aplicação.

Observação

No caso de não existir um roteiro de testes da aplicação, a equipe de manutenção deve criar um roteiro específico para a mudança que está em desenvolvimento.

7.3.4 Migração

No ciclo de vida de um *software* podem ocorrem mudanças que exijam migrar os dados de um ambiente para outro. Normalmente, ocorrem em função de uma mudança adaptativa ou da realização da engenharia reversa em alguma aplicação.

A migração dos dados é uma tarefa delicada e exige a elaboração de um plano de migração para a garantia da integridade das informações. Esse plano de migração deve conter:

- análise dos requisitos e definição da migração;
- criação do processo de **de-para** da migração;
- definição de conversões de dados necessárias;
- identificação dos riscos da migração;
- execução do processo de migração;

- verificação dos resultados da migração;
- definição do processo de validação dos usuários;
- criação de um processo de retorno, em caso de insucesso da migração;
- atualização da documentação.

Uma vez elaborado o plano de migração, a equipe de manutenção deve apresentá-lo aos usuários para que seja avaliado em conjunto e aprovado por todos os envolvidos no processo. Além disso, deve descrever e apresentar aos usuários as necessidades relacionadas às pessoas e ao tempo de dedicação necessário para a execução das atividades de migração.

Observação

O termo **de-para** significa que o desenvolvedor deve identificar, para cada campo da tabela de dados Origem, qual o seu equivalente na tabela de dados Destino.

7.3.5 Retirada de produção

Consiste em uma análise técnica ou econômica para determinar se o *software* deve permanecer em produção ou ser descontinuado. Essa decisão deve ser conjunta, entre a equipe de desenvolvimento e os usuários, a fim de garantir o investimento necessário para a manutenção do *software* existente ou para a aquisição ou o desenvolvimento de um novo produto.

Após a decisão pela descontinuidade, é elaborado o plano de retirada de produção envolvendo o tempo em que a manutenção deve permanecer em operação até a desativação do sistema, o tempo de transição (paralelismo) entre o novo sistema e o sistema atual, a obtenção e a documentação dos requisitos do sistema a ser descontinuado, o plano de migração de dados e o registro e o arquivamento da documentação.

Lembrete

A descontinuidade de um *software* não tem prazo determinado. Tudo depende de como foi construído, de como atende aos usuários atualmente e dos custos para mantê-lo funcionando.

7.4 Impactos na manutenção

Como visto até aqui, a manutenção é diretamente afetada pela forma que o *software* foi construído, a qual pode gerar maior ou menor esforço e mais ou menos custos durante o ciclo de manutenção, causando impactos diversos no processo.

Destacam-se os impactos gerados pelas práticas mais comuns no mercado de Tecnologia da Informação: o desenvolvimento orientado a objetos, o desenvolvimento baseado em componentes e os processos de desenvolvimento ágeis, que são abordados a seguir e ilustrados na Figura 59.

Figura 59 – Impactos na manutenção de *software* baseado em métodos ágeis

7.4.1 Desenvolvimento orientado a objetos

O desenvolvimento orientado a objetos se baseia em objetos do mundo real que interagem entre si e são acionados por pessoas do mundo real chamadas de atores. Um projeto orientado a objetos é composto de classes que são a representação de um objeto de negócio do mundo real com suas próprias características (atributos) e seus próprios comportamentos (métodos) que executam as ações necessárias para o negócio. Sua implementação garante códigos reduzidos nos métodos e responsabilidade dos objetos, permitindo a extensão desses comportamentos. A codificação se dá pela interação desses objetos, permitindo a separação clara das responsabilidades e ações de cada classe. Os conceitos de herança e agregação aumentam a capacidade de adaptação de um *software* desenvolvido com orientação a objetos.

A concepção do projeto normalmente é no padrão *Model View Controller* (MVC), em que cada camada *view* é responsável pela interface com o usuário, a camada *model*, pela construção das regras de negócio, e a *controller*, pela ligação entre as camadas *view* e *model*. A construção orientada a objetos associada ao padrão MVC permite o perfeito isolamento das funções e a evolução da aplicação. Por exemplo, um *software* orientado a objetos no padrão MVC viabiliza a existência de aplicações para qualquer mídia na camada visual, e a camada de negócio permanece sem alterações.

Seguindo os princípios básicos da orientação a objetos e do padrão de arquitetura MVC na construção de um *software*, vários impactos podem ser destacados na manutenção de uma aplicação. Destacam-se os seguintes durante o processo de manutenção, ilustrados no Quadro 21:

Quadro 21 – Impacto na manutenção de *software* desenvolvido com orientação a objetos

Tipo	Impacto	Observações
Corretiva	Ajuste pontual nas classes envolvidas	Testes devem ser realizados em todas as funcionalidades que utilizam as classes alteradas. Teste de regressão é recomendado
Perfectiva	O modelo objeto permite a extensão das classes existentes e a criação de novas classes sem afetar a aplicação existente	Depende da correta modelagem dos objetos para garantir que as classes existentes tenham apenas extensões e não se modifique o que já está funcionando Os testes devem envolver o que foi criado de novo e os testes de regressão das demais funcionalidades
Adaptativa	Pela estrutura de objetos, deverá gerar alterações nas classes envolvidas e nas camadas que incluírem novos comportamentos	Demandam a realização de testes completos do sistema
Preventiva	Dependerá da alteração que for gerada pelo processo	A alteração pode gerar uma manutenção corretiva, perfectiva ou adaptativa

7.4.2 Desenvolvimento baseado em componentes

O desenvolvimento baseado em componentes tem como objetivo criar estruturas de código que possibilitem a sua utilização em mais de um sistema. Esse conceito é conhecido como reúso e tem se tornado uma abordagem comum no mercado de Tecnologia da Informação. A ideia é criar componentes que não sejam específicos para um determinado sistema, mas com uma visão de uso genérico para atender a todos os sistemas. Por exemplo, ao criar uma classe-cliente para um sistema de vendas, o projetista deve pensar que essa classe pode ser usada no sistema de cobrança, no de faturamento, dentre outros. Os benefícios fornecidos por uma abordagem baseada em componentes são: criar um a estrutura de código que aumenta a velocidade de desenvolvimento dos sistemas e atender à demanda dos sistemas atuais, que se tornam cada vez maiores e mais complexos.

Na visão da manutenção para sistemas baseados em componentes, podemos analisar os impactos a partir das alterações nos componentes construídos e submetidos a mudanças pelos processos de manutenção, ilustrados no Quadro 22:

Quadro 22 – Impacto na manutenção de *software* desenvolvido com base em componentes

Tipo	Impacto	Observações
Corretiva	As correções nos componentes são mais delicadas, em razão do uso genérico. As correções devem ser publicadas para todos os que utilizam o componente	Necessários testes rigorosos com o componente e as partes dos sistemas que o utilizam, antes de colocar em produção
Perfectiva	Alterações evolutivas nos componentes devem manter a compatibilidade com as versões anteriores. Os componentes não podem ser simplesmente alterados e devem manter o legado estável	Necessários testes com as novas funções do componente e dos sistemas que o utilizam
Adaptativa	Evoluções tecnológicas devem gerar novas versões dos componentes. Não alterar os existentes	Testes rigorosos com os novos componentes como se fossem criados no momento
Preventiva	Depende da alteração que for gerada pelo processo	A alteração pode gerar uma manutenção corretiva, perfectiva ou adaptativa

Saiba mais

Para mais informações sobre o desenvolvimento baseado em componentes e orientado a objetos, consulte o livro:

SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson, 2013.

7.4.3 Desenvolvimento ágil

O desenvolvimento ágil é um processo de desenvolvimento de *software*, e não uma estrutura para construir um *software*. Suas características estão relacionadas à rapidez, à integração cliente-equipe e à alta qualidade durante o ciclo de construção, podendo utilizar uma estrutura orientada a objetos ou baseada em componentes. Portanto, a manutenção é afetada pelas características do processo ágil que alteram profundamente a relação cliente-desenvolvedor, e não pela estrutura de construção da aplicação. O desenvolvimento pode utilizar tanto a orientação a objetos quanto o desenvolvimento baseado em componentes. Olhando sob o aspecto dos processos ágeis, podemos analisar os impactos da manutenção da seguinte forma, no Quadro 23:

Quadro 23 – Impacto na manutenção de *software* baseado em métodos ágeis

Tipo	Impacto	Observações
Corretiva	No processo ágil, a correção é realizada de forma interativa para atender rapidamente ao usuário	Validações são realizadas ao fim da codificação
Perfectiva	Novas funcionalidades são tratadas como mudanças no *software*	O uso de métodos ágeis na manutenção implica a existência de equipe treinada e que entenda do negócio em evolução
Adaptativa	Tratadas como um novo projeto de acordo com o método ágil	Deve seguir o padrão de desenvolvimento ágil utilizado
Preventiva	Dependerá da alteração que for gerada pelo processo	A alteração pode gerar uma manutenção corretiva, perfectiva ou adaptativa

É importante salientar que, independentemente do processo de desenvolvimento ou da arquitetura utilizada para a criação do *software*, a manutenção sempre apresentará alto grau de dificuldade no gerenciamento e no controle das alterações, pois sempre impactará diretamente os usuários finais que já utilizarem o *software* em produção, podendo trazer impactos à imagem da empresa e também à parte financeira.

8 GESTÃO DA CONFIGURAÇÃO

O Gerenciamento de Configuração de *Software* (GCS), também conhecido como *Software Configuration Management* (SCM), é um processo da engenharia de *software* que tem por objetivo estabelecer as condições para a organização dos artefatos de *software*, controlar suas versões e suas alterações, manter a integridade de tudo o que foi produzido durante o ciclo de vida de um projeto de software e garantir seu armazenamento adequado. Esses conceitos são ilustrados na Figura 60.

8.1 Conceitos de gestão da configuração

Segundo o SEI (2006), a gestão de configuração envolve identificar o que deve ser controlado, gerenciar suas alterações e manter a integridade e a rastreabilidade dos artefatos de um *software*.

Os objetivos essenciais da GCS são: aumentar a produtividade do time, diminuir o esforço manual de controle e reduzir os erros causados pela falta de controle dos artefatos de projeto.

Já que uma mudança pode ocorrer a qualquer tempo, as atividades de GCS são desenvolvidas para:

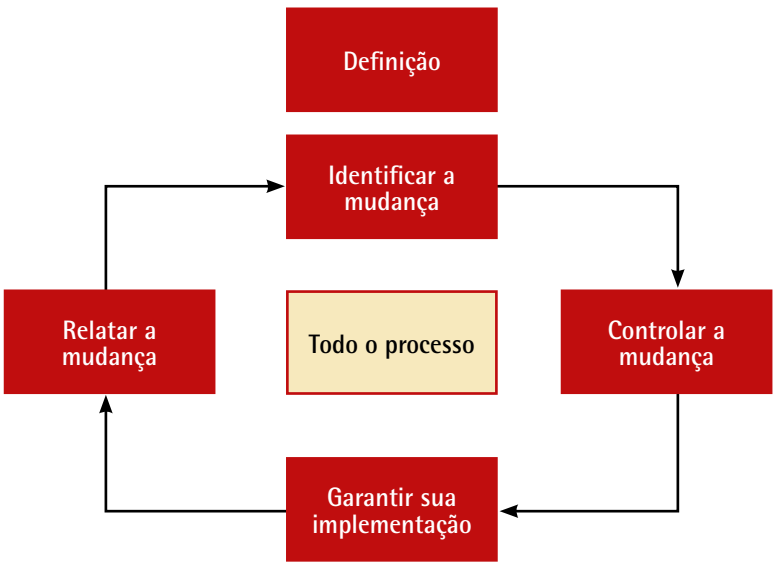

Figura 60 – Conceitos de gestão da configuração de *software*

Lembrete

Manter a integridade dos artefatos significa garantir que, dado um artefato, somente pessoas autorizadas podem alterá-lo, e todas as alterações serão registradas.

8.1.1 Problemas mais comuns

Durante o ciclo de desenvolvimento de um *software* são produzidos vários artefatos solicitados pelo cliente, pela metodologia e pelos usuários, bem como diversos arquivos de código-fonte. Manter esse conjunto de artefatos organizado e controlado não é uma tarefa trivial, principalmente, quando o número de desenvolvedores é muito grande, trabalhando em locais físicos diferentes e compartilhando esses artefatos com outras equipes e com o próprio cliente. As seguintes frases são muito ouvidas em cenários de desenvolvimento de *software*: "Qual versão do documento de requisitos enviamos ao cliente, mesmo?", "Quem enviou?", "Procure no *e-mail*...". São comentários assustadores em qualquer ambiente de projeto de *software*.

São apresentadas, a seguir, situações corriqueiras que envolvem essa concorrência pela informação, compartilhamento de arquivos e o controle sistemático desses artefatos produzidos.

Observação

Para fins conceituais, **artefato** é todo e qualquer documento produzido durante o projeto, inclusive o código-fonte.

8.1.1.1 Problema 1 – Relacionado a desenvolvedor com os artefatos isolados

Embora o projeto tenha vários desenvolvedores, um desenvolvedor trabalha na sua própria máquina, mantendo os arquivos sob seu domínio e, eventualmente, coloca-os no servidor, mas continua alterando a versão na sua própria máquina, como na Figura 61.

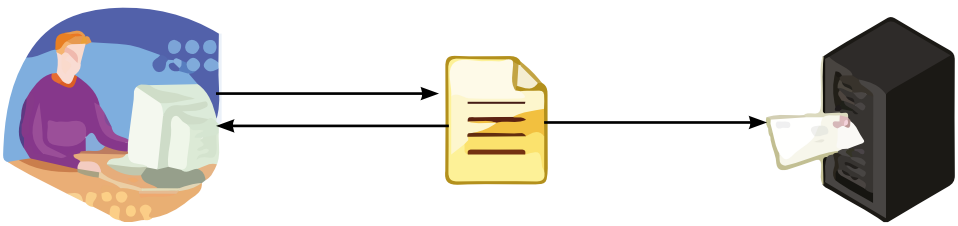

Figura 61 – Exemplo de desenvolvimento isolado

Nesse cenário, quando outro desenvolvedor precisar modificar o artefato, ele irá buscar no servidor, e a versão poderá estar desatualizada, ou nem estar disponível.

8.1.1.2 Problema 2 – Relacionado a artefatos compartilhados

Esse problema ocorre quando dois ou mais desenvolvedores estão fazendo alterações no mesmo artefato. Nessa situação, o desenvolvedor A modifica o artefato, e, mais tarde, o desenvolvedor B também faz alterações no mesmo artefato, sem saber que ele já foi alterado pelo desenvolvedor A, conforme ilustrado na Figura 62.

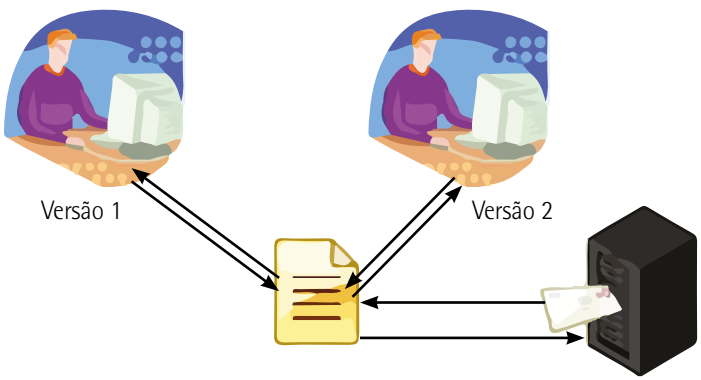

Figura 62 – Exemplo de artefatos compartilhados

Como consequência, quando o desenvolvedor A tentar compilar o artefato, serão apontados erros, mas nenhum deles causado pelas alterações que ele fez.

8.1.1.3 Problema 3 – Relacionado a manutenção simultânea

Ocorre quando cada desenvolvedor trabalha com uma cópia local do artefato, gerando dificuldades para identificar o que já foi realizado e para saber que defeitos foram corrigidos, conforme a Figura 63.

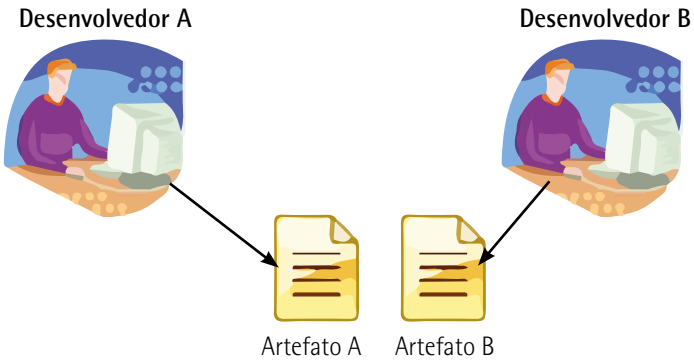

Figura 63 – Exemplo de manutenção simultânea

Nesse cenário, há grande dificuldade para identificar qual é a versão correta a ser utilizada no *software*.

8.1.1.4 Problema 4 – Relacionado com atualizações simultâneas

Essa situação ocorre quando um desenvolvedor A corrige um problema em sua versão de artefato e armazena-o num repositório central. O desenvolvedor B corrige outro problema no mesmo artefato e também o armazena no repositório central, ilustrado na Figura 64.

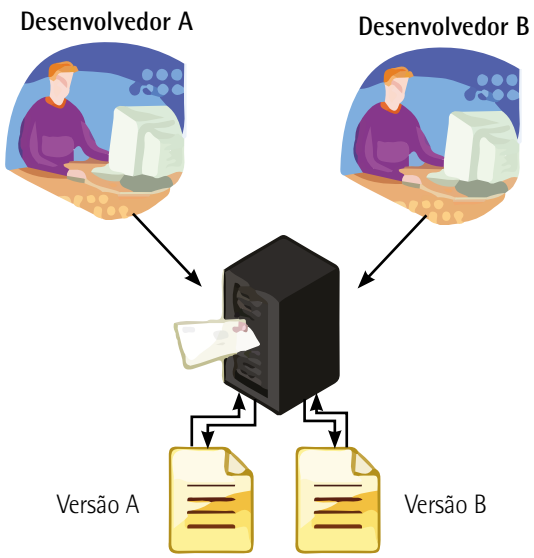

Figura 64 – Exemplo de atualização simultânea

No cenário descrito, os desenvolvedores não se preocupam com quem atualizou o quê, gerando retrabalho para todos.

A gestão da configuração existe para solucionar esses problemas e outros que surgem em relação à efetiva organização dos artefatos, permitindo à equipe de projeto ter mais rapidez na identificação e na solução de problemas, identificar quem fez as mudanças ao longo do tempo, controlar as versões desses artefatos e estabelecer padrões para toda a equipe.

8.1.2 Conceitos de gestão da configuração

Os profissionais de Tecnologia da Informação costumam confundir o termo **manutenção** com **configuração**, e o termo **configuração** com **controle de versões**, dentre outros. Visando ao melhor alinhamento do vocabulário, esses conceitos diferentes são vistos a seguir:

- Manutenção: mudanças e/ou alterações que são realizadas no *software* após sua conclusão e entrada em produção. Necessariamente, envolve alterações no código-fonte ou no ambiente da aplicação.

- Configuração: atividades de gerenciamento de todo o fluxo dos artefatos produzidos em um *software*, desde seu início até sua descontinuidade. Para isso, envolve o uso de ferramentas e não altera código de programas.

- Controle de versões: uma das atividades da GCS. Envolve o uso de uma ferramenta que controla o armazenamento dos artefatos e gerencia todas as alterações que esse artefato possa sofrer durante seu ciclo de vida.

- Itens de configuração: artefatos criados como parte do processo de desenvolvimento de um *software*. Uma vez identificados e determinados como controlados, esses itens de configuração vão estar sob a gestão da configuração.

- *Baseline*: fotografia do momento de um artefato, quando este é aprovado pelo cliente. Após a criação desta *baseline*, esse artefato só poderá sofrer alterações mediante autorização e aprovação formal do cliente, o que gera uma nova *baseline*. Trata-se de um ponto de controle no desenvolvimento de *software*, caracterizado pela entrega de um item de configuração para aprovação formal, por meio de uma RTF, por exemplo. Os artefatos que necessitam de criação de *baselines* são definidos pelos desenvolvedores e pelo cliente nas fases iniciais do projeto, e alguns exemplos por fase de desenvolvimento são ilustrados na Figura 65.

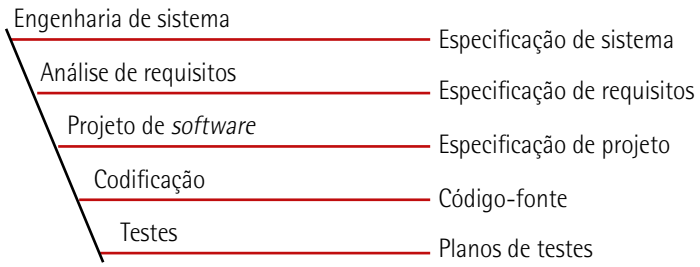

Figura 65 – Exemplo de artefatos que precisam de *baseline*

8.2 Processo de gestão da configuração

Antes de descrever as tarefas, é preciso definir a estrutura de armazenamento dos artefatos, a nomenclatura de cada um dos artefatos envolvidos no processo de gestão da configuração e quem pode fazer o quê com os artefatos.

Inicialmente é necessário identificar como os nomes dos artefatos são descritos. O nome deve conter um conjunto de caracteres que defina o artefato em si de modo único e contenha o nome do projeto e a versão a que se refere. Essa denominação é predefinida pela equipe de gestão de configuração da organização para atender todo o processo de desenvolvimento de *software*. Por exemplo, para denominar os artefatos para o projeto XPT001 que tem os seguintes itens de configuração:

Tabela 10 – Artefatos e respectivas nomenclaturas

Artefato	Nomenclatura
Documento de requisitos	XPT001_DOCREQ_V1.0.
Especificação de casos de uso	XPT001_ESPEC_UC_V1.0
Modelo de dados	XPT001_MER_V1.0.

Uma vez definida a nomenclatura dos artefatos, é preciso ter também a estrutura de armazenamento, que corresponde à árvore de diretórios nas quais os artefatos são distribuídos no projeto. Também são predefinidos pela equipe de gestão da configuração. Um exemplo dessa estrutura para o projeto XPT001 é apresentado a seguir.

```
/XPT001
   /versao1.0
            /Analise
            /Projeto
            /Codificação
            /Testes
   /versao1.1
            /Analise
            /Projeto
            /Codificação
```

Definidos a árvore de diretórios e os artefatos que fazem parte do GCS, o controle de acesso de segurança para eles também terá de ser definido. O controle de acesso é realizado com base na equipe do projeto fornecida e com as permissões atribuídas pelo gerente. Por exemplo, programadores têm acesso a leitura e gravação na pasta Codificação, mas apenas de leitura nas pastas Análise e Projetos.

Uma vez definida a nomenclatura, o controle de acesso à informação e a estrutura de organização dos artefatos, é necessária a definição do processo de gestão da configuração, que é dividido em cinco tarefas básicas:

- identificação dos itens de configuração;
- controle de versão;
- controle de mudanças;
- auditoria na configuração de *software*;
- registro do *status*.

8.2.1 Identificação dos itens de configuração

Um item de configuração é a menor unidade produzida em um projeto passível de controle. Esse item pode ser qualquer artefato produzido durante o ciclo de vida do projeto que precise ser controlado em suas diversas versões e aprovações pelo cliente.

O projeto pode produzir diversos artefatos, mas nem todos estão sob a gestão da configuração. A Tabela 11 apresenta uma lista dos artefatos mais comuns gerados durante o ciclo de desenvolvimento de um *software*.

Tabela 11 – Lista de artefatos de um projeto de *software*

ID	Artefato	ID	Artefato
1	Documento de requisitos	2	Cronograma
3	Plano de riscos	4	Protótipo de telas
5	Especificação caso-uso	6	Diagrama de casos de uso
7	Diagrama de classes	8	Documento de arquitetura
9	Diagramas de sequência	10	Diagramas de estados
11	Códigos-fonte	12	Casos de teste
13	Roteiro de testes	14	Plano de implantação
15	Manual de usuário	16	Modelo de dados
17	Plano de instalação	18	Pedidos de mudança

Essa lista traz exemplos de artefatos, que não se limitam a esses itens. Cada projeto tem o seu conjunto de artefatos definidos pela metodologia ou pelo cliente, e sobre esses artefatos deve ser gerada a lista de itens de configuração do projeto. Como informação complementar, em alguns projetos e clientes, as atas de reunião e os *e-mails* trocados durante o projeto são considerados itens de configuração.

Outros artefatos de *software*, como os programas executáveis, não são controlados pela configuração, mas o código-fonte a partir do qual é gerado tem de ser controlado. Outros itens precisam ser controlados para garantir a integridade do projeto, como versões de editores, compiladores e de *frameworks*, por exemplo, JVM 5.0, Struts 2.0, JSF 2, EJB 3.2, dentre outros.

8.2.2 Controle de versões

Combina procedimentos e ferramentas para gerenciar diferentes versões de itens de configuração criadas durante o processo de desenvolvimento de *software*. Permite conhecer e controlar a evolução dos artefatos por meio desses procedimentos, como ilustrado na Figura 66.

Algumas ferramentas mais utilizadas e conhecidas para o controle de versões são Visual Source Safe, PVCS, CVS, SubVersion, dentre outras.

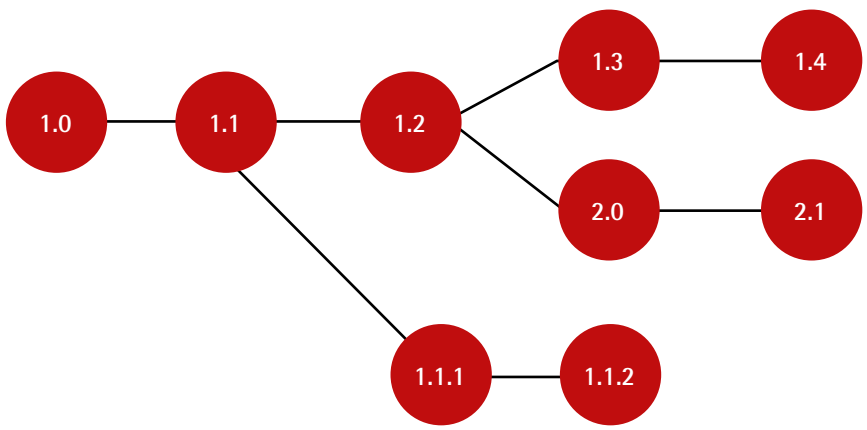

Figura 66 – Exemplo de evolução de versão de um artefato

Os procedimentos de controle de versões devem possuir um servidor próprio para ser o repositório dos arquivos controlados e não existem apenas para os desenvolvedores, mas sim para todos os envolvidos no projeto de *software*, como ferramenta central para a obtenção de qualquer informação sobre o *software*.

Alguns motivos para uso do controle de versões são:

- efetuar o registro histórico dos artefatos do projeto ao longo do tempo;
- permitir que desenvolvedores trabalhem juntos, sem que um atrapalhe o outro;
- promover segurança na manipulação e na alteração do código-fonte.

Com a utilização efetiva do controle de versões, basta acessar a ferramenta para saber da informação necessária. O controle de versões ainda permite o compartilhamento de todos os arquivos de forma segura e organizada, a recuperação fácil de qualquer artefato e a consulta de qualquer versão. Além disso, funciona como um *backup* de todas as informações do projeto.

Observação

Backup é uma cópia de segurança de tudo o que é produzido, realizada pela ferramenta de controle de versões de forma automática.

Para fazer funcionar um controle de versões, diversas operações são realizadas, independentemente da ferramenta utilizada. O procedimento básico do trabalho de controle de versões é:

- retirar uma cópia do artefato para alteração do repositório;
- fazer as alterações na cópia realizada;
- validar as alterações;
- gravar as alterações no repositório.

Vamos conhecer algumas operações importantes no controle de versões e alinhar a terminologia utilizada nessa tarefa:

- *Check out*: ação para retirar o artefato do repositório e copiar para o seu ambiente para poder fazer alterações. Ao realizar essa ação, você também pode bloquear o artefato para não permitir que ele seja alterado até você liberar o *check-in*.
- *Check-in*: ação de devolver o artefato retirado para alteração para o repositório central com as modificações realizadas. Se for código-fonte, somente faça se o programa estiver compilando sem erros. Também chamado de *commit*. Às vezes, é necessário realizar uma combinação de possíveis outras alterações feitas por outros usuários, chamada de *merge*.
- *Diff*: ação que compara a versão que está no repositório com a versão que você está alterando, ou compara duas versões que estão no repositório. São exibidas as diferenças entre os dois artefatos.
- *Merge*: atividade realizada antes de um *check out*, e depois de um *diff*, com o objetivo de verificar as diferenças entre duas versões de um artefato e garantir a integridade das alterações.
- *Tag* ou *label*: são rótulos atribuídos a um artefato para identificar o projeto e as versões aos quais ele pertence. Constituem uma das principais operações para o controle de versões. Todo artefato precisa de uma *tag*. Esta pode ser usada quando uma versão é criada, antes de se fazer uma alteração que envolva muitos artefatos ou numa situação julgada relevante para "marcar" a *tag*.

- *Branch*: são usados *branches* quando é preciso trabalhar em duas versões distintas de um projeto ao mesmo tempo. Por exemplo, após colocar a versão 1.0 de seu *software* em produção, você precisa continuar o desenvolvimento da versão 2.0. Nesse caso, caso ocorra um problema na versão 1.0, há como identificar tudo o que faz parte dessa versão para gerar uma nova *build*.

- *Build*: uma versão do *software* produzida para ser utilizada em testes, aceitação ou em produção. Podem ser manual ou automaticamente gerada por uma ferramenta específica.

8.2.3 Controle de mudanças

O controle de mudanças estabelece um processo para identificar e analisar as mudanças, aprovar e controlar sua implementação, garantir sua correção e publicar as mudanças de um item de configuração. É recomendado haver um comitê para o controle de mudanças, mas não é obrigatório. Essas atividades estão ilustradas na Figura 67.

Figura 67 – Atividades de um controle de mudança

Caso exista o comitê de gestão de mudança, ele é responsável pela análise, pela aprovação ou não da mudança, pela priorização da implementação e pelo controle do escopo da alteração. Caso não exista um comitê, deve haver um representante do usuário e outro representante da construção para realizar esse papel.

A mudança ocorre a partir da identificação de uma necessidade pelos usuários ou por qualquer outro envolvido no projeto.

Essa solicitação é conhecida como pedido de mudança ou *change request*.

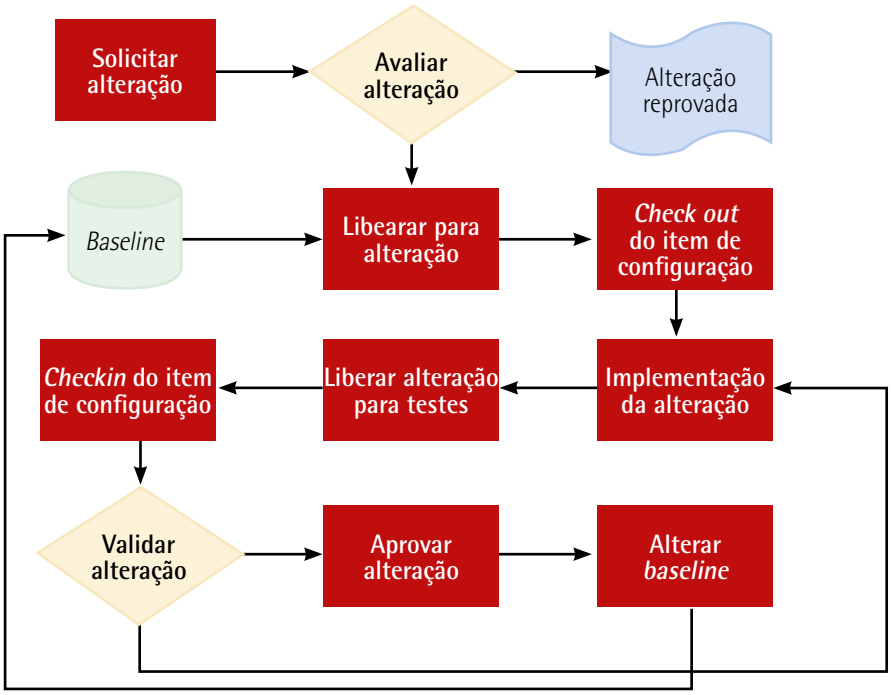

Figura 68 – Atividades de mudança dentro do GCS

Conforme apresentado na Figura 68, ao ser recebida, a mudança precisa ser avaliada quanto à sua necessidade. Caso aprovada, o item de configuração é retirado da *baseline* por meio de uma ação de *check out*, a qual permite que o item seja modificado pelo desenvolvedor. Após a alteração do artefato, é feito o registro com a descrição da modificação, quem alterou e quando alterou, e é realizado o *ckeck-in* do item. Este item é submetido à validação do solicitante do pedido de alteração. Uma vez validado, uma nova *tag (label)* e uma nova *baseline* são geradas para o item de configuração.

Todo o pessoal envolvido no projeto ou na manutenção dos itens sob gerência da configuração deve ser treinado no processo e nas ferramentas, a fim de garantir sua correta utilização.

8.2.4 Auditoria de configuração

A auditoria de configuração tem como objetivo garantir que a mudança esteja correta, que o processo de gestão da configuração esteja sendo seguido corretamente e que o uso da ferramenta também esteja correto.

As auditorias devem ser parte do processo de gestão da configuração e ser realizadas por pessoal externo ao projeto, com o objetivo de garantir a isenção dos apontamentos realizados. A verificação de que a mudança está correta é feita em RTF. Esta tem foco na correção técnica do item de configuração e deve ser realizada antes da atividade de validação da mudança pelos envolvidos.

A auditoria tem foco na verificação de que os procedimentos descritos no processo de gestão de configuração estejam sendo realizados por todos os envolvidos no trabalho. Os seguintes questionamentos precisam ser respondidos em uma auditoria da configuração:

- Os itens de configuração estão nomeados de acordo com o padrão definido?
- A estrutura de diretórios está de acordo com o padrão?
- Os itens de configuração estão colocados corretamente em suas pastas de armazenamento?
- O controle de acesso aos artefatos está implementado?
- Existe um registro de revisão da mudança?
- Histórico e comentários da mudança, autor e data estão registrados?

8.2.5 Registro do *status*

O registro de *status* consiste em registrar nos itens de configuração as informações relacionadas à mudança efetuada, quem a fez, a data da mudança e a informação de outros itens de configuração que podem ser afetados.

8.3 Padrões de gestão da configuração

Existem diversos padrões de gestão de configuração já definidos por modelos de qualidade e processos ISO para que uma organização possa escolher, de acordo com as suas próprias necessidades ou com o modelo de qualidade que utiliza. Os principais padrões de gestão da configuração sugeridos são os propostos pelo CMMI, pela ISO/IEC 12207, pela ISO/IEC 9000-3 e pela ISO/IEC 15504, apresentados na Figura 69.

Figura 69 – Modelos e processos que definem padrões de GCS

8.3.1 Padrão de gestão de configuração – CMMI

A gestão da configuração no modelo de qualidade CMMI está previsto no Nível 2 – Gerenciado e é parte dos requisitos básicos de maturidade no desenvolvimento de *software*.

Segundo o CMMI (SEI, 2006), o objetivo do gerenciamento de configuração é manter a integridade dos produtos de trabalho, utilizando a identificação, o controle, a comunicação do *status* e auditorias das configurações, de acordo com a Figura 70.

Figura 70 – Gestão de configuração segundo o CMMI

Essa área de processo envolve as seguintes atividades:

- selecionar os artefatos que compõem a configuração e as *baselines* do *software*;
- controlar as mudanças nos itens de configuração;
- manter a integridade das *baselines*;
- fornecer a situação precisa dos itens de configurações para desenvolvedores, usuários finais e clientes.

O CMMI descreve a área de processo da gerência da configuração em metas genéricas e específicas, de acordo com a Tabela 12.

Tabela 12 – Práticas do Modelo CMMI para a GCS

Práticas genéricas	Práticas específicas
Estabelecer as *baselines*	• Identificar os itens de configuração • Estabelecer um sistema de gestão de configuração • Criar ou liberar as *baselines*
Rastrear e controlar as mudanças	• Rastrear as solicitações de mudança • Controlar os itens de configuração
Estabelecer a integridade	• Estabelecer registros de configuração • Rastrear as solicitações de mudança • Executar auditorias da configuração

Fonte: CMMI (2014).

As **práticas genéricas** compõem as atividades de estabelecer *baselines*, rastrear as mudanças e estabelecer a integridade dos itens de configuração.

As **práticas específicas** descrevem as tarefas que definem o que precisa ser realizado para que as práticas genéricas atinjam a meta.

Para cada prática específica há um conjunto de atividades detalhadas denominadas de **subpráticas**, que têm o objetivo de atingir as metas específicas, com sugestão de produtos de trabalho e descrição de processos. Lembrando que o CMMI não define como você deve realizar a gestão de configuração. São descritas as atividades necessárias para definir o que deve ser feito para se atingir o nível de maturidade esperado pela equipe e pela organização envolvida.

8.3.1.1 Estabelecer *baselines*

Consiste no conjunto de tarefas para determinar quais são os itens de configuração que devem ser controlados. Envolve as seguintes atividades:

- **Identificar itens de configurações**
  - selecionar os itens de configurações a partir dos artefatos previstos no projeto;
  - atribuir identificadores únicos para os itens de configurações;
  - especificar quando cada item de configuração será colocado sob GCS;
  - identificar o proprietário responsável por cada item de configuração.
- **Estabelecer um sistema de gerenciamento de configurações**
  - estabelecer um mecanismo para armazenar e recuperar itens de configurações no sistema de gerenciamento de configurações;
  - armazenar e recuperar versões arquivadas de itens de configurações;
  - armazenar, atualizar e recuperar registros de gerenciamento de configurações;
  - criar relatórios de gerenciamento de configurações a partir do sistema de gerenciamento de configurações.
- **Criar ou liberar *baselines***
  - obter autorização do comitê ou do cliente antes de criar ou liberar *baselines* de um item de configuração;
  - somente utilizar artefatos recuperados do sistema de gerenciamento de configurações;
  - manter os itens de configuração disponíveis.

8.3.1.2 Rastrear e controlar mudanças

Consiste no conjunto de tarefas para controlar e rastrear mudanças nos itens de configuração definidos. Envolve as seguintes atividades:

- **Rastrear solicitações de mudanças**
  - iniciar e registrar as solicitações de mudanças;
  - analisar o impacto das mudanças e as correções propostas nas solicitações de mudanças;
  - revisar as solicitações de mudanças que serão tratadas na próxima *baseline* com os que serão afetados pelas mudanças e obter a sua concordância;
  - rastrear os *status* das solicitações de mudanças até o encerramento.
- **Controlar itens de configurações**
  - controlar mudanças nos itens de configuração durante toda a vida do produto;
  - obter a autorização apropriada antes que os itens de configurações modificados entrem no sistema de gerenciamento de configurações;
  - executar o *check-in* e o *check out* dos itens de configurações de uma maneira que mantenha a correção e a integridade dos itens de configuração;
  - registrar as mudanças nos itens de configurações e os motivos das mudanças.

8.3.1.3 Estabelecer a integridade

Consiste no conjunto de tarefas para controlar e rastrear mudanças nos itens de configuração definidos. Envolve as seguintes atividades:

- **Estabelecer registros de gerenciamento de configurações**
  - registrar com detalhes o *status* de cada item de configuração, de forma que cada um seja conhecido e versões anteriores possam ser recuperadas;
  - assegurar que todos os envolvidos tenham acesso ao *status* dos itens de configuração e conhecimento destes;
  - especificar e identificar a versão mais atual das *baselines*.

- **Executar auditorias de configurações**

  – analisar a integridade das baselines;

  – confirmar que os registros de configurações identificam corretamente a configuração dos itens de configuração;

  – confirmar a conformidade com os padrões e procedimentos de gerenciamento de configurações aplicáveis;

  – rastrear os itens de ação da auditoria até o encerramento.

8.3.2 Padrão de gestão de configuração – ISO/IEC 12207

A norma ISO/IEC 12207 descreve o processo de gestão da configuração na parte referente aos processos de apoio e se baseia na aplicação de tarefas no ciclo de desenvolvimento de *software* para identificar e definir os itens de configuração, estabelecer suas *baselines*, controlar as mudanças e registrar a situação dos itens durante o processo. Resumindo, a ISO/IEC 12207 descreve os seguintes processos para a gerência de configuração, ilustrados na Figura 71:

- implementação do processo;
- identificação da configuração;
- controle da configuração;
- relato da situação da configuração;
- avaliação da configuração;
- gerência de liberação e distribuição.

Figura 71 – Gestão de configuração segundo o ISO/IEC 12207

8.3.2.1 Implementação do processo

Descreve as atividades que são realizadas durante o gerenciamento da configuração, formando o plano de gerenciamento de configuração, cujo objetivo é padronizar os procedimentos e as ferramentas para todas as equipes de desenvolvimento da organização.

O plano de GCS também inclui o seu relacionamento com o plano de desenvolvimento de *software* e o plano de manutenção do *software*.

8.3.2.2 Identificação da configuração

A partir dos artefatos previstos no plano de desenvolvimento do *software*, os itens que têm de estar sob a gestão da configuração devem ser selecionados e nomeados, bem como estabelecidas versões para o controle das *baselines* de configuração.

A lista de itens sob a gestão da configuração deve ser aprovada pelos envolvidos no projeto.

8.3.2.3 Controle da configuração

Detalha o conjunto de atividades para identificação e registro dos pedidos de mudanças, implementação, verificação e liberação do item de *software* modificado. Nessa tarefa, são realizadas análise, classificação, priorização e avaliação das mudanças, além de aprovação ou rejeição do pedido de mudança e controle de acesso aos itens de configuração.

Devem existir registros de todas as alterações executadas para permitir uma eventual necessidade de rastreabilidade.

8.3.2.4 Avaliação da configuração

Descreve um conjunto de procedimentos para garantir que a implementação das mudanças esteja de acordo com as especificações e que os itens de configuração estejam adequadamente atualizados.

8.3.2.5 Relato da situação da configuração

Processo que prepara e exibe os registros da situação atual e do histórico dos itens de configuração do *software*. Os relatórios de situação podem conter, para cada item, suas últimas versões, o número de alterações, identificadores de liberação, a quantidade de liberações, dentre outros.

8.3.2.6 Gerência de liberação e distribuição

Na liberação e na distribuição dos itens de configuração, a documentação deve estar atualizada. O código e a documentação devem estar armazenados, empacotados e distribuídos de acordo com os procedimentos estabelecidos no plano de gerenciamento da configuração.

8.3.3 Padrão de gestão de configuração – ISO/IEC 9000-3

A ISO/IEC 9000-3 é a norma específica de organizações de desenvolvimento de *software* e a mais utilizada na implantação do SGQ. Segundo a descrição da norma, a gestão da configuração é aplicada durante o ciclo de vida de um produto, a fim de fornecer visibilidade e controle de suas funcionalidades e características físicas. Na norma ISO/IEC 9000-3, o gerenciamento de configuração

está descrito em três partes: identificação, controle e situação dos itens sob a gestão de configuração, ilustradas na Figura 72.

Figura 72 – Gestão de configuração segundo o ISO/IEC 9000-3

Saiba mais

Uma norma específica para o gerenciamento da configuração foi criada pela ISO: é a norma ISO/IEC 10007:95, que fornece diretrizes para gestão da configuração e pode ser acessada em <www.iso.org>.

Segundo o modelo, a organização que produz o *software* deve desenvolver e criar um plano de gestão que inclua as atividades de gestão de configuração, ferramentas, técnicas, metodologias usadas e o estado no qual os itens de configuração devem ser submetidos ao controle da configuração.

8.3.3.1 Identificação e rastreabilidade de configuração

A organização deve manter procedimentos para identificar os artefatos de *software*, durante todas as fases do ciclo de vida, nos quais devem ser aplicadas atividades para assegurar que cada item seja identificado, incluindo: especificações funcionais e técnicas, ferramentas de desenvolvimento usadas no projeto, todas as interfaces com outros itens de *software* e *hardware* e todos os documentos produzidos durante o processo de desenvolvimento do *software*.

8.3.3.2 Controle de alterações

A organização deve estabelecer e manter procedimentos para identificar, documentar, analisar criticamente e autorizar quaisquer alterações em itens de configuração. Para isso, antes da aceitação de uma alteração, uma análise de impacto da alteração verifica sua validade e os possíveis efeitos causados sobre outros itens de configuração. Durante o controle de alterações, a atividade de comunicação aos envolvidos quanto às alterações e para demonstrar a rastreabilidade das alterações precisa ser mantida.

8.3.3.3 Relatório de situação da configuração

A organização estabelece e mantém procedimentos para registrar, administrar e relatar a situação de itens de configuração, quanto a alterações, versões, requisições e implementação das alterações aprovadas.

8.3.4 Padrão de gestão de configuração – ISO/IEC 15504 (Spice)

Na norma ISO/IEC 15504, o processo de gerenciamento da configuração está no Nível 2, como no CMMI, e é considerado uma atividade de suporte aos demais processos do modelo de qualidade.

Figura 73 – Gestão de configuração segundo o ISO/IEC 15504

A norma estabelece que o processo de gerência da configuração tem como objetivo criar e manter a integridade de todos os produtos que são elaborados e resultam do ciclo de vida de desenvolvimento do *software*. Como nos demais padrões, nem todos os produtos são controlados, sendo necessário que o conjunto de artefatos que precisam ter o controle do sistema de gerenciamento da configuração seja definido pela equipe do projeto e pelo cliente.

8.3.4.1 Estabelecer a biblioteca de gestão da configuração

Criar um repositório de *software* com controle de acesso para o armazenamento e a recuperação de itens sob a gestão da configuração, bem como para o compartilhamento dos itens entre as diversas equipes envolvidas no processo.

8.3.4.2 Identificar itens da configuração

Identificar cada artefato produzido no projeto de *software* e selecionar aqueles que ficarão sob a gerência da configuração.

8.3.4.3 Manter descrições dos itens da configuração

Fornecer e manter descrições para cada item de configuração, identificando-o com sua descrição de função, o responsável pelo item e a partir de quando está sob o processo de gerência da configuração.

8.3.4.4 Administrar pedidos de mudança

Gravar, revisar, aprovar e rastrear todas as requisições de mudanças e todos os relatórios de problemas para todos os itens da configuração e suas versões.

8.3.4.5 Controlar as mudanças

Estabelecer um controle de acesso para manter a clareza e a integridade dos itens da configuração gerenciados pela biblioteca de configuração.

8.3.4.6 Construir versões do produto

As versões dos itens de configuração (*baselines*) somente podem ser geradas a partir da biblioteca de configuração e depois de formalmente autorizado.

8.3.4.7 Manter históricos dos itens da configuração

Manter um histórico de cada item da configuração e das ações realizadas em um nível de detalhe suficiente para permitir que possam ser feitas recuperações de versões anteriores.

8.3.4.8 Relatar o estado da configuração

Reportar os resultados das atividades realizadas em cada item da configuração. Genericamente, os padrões descrevem um núcleo comum para a gestão da configuração relacionado a identificação, controle de versões e mudanças dos itens sob gerência da configuração. Exceção feita aos processos de segurança e auditoria não abrangidos por todos os padrões. O Quadro 24 apresenta um resumo dos padrões descritos.

Quadro 24 – Comparativo entre os padrões de gestão de configuração

Atividade de GCS	CMMI	ISO/IEC 12207	ISO/IEC 9000-3	ISO/IEC 15504
Identificar item de configuração	A identificação dos artefatos de *software* que devem ser controlados, a partir da relação produzida no projeto	Além da identificação do item, descreve a criação do processo de GCS	Identifica os artefatos que devem ser controlados pelo GCS	Além da identificação dos itens, descreve a criação da biblioteca para o repositório da GCS
Controlar as versões	Os itens de configuração devem ser descritos, e suas *baselines*, mantidas mediante aprovação	Não descrito explicitamente na norma	Não descrito explicitamente na norma	Construir versões dos itens de configuração apenas a partir da biblioteca de itens
Gerenciar mudanças	Criar um processo formal para analisar, classificar e aprovar as mudanças identificadas nos itens de configuração	Identificar os registros dos pedidos de mudanças, análise, classificação, priorização e avaliação das mudanças, além da aprovação ou rejeição do pedido de mudança	Fazer uma análise de impacto da alteração para verificar sua validade e os possíveis efeitos causados sobre outros itens de configuração	Gravar, revisar, aprovar e rastrear as mudanças e garantir que estas sejam feitas por pessoal autorizado e aprovadas ao seu final
Auditoria de configuração	São realizadas regularmente para verificar aderência ao processo de GCS	Não estão descritas no processo	Não estão descritas no processo	Não estão descritas no processo
Registro de *status* da configuração	Todas as mudanças devem ter suas informações de descrição, de quem alterou e quando mantidas em histórico	Descrever a situação de cada item contendo: o número de alterações, identificadores de liberação, a quantidade de liberações, dentre outras	Relatar a situação dos itens quanto às alterações, versões, requisições e à implementação das alterações aprovadas	Manter um histórico dos itens com um nível de detalhe suficiente para permitir a recuperação de versões anteriores

A gestão da configuração é um processo essencial para uma gestão controlada e madura dos artefatos produzidos durante o ciclo de desenvolvimento de um *software*. As organizações de

Tecnologia da Informação devem possuir esse processo implantado para iniciar qualquer modelo de qualidade de *software*.

O processo de GCS é simples e barato, pois as ferramentas não possuem custos de aquisição de licença. A definição do processo e o treinamento das equipes constituem a parte mais demorada do gerenciamento da configuração.

Alguns benefícios que uma organização pode alcançar com a implantação do GCS são:

- ganhos em maturidade no desenvolvimento de *software*;
- maior capacidade de executar projetos mais complexos;
- capacidade de retornar a versões anteriores do *software*;
- melhoria da comunicação dos desenvolvedores;
- melhoria da competitividade;
- aumento da qualidade.

Saiba mais

O IEEE também possui um padrão para gestão de configuração disponível em <www.ieee.org>, descritos nas normas IEEE 1042/1987 e IEEE 828/1998.

8.4 *Application Lifecycle Management* (ALM)

O contexto abordado até o momento sobre gestão de mudanças e gestão da configuração tem ligação direta com o conceito de gerência do ciclo de vida das aplicações, ou *Application Lifecycle Management* (ALM).

ALM é a gestão integrada entre as necessidades de atender ao negócio e aos processos de Engenharia de *Software*, por meio da criação de procedimentos e do suporte de ferramentas que fazem a automação dos processos. Permite o gerenciamento contínuo do processo de desenvolvimento de *software*, deixando clara a interligação de requisitos, manutenção, mudança, configuração e distribuição. Para isso, compõe-se de cinco processos:

- gerenciamento de requisitos;
- gerenciamento de configuração;
- gestão de mudança;

- versionamento e integração;
- distribuição de *software*.

8.4.1 Gerenciamento de requisitos

Os requisitos de um *software* são as necessidades dos usuários que têm de ser atendidas por um novo *software* ou um pedido de mudança para ser incluída em uma aplicação já existente. O gerenciamento de requisitos é composto por um conjunto de atividades para levantar as necessidades dos usuários, entender essas necessidades, documentá-las, validar com os usuários e conseguir rastrear todo e qualquer requisito ao longo do ciclo de vida do *software*. As principais atividades envolvidas na gestão de requisitos são:

- documentar os requisitos funcionais e os não funcionais;
- identificar, analisar e selecionar as necessidades de negócio;
- priorizar o que deve ser feito;
- analisar a interdependência dos requisitos;
- relacionar com os requisitos de arquitetura;
- validar os requisitos com os usuários.

8.4.2 Gerenciamento da configuração

O processo de construção de uma aplicação envolve o controle de diversos artefatos de *software*, os quais podem sofrer diversas modificações durante seu ciclo de vida. Como vimos, o GCS é responsável por manter e gerenciar esses artefatos e por manter a rastreabilidade e o versionamento desses itens. Muitas pessoas confundem ALM com gerenciamento da configuração, mas o GCS isolado no contexto de desenvolvimento de *software* não garante o processo de ALM completo.

8.4.3 Gestão de mudança

Processo para avaliar, coordenar e decidir sobre a realização de mudanças em itens de configuração. Possui relacionamento direto com o gerenciamento da configuração. As mudanças aprovadas são implementadas em todos os artefatos relacionados.

8.4.4 Versionamento e integração

Consiste nas atividades de reunir os diversos elementos que compõem uma versão de *software* e agrupá-las em um único pacote, a fim de ser testado e distribuído dentro do ambiente do projeto. Para esse processo, temos:

- recuperar todos os artefatos do repositório de código;
- mapear os artefatos com a nova versão da aplicação;
- compilar o código;
- organizar a aplicação, conforme definido;
- criar um pacote de instalação.

8.4.5 Distribuição do *software*

Tem como objetivo garantir a consistência das diversas versões de uma aplicação, pois há várias versões durante o ciclo de vida de um *software*. Sem um processo de GCS adequado, corre-se alto risco com a perda da integridade da aplicação, gerando grande retrabalho e gastos para a correta distribuição. Para o correto funcionamento da distribuição do *software*, é essencial um processo de GCS consistente e maduro. Algumas atividades de distribuição:

- identificar o pacote correto de versionamento;
- documentar as dependências entre as versões;
- automatizar a distribuição das versões ou correções;
- distribuir os pacotes de *software* para validação, testes e produção.

Resumo

Nesta unidade, abordamos que a manutenção de *software* é uma atividade essencial para que a aplicação desenvolvida tenha um longo ciclo de vida. No entanto, a manutenção depende da estrutura da aplicação desenvolvida, ou seja, documentação, padrão de codificação e foco em criar um *software* modularizado para facilitar o processo de desenvolvimento futuro e reduzir os custos das mudanças que podem atingir até 60% do custo original.

Vimos que essas manutenções surgem a partir da evolução das necessidades de negócio para atender à evolução das regras ou para alinhar a competição no mercado, de ser uma demanda baseada na exigência da lei ou da necessidade de atender a uma norma solicitada pelo cliente, para corrigir defeitos que surgem ao longo da vida útil de um sistema e para atender à evolução tecnológica de *hardware* e *software*.

Aprendemos também que a norma ISO/IEC 12207 descreve um procedimento para manutenção na engenharia dos seus processos fundamentais, porém a norma ISO/IEC 14764 (2006) é específica para

manutenção e descreve conceitos e um processo a ser aplicado para os diversos tipos de manutenção. Esses tipos podem ser: manutenções corretivas, que se referem à correção de defeitos que ocorrem após o *software* entrar em produção; manutenções perfectivas, relacionadas às novas funcionalidades solicitadas pelos usuários e também denominadas de evolutivas ou de melhoria; manutenções adaptativas, voltadas para atender as demandas técnicas da evolução do *software* no ambiente computacional; e manutenções preventivas, que são análises prévias para tentar identificar defeitos antes que ocorram em produção.

Vimos diversos conceitos intrínsecos ao processo de manutenção, como: evolução, referente a mudanças que incluem comportamentos na aplicação; alteração, que modifica algo já existente; correção, para tratar defeitos; melhoramento, referente a melhorias técnicas; e manutenibilidade, que é um requisito não funcional referente à facilidade com a qual um *software* pode sofrer uma mudança.

Abordamos que, além dos tipos de manutenção apresentados, a própria engenharia reversa também pode ser tratada como tal, uma vez que sua função é criar uma nova aplicação a partir de uma existente, utilizando o código-fonte como a principal referência de requisitos. Ocorre quando a manutenção de um sistema começa a se tornar inviável em termos técnicos (sistema obsoleto) ou financeiros (manutenções demoradas e difíceis, e recursos especializados escassos).

Vimos ainda outro tipo de manutenção que onera bastante os custos: a manutenção de sistemas legados, que são aplicações muito antigas, das quais não há documentação, nem pessoas que conheçam o código. Aliada a isso, há a resistência dos desenvolvedores mais experientes em manter esses *softwares*, que acabam nas mãos de desenvolvedores menos experientes e tornam-se um fator de desmotivação, constituindo um desafio para as equipes de manutenção.

Abordamos que, em relação à equipe de manutenção, é importante que se tenha os papéis claramente definidos quanto às suas responsabilidades, para que o processo de manutenção sofra menos impacto e conflitos sejam evitados. Os principais envolvidos são: os usuários finais, que são aqueles responsáveis pelas solicitações de correções ou de melhoria e que também vão dar o aceite nas mudanças; os responsáveis pelo produto, aqueles que dão suporte financeiro para que a aplicação sofra as mudanças; a gerência de Tecnologia da Informação, que cuida de receber as solicitações dos usuários e gerar a demanda para a gerência de manutenção, que é responsável por realizar análise, classificação, priorização e implementação das mudanças junto com a equipe de manutenção.

Estudamos também que outro fator que afeta os custos das manutenções são as condições iniciais do processo, podendo-se classificar as manutenções em estruturada e não estruturada. As manutenções não estruturadas se referem àquelas aplicações cuja única documentação é o código-fonte, tornando o esforço maior, de alto risco e alta complexidade. As manutenções estruturadas se dão em *softwares* desenvolvidos com documentação, padrões e preocupação com a qualidade, conferindo-lhes um alto grau de manutenibilidade.

Vimos que a norma ISO/IEC 14764 (2006) define cinco atividades básicas para um processo de manutenção: a definição do processo, a análise da mudança, a implementação da mudança, o aceite e a revisão da alteração, e o processo de migração e de retirada do *software* de produção.

Abordamos ainda que a definição do processo, junto com a definição dos papéis e das responsabilidades, estabelece qual é o fluxo de um pedido de mudança realizado por um usuário. Basicamente define que, após receber o pedido de mudança, este seja avaliado quanto a novas funções ou correção. Em se tratando de correção, deve-se avaliar a gravidade. Se for grave, implementar imediatamente; caso contrário, fazer a avaliação e a classificação da mudança como para as solicitações de novas funcionalidades. Após isso, obtém-se a aprovação dos usuários e priorizados na fila de manutenção. As mudanças não aprovadas devem ser documentadas e armazenadas.

Vimos que a análise da mudança consiste na avaliação profunda da solicitação, bem como de seus riscos e seus impactos na aplicação existente. Avaliam-se prazo e custos e submete-se à aprovação dos usuários. O processo de implementação da mudança deve seguir os procedimentos de desenvolvimento utilizados pela organização.

Abordamos também que, após a codificação e os testes das mudanças, estas deverão ser submetidas à aceitação dos usuários quanto à qualidade e ao preenchimento dos requisitos solicitados. Envolvem também as ações de garantia da qualidade, como revisões, inspeções e testes.

Vimos ainda que o processo de migração refere-se à elaboração de um procedimento para que os dados de um *software* em produção sejam levados para outro ambiente ou para serem utilizados em outra aplicação. Esse processo deve ser formalizado por meio de um plano de migração que descreva detalhadamente o que será migrado, o **de-para** dos campos e as regras de conversão, que precisam do aceite dos usuários.

Abordamos também que o processo de retirada de produção está relacionado à descontinuidade da aplicação por alcançar o fim do seu

ciclo de vida, seja pela inviabilidade técnica ou econômica, seja pela sua substituição por um novo *software*. Também exige um plano formal validado pelos usuários descrevendo o que e como o processo será realizado, principalmente, o paralelismo e a transição das aplicações.

Como visto, as manutenções são diretamente afetadas pela forma que foram desenvolvidas. Os desenvolvimentos orientados a objetos e baseados em componentes são os que causam menos impactos, mas não estão livres de problemas. Ambos se baseiam na característica de reúso de código, que minimiza os testes, mas, em caso de modificação, geram um esforço maior para garantir que tudo o que os utiliza continue funcionando adequadamente.

Vimos também que outro impacto a analisar é o relacionado ao processo de desenvolvimento ágil que, em razão das suas características, facilita a interação de usuário e desenvolvedor durante o ciclo de manutenção. Porém, as vantagens e desvantagens são as mesmas do processo de desenvolvimento utilizado, seja este orientado a objetos ou baseado em componentes.

Aprendemos que o Gerenciamento de Configuração de *Software* (GCS), também conhecido como *Software Configuration Management* (SCM), é um processo da engenharia de *software* que tem por objetivo estabelecer as condições para a organização dos artefatos de *software*, controlar suas versões, controlar as suas alterações e manter a integridade de tudo o que foi produzido durante o ciclo de vida de um projeto de *software*, bem como garantir seu armazenamento adequado.

Estudamos que, para realizar um processo de GCS eficiente, é preciso definir um repositório para o armazenamento dos artefatos, a estrutura de diretórios, a nomenclatura de cada um dos artefatos envolvidos no processo de gestão da configuração e o controle de acesso aos artefatos. O próximo passo consiste em identificar, a partir do conjunto de artefatos, quais devem ter o controle, além de serem definidos como itens de configuração. Esses itens estão submetidos a um controle rigoroso das versões relacionadas às mudanças que podem sofrer ao longo do ciclo de vida, mediante um processo de análise, classificação, priorização e validação das alterações. Todas essas mudanças devem ser rastreadas a qualquer momento para exibir a situação do artefato. Um processo de auditoria dos itens de configuração deve estar estabelecido para a verificação e a garantia de aderência aos padrões de GCS.

Vimos que esse é o processo básico da GCS, mas existem padrões que descrevem detalhadamente o processo de gestão da configuração, como o do modelo CMMI, a ISO/IEC 12207, a ISO/IEC 9000-3 e a ISO/IEC 15504.

Exercícios

Questão 1. Segundo Boehm (1988), no ciclo de vida de um *software* a fase de manutenção é a mais problemática e representa mais de 60% de todo o esforço de TI de uma organização. A norma ISO/IEC 14764 (2006) é específica para descrever o ciclo de manutenção do *software*, definindo os conceitos envolvidos e estabelecendo um processo claro para tratar e apoiar os desenvolvedores nessa complexa etapa. A norma citada especifica quatro tipos distintos de manutenção e em apenas uma descreve como corrigir defeitos encontrados em um *software* em produção, defeitos normalmente identificados pelos usuários finais. Qual seria esse tipo de manutenção, descrito na norma ISO/IEC 14764? Assinale a alternativa correta.

A) Manutenção perfectiva.

B) Manutenção adaptativa.

C) Manutenção corretiva.

D) Manutenção preventiva.

E) Manutenção perfectiva e adaptativa.

Resposta correta: alternativa C.

Análise das alternativas

A) Alternativa incorreta.

Justificativa: não corrige defeitos de *software* em produção. A manutenção perfectiva realiza ações para incluir novas funcionalidade ou realizar alterações cujos objetivos sejam satisfazer novas necessidades dos usuários.

B) Alternativa incorreta.

Justificativa: não corrige defeitos de *software* em produção, muito menos defeitos encontrados pelo usuário final. A manutenção adaptativa realiza processos para adequar o *software* a novas tecnologias, modelos de gestão etc., isto é, de uma nova legislação.

C) Alternativa correta.

Justificativa: a manutenção corretiva executa ações para corrigir defeitos encontrados em um *software* em produção, normalmente observados pelo usuário final. Na maioria dos casos, essas manutenções possuem uma equipe dedicada que realiza atividades rotineiras para a solução de defeitos.

D) Alternativa incorreta.

Justificativa: não corrige defeitos de *software* em produção. A manutenção preventiva é proativa e planejada, realiza processos para melhorar algum aspecto deficiente no *software*, como desempenho, segurança, defeitos ainda não identificados pelos usuários, usabilidade etc.

E) Alternativa incorreta.

Justificativa: tanto em conjunto como individualmente, essas manutenções não corrigem defeitos de *software* em produção. Quando necessário os tipos de manutenção podem ser aplicados de forma sequencial, e não de forma simultânea.

Questão 2. A norma ISO/IEC 14764 (2006) define cinco atividades básicas para um processo de manutenção: a definição do processo, a análise da mudança, a implementação da mudança, o aceite e revisão da alteração, o processo de migração e de retirada de produção do *software*. Em relação ao processo de migração, assinale a alternativa correta.

I – Procedimento elaborado para avaliação profunda da solicitação, seus riscos e seus impactos na aplicação existente.

II – Procedimento elaborado para que os dados de um *software* em produção sejam levados para outro ambiente.

III – Procedimento realizado para que os dados de um *software* em produção sejam utilizados em outra aplicação. Esse procedimento deve ser formalizado através de um detalhamento do que será migrado.

IV – Procedimento que está relacionado à descontinuidade da aplicação por alcançar o fim do seu ciclo de vida, seja pela inviabilidade técnica ou econômica, seja pela sua substituição por um novo *software*.

É correto apenas o que se destaca em:

A) I, II.

B) I, II, III, IV.

C) II, III.

D) III.

E) II.

Resposta desta questão na plataforma.

FIGURAS E ILUSTRAÇÕES

Figura 5

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *NBR ISO 9001*: Sistemas de gestão da qualidade – Requisitos. Rio de Janeiro, 2000b. p. 2.

Figura 9

McCALL, J.; RICHARDS, P. K.; WALTERS, G. F. Factors in Software Quality: Concept and Definitions of Software Quality. *RADC_TR-77-369*, v. 1, Nov. 1977. Final Technical Report. Adaptado.

Figura 10

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). *ISO/IEC 9126-1*: International Standard. Information Technology – Software engineering – Product quality – Part 1: Quality model. Genebra, 2001. p. 7.

Figura 11

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). *ISO/IEC 9126-1*: International Standard. Information Technology – Software engineering – Product quality – Part 1: Quality model. Genebra, 2001. p. 1. Adaptado.

Figura 17

COUTO, A. B. *CMMI*: integração dos modelos de capacitação e maturidade de sistemas. Rio de Janeiro: Ciência Moderna, 2007. P. 47. Adaptado.

Figura 50

CRISPIN, L; GREGORY, J. *Agile Testing*: A Practical Guide for Testers and Agile Teams. New York: Addison Wesley, 2009. Adaptado.

Figura 51

BECK, K. *TDD*: desenvolvimento guiado por testes. São Paulo: Bookman, 2010. Adaptado.

REFERÊNCIAS

ANTONIONI, J. A. *Qualidade em software*: manual de aplicação da ISO-9000. São Paulo: Makron Books, 1995.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *NBR ISO 9000*: Sistemas de gestão da qualidade – Fundamentos e vocabulário. Rio de Janeiro, 2000a.

___. *NBR ISO 9001*: Sistemas de gestão da qualidade – Requisitos. Rio de Janeiro, 2000b.

ASSOCIAÇÃO BRASILEIRA PARA PROMOÇÃO DA EXCELÊNCIA DO *SOFTWARE* BRASILEIRO (SOFTEX). *MPS.BR – Melhoria de processo do software brasileiro*: guia geral – Versão 1.1. 2006.

BECK, K. *TDD*: desenvolvimento guiado por testes. São Paulo: Bookman, 2010.

BOEHM, B. Software Engineering. *IEEE Trans Computers*, Dec. 1976, p. 1226-41.

___. A Spiral Model of Software. *Development and Enhancement*, IEEE Computer, 1988.

CHIKOFSKY, E. J.; CROSS II, J. H. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, v. 7, n. 1, p. 13-7, 1990.

CORTÊS, M. *et al. Modelos de qualidade de software*. Campinas: Unicamp, 2001.

COUTO, A. B. *CMMI*: integração dos modelos de capacitação e maturidade de sistemas. Rio de Janeiro: Ciência Moderna, 2007.

CRISPIN, L.; GREGORY, J. *Agile Testing*: A Practical Guide for Testers and Agile Teams. New York: Addison Wesley, 2009.

CROSBY, P. B. *Qualidade, falando sério*. São Paulo: McGraw-Hill, 1990.

DIJKSTRA, E. W. *Notes on Structured Programming*. Circulated privately, Apr. 1970.

GUERRA A. C.; COLOMBO, R. M. T. *Tecnologia da informação*: qualidade de produto de software. Brasília: PBQP Software, 2009.

HETZEL, W.C. *The Growth of Software Testing*. Englewood: Prentice-Hall, 1988.

HUMPHREY, W. *Managing the Software Process*. New York: Addison Wesley, 1989.

IEEE. *IEEE Std 610.12* – Glossário de padrão de engenharia de software. 1990.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). *ISO 10007*: Quality management– Guidelines for configuration management. Genebra, 1995.

___. *ISO/IEC 9000-3*: International Standard. Information Technology – Guidelines for the application of ISO 9001:2000 to computer software. Genebra, 2004.

___. *ISO/IEC 9126-1*: International Standard. Information Technology – Software engineering – Product quality – Part 1: Quality model. Genebra, 2001.

___. *ISO/IEC 9126-2*: International Standard. Information Technology – Software engineering – Product quality – Part 2: External metrics. Genebra, 2003a.

___. *ISO/IEC 9126-3*: International Standard. Information Technology – Software engineering – Product quality – Part 3: Internal metrics. Genebra, 2003b.

___. *ISO/IEC 9126-4*: International Standard. Information Technology – Software engineering – Product quality – Part 4: Quality in use metrics. Genebra, 2004.

___. *ISO/IEC 12207*: International Standard. Information Technology – Systems and software engineering – Software life cycle processes. Genebra, 2008.

___. *ISO/IEC 14598-1*: International Standard. Information Technology – Software Product Evaluation – Part 1: General Overview. Genebra, 1999a.

___. *ISO/IEC 14598-2*: International Standard. Information Technology – Software Product Evaluation – Part 2: Planning and Management. Genebra, 2000.

___. *ISO/IEC 14598-3*: International Standard. Information Technology – Software Product Evaluation – Part 3: Process for Developers. Genebra, 2000.

___. *ISO/IEC 14598-4*: International Standard. Information Technology – Software Product Evaluation – Part 4: Process for Acquirers. Genebra, 1999b.

___. *ISO/IEC 14598-5*: International Standard. Information Technology – Software Product Evaluation – Part 5: Process for Evaluators. Genebra, 1998.

___. *ISO/IEC 14598-6*: International Standard. Information Technology – Software Product Evaluation – Part 6: Evaluation Modules. Genebra, 2001.

___. *ISO/IEC 14764*: Software Engineering – Software Life Cycle Processes – Maintenance. 2. ed. Genebra, 2006.

___. *ISO/IEC 15504-1 (Spice)*: International Standard. Information Technology – Process assessment – Part 1: Concepts and vocabulary. Genebra, 2004.

___. *ISO/IEC 15504-2 (Spice)*: International Standard. Information Technology – Process assessment – Part 2: Performing an assessment. Genebra, 2003c.

___. *ISO/IEC 15504-3 (Spice)*: International Standard. Information Technology – Process assessment – Part 3: Guidance on performing an assessment. Genebra, 2004.

___. *ISO/IEC 15504-4 (Spice)*: International Standard. Information Technology – Process assessment – Part 4: Guidance on use for process improvement and process capability determination. Genebra, 2004.

___. *ISO/IEC 15504-5 (Spice)*: International Standard. Information Technology – Process assessment – Part 5: An exemplar software life cycle process assessment model. Genebra, 2012.

___. *ISO/IEC 15504-6 (Spice)*: International Standard. Information Technology – Process assessment – Part 6: An exemplar system life cycle process assessment model. Genebra, 2013.

___. *ISO/IEC 15504-7 (Spice)*: International Standard. Information Technology – Process assessment – Part 7: Assessment of organizational maturity. Genebra, 2008.

___. *ISO/IEC 15504-8 (Spice)*: International Standard. Information Technology – Process assessment – Part 8: An exemplar process assessment model for IT service management. Genebra, 2012.

___. *ISO/IEC 15504-9 (Spice)*: International Standard. Information Technology – Process assessment – Part 9: Target process profiles. Genebra, 2011.

___. *ISO/IEC 25000*: International Standard. Information Technology – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. Genebra, 2014.

KOSCIANSKI, A.; SOARES, M. S. *Qualidade de software*. Rio de Janeiro: Novatec, 2010.

McCABE, T. A Complexity Measure. *IEEE Transactions on Software Engineering*, p. 308-20, 1976.

McCALL, J.; RICHARDS, P. K.; WALTERS, G. F. Factors in Software Quality: Concept and Definitions of Software Quality. *RADC_TR-77-369*, v. 1, Nov. 1977. Final Technical Report.

MYERS, G. *The Art of Software Testing*. 2. ed. New Jersey: John Wiley and Sons, 2004.

PAULA FILHO, W. *Engenharia de software*: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2011.

PEZZÉ, M.; YOUNG, M. *Teste e análise de software*: processo, princípios e técnicas. Porto Alegre: Bookman, 2008.

PFLEEGER, S. L. *Engenharia de software*: teoria e prática. 2. ed. São Paulo: Prentice Hall Brasil, 2007.

PRESSMAN, R. S. *Engenharia de software*. São Paulo: McGraw-Hill, 2006.

RAKITIN, S. R. *Software Verification and Validation*: a Practitioner's Guide. London: Artech House, 1997.

RAPCHAN, F. *A Norma ISO 9000-3*. Vitória: Ufes, 2005. Disponível em: <www.geocities.ws/chicorapchan/artigos/9000-3.pdf>. Acesso em: 20 jul. 2014.

RIOS, E.; MOREIRA, T. *Testes de software*. 3. ed. São Paulo: Alta Books, 2013.

ROCHA, A. R. C. *et al. Qualidade de software*: teoria e prática. São Paulo: Prentice Hall, 2001.

SCHULMEYER, G. *Handbook of Software Quality Assurance*. New York: Prentice Hall, 1999.

SOFTWARE ENGINEERING INSTITUTE (SEI). *Standard CMMI Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.2*: Method Definition Document. 2006. Disponível em: <http://www.sei.cmu.edu/cmmi>. Acesso em: 20 jul. 2014.

SOMMERVILLE, I. *Engenharia de software*. São Paulo: Pearson, 2013.

TELES, F. S. *Um processo para análise de desempenho de produtos de software*. Pernambuco: UFPE, 2005.

VILAS BOAS, A. L. C. *Qualidade e avaliação de produto de software*. Lavras: Ufla/Faepe. 2004.

Sites

<http://agilemanifesto.org>.

<http://desenvolvimentoagil.com.br/>.

<http://www.iso.org/>.

<http://jakarta.apache.org/jmeter>.

<http://www.atlassian.com/software/jira>

<http://www.mantisbt.org>

<http://www.softex.br/mpsbr/>.

<www.abnt.org.br>.

<www.edilms.eti.br>.

<www.ieee.org.br>.

<www.reliasoft.com.br>.

<www.sei.cmu.edu>.

Interativa

Informações:
www.sepi.unip.br ou 0800 010 9000