

# Engenharia de *Software*



Educação a Distância  
Cruzeiro do Sul Educacional  
Campus Virtual



# Material Teórico



Engenharia de requisitos e gerenciamento de projetos

## **Responsável pelo Conteúdo:**

Prof.<sup>a</sup> Dr.<sup>a</sup> Ana Paula do Carmo Marchetti Ferraz

## **Revisão Textual:**

Prof.<sup>a</sup> Me. Luciene Oliveira da Costa Santos



# UNIDADE

## Engenharia de requisitos e gerenciamento de projetos



- Introdução
- Requisitos
- Tipos de requisitos
- O processo de gerenciamento de configuração de software



### Objetivo de APRENDIZADO

Apresentaremos algumas questões relacionadas ao planejamento e gerenciamento de projetos e à engenharia de requisitos.

Esses assuntos nos ajudarão a entender um pouco melhor os conceitos de planejamento de desenvolvimento de produto de software, por meio dos requisitos e recursos existentes para nos ajudar na gestão do desenvolvimento destes.

Organize-se de forma a não deixar para o último dia a realização das atividades (AS e AP), pois podem ocorrer imprevistos. Encerrada a unidade, encerra-se a possibilidade de obter a nota relativa a cada atividade.

Para ampliar seu conhecimento, bem como aprofundar os assuntos discutidos, pesquise, leia e consulte os livros indicados nas Referências e/ou na Bibliografia.

As Referências estão indicadas ao final dos textos de conteúdo de cada unidade.

A Bibliografia Fundamental para esta disciplina é: SOMMERVILLE, Ian. **Engenharia de Software**. 9.ed. São Paulo: Pearson, 2011. A Bibliografia Complementar está indicada em item específico, em cada unidade.

## Contextualização

Na unidade anterior, você teve a oportunidade de aprender sobre gerência e planejamento de projeto e conhecer um pouco mais sobre a importância dessas atividades. Até na nossa vida diária planejar nossas ações e gerenciá-las são fundamentais para que tenhamos bons resultados.

Nesta unidade, você terá a oportunidade de aprender os conceitos e as definições de requisitos de software, assim como a importância do gerenciamento das configurações de um software.

Entender o que são requisitos de software é fundamental para o sucesso de um projeto de software. Outro item fundamental neste projeto são as ferramentas que são utilizadas para gerenciar as configurações de software.

Definir requisitos e gerenciar configuração não são tarefas de simples execução, mas são fundamentais para manter a qualidade do software durante seu desenvolvimento.



### Explore

Sugerimos que assista aos vídeos disponíveis em: <http://www.youtube.com/watch?v=UZfpUdYLsao> e <http://www.youtube.com/watch?v=cQiQN4NsBZ4> para entender um pouco melhor sobre a importância da definição clara dos requisitos e o gerenciamento de configurações.

Se você tiver dúvidas durante o estudo desta unidade, interaja com seus colegas, discuta sobre o tema, peça auxílio ao seu tutor.

Tenha disciplina em seus estudos, programe-se, tenha metas a serem alcançadas semanalmente e procure atingi-las.

Seu aprendizado só depende de você. Dedique-se, interaja com seus colegas e não acumule dúvidas.

Adquira o hábito da pesquisa, visite sites, pesquise em livros e periódicos. Dê pequenas pausas na leitura e pesquise sobre os conceitos abordados nas unidades.

Lembre-se de compartilhar suas descobertas com seus colegas de turma.

## Introdução



Toda vez que falamos sobre analistas e projetistas chegamos a um impasse: saber até onde a atividade de um vai e quando começa a do outro.

O analista decide o que fazer e o projetista decide como fazer. Essas duas funções andam juntas e “o que” e o “como” fazem parte dos requisitos.

Segundo Sommerville (2011), os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferecerá e seu funcionamento, ou seja, requisitos: é um conjunto de necessidades a serem atendidas, condição.

Entender o que são requisitos de software é um ponto fundamental para o sucesso de um projeto de software.

A Análise de Requisitos é uma tarefa que envolve, antes de tudo um trabalho de descoberta, refinamento, modelagem e especificação das necessidades e desejos relativos ao software que deverá ser desenvolvido.

Nessa tarefa, tanto o cliente como o desenvolvedor vão desempenhar um papel de grande importância, uma vez que caberá ao primeiro a formulação (de modo concreto) das necessidades em termos de funções e desempenho, enquanto o segundo atua como indagador, consultor e solucionador de problemas.

Essa etapa permite que o engenheiro de sistemas especifique as necessidades do software em termos de funções e de desempenho, estabeleça as interfaces com os demais elementos do sistema e especifique as restrições de projeto.

Ao engenheiro de software (ou analista), a análise de requisitos permite uma alocação mais precisa de recursos e a construção de modelos do processo, dos dados e dos aspectos comportamentais que serão tratados pelo sistema.

Ao projetista, essa etapa proporciona a obtenção de uma representação da informação e das funções, podendo ser traduzida em projeto procedural, arquitetônico e de dados.

## Requisitos



Por meio dos requisitos é possível definir critérios de avaliação da qualidade do software a serem verificados uma vez que o software esteja concluído.

O termo *requisito* não é usado de forma consistente pela indústria de software. Em alguns casos, o requisito é apenas uma declaração abstrata em alto nível de um serviço que o sistema deve oferecer ou uma restrição a um sistema.

No outro extremo, uma definição detalhada e formal de uma função de sistema. Davis (1993) explica porque estas diferenças existem: Se uma empresa pretende fechar um contrato para um projeto de desenvolvimento de software de grande porte, deve definir as necessidades de forma abstrata o suficiente para que a solução para essas necessidades não seja predefinida. Os requisitos precisam ser descritos de modo que várias contratantes possam concorrer pelo contrato e oferecer diferentes maneiras de atender às necessidades da organização do cliente. Uma vez que o contrato tenha sido adjudicado, o contratante deve escrever para o cliente uma definição mais detalhada do sistema, para que este o entenda e possa validar o que o software fará. Ambos os documentos podem ser chamados de documentos de requisitos para o sistema (SOMMERVILLE, 2011, p. 57).

## As atividades da análise de requisitos

Todo processo de análise relacionada nessa etapa define dois tipos de requisitos: Funcionais e Não Funcionais.



### Informação

Segundo Pressman (1996):

Requisito: é a condição necessária para a obtenção de certo objetivo, ou para o preenchimento de certo objetivo.

Especificação: é a descrição minuciosa das características que um material, uma obra, ou um serviço deverá apresentar.

Portanto, Especificação é diferente de Requisito e as duas atividades juntas nos ajudam a melhorar a descrição dos objetivos do projeto.

Às vezes, na literatura podemos encontrar:

- Especificação de Requisitos
- Especificação de Projeto
- Os termos são sinônimos.



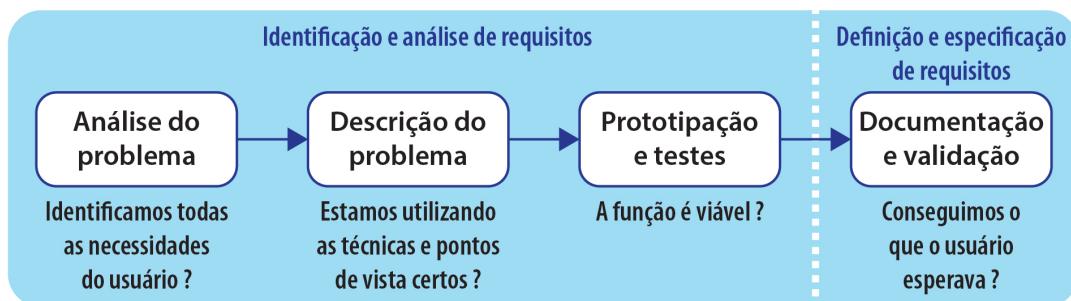
### Trocando Ideias

No começo do projeto, o analista estuda os documentos de especificação do sistema e o plano do software, como forma de entender o sistema e revisar o escopo utilizado para definir as estimativas de projeto. Quando um cliente pede para construirmos um novo sistema, ele tem uma noção do que o sistema fará, se substituirá um antigo, ou se apenas criará novos recursos ao antigo. Não importa se é um sistema novo, ou adequação de um antigo, cada um tem um propósito, uma funcionalidade.

Para iniciarmos o processo de definição de requisitos, precisamos, antes de qualquer procedimento, conversar com nosso cliente, fazer perguntas, demonstrar sistemas similares, desenvolver protótipos. Em seguida, é necessário registrarmos essas informações em um documento ou base de dados.

Geralmente, os requisitos são escritos, num primeiro momento, numa linguagem que o cliente possa entender para aprovará-los; posteriormente, eles são reescritos em uma representação mais matemática ou visual para que os projetistas possam transformar os requisitos em um projeto de sistema. Esse processo de reescrever os requisitos numa linguagem diferenciada é o processo de modelagem.

Na seguinte Figura temos a descrição do processo de requisitos.



**Figura 1** – Processo de definição de requisitos. Fonte: Pfleeger (2004, p. 112).

Existem inúmeras maneiras de identificação de requisitos de acordo com a experiência e preferência pessoal; entretanto, uma forma adequada é dividir o projeto em parte e definir quem teria as informações adequadas de forma a ajudar na definição dos requisitos para aquela fase do sistema.

Conversando, por meio de entrevista, é possível definir a amplitude do nosso sistema ou da parte que estamos responsáveis.

Durante a conversa, é necessário não deixar passar nenhuma dúvida. Ao final, durante o processo de documentação dos requisitos, eles devem ser categorizados em 3:

- 1 - Requisitos que devem ser totalmente satisfeitos.
- 2 - Requisitos que são altamente desejáveis, mas não são necessários.
- 3 - Requisitos que são possíveis, mas poderiam ser eliminados.

É importante salientarmos que nenhum requisito especifica “como o sistema deverá ser implementado”. O requisito informa o propósito do sistema.

## Tipos de documentos de requisitos

Como o enfoque é no problema do nosso cliente, a identificação e a análise dos requisitos servem para propósitos distintos, porém relacionados.

A identificação dos requisitos nos habilita a escrever um documento denominado definição de requisitos. Esse documento deve ser escrito em linguagem natural de forma que o usuário possa lê-lo, entendê-lo e interpretá-lo.

Ele representa um consenso entre o que o cliente deseja e o que é possível e será realizado.

Posteriormente, outro documento é gerado para conter as especificações dos requisitos.

Esse documento redefine o anterior para uma linguagem mais técnica, apropriada para o desenvolvimento do projeto.

Em algumas vezes, um único documento é gerado e serve para os dois propósitos: comunicação com cliente e projetista.

Uma coisa importante a ser descrita é que deve haver uma correspondência direta entre cada requisito do documento de definição e de especificação de requisitos.

É nesse momento que se inicia o processo de Gerência de Configurações utilizadas durante o ciclo de vida e que veremos com mais detalhes na próxima unidade. Segundo Pfleeger (2004), Gerência de Configurações é um conjunto de procedimentos que controlam:

- Os requisitos que definem o que o sistema fará.
- Os módulos de projetos que serão gerados a partir dos requisitos.
- O código do programa que implementa o projeto.
- Os testes que verificam a funcionalidade do sistema.
- Os documentos que descrevem o sistema.

Para Pfleeger (2004, p. 114), “[...] a gerência de configuração fornece as ‘linhas’ que unem as partes do sistema entre si, unificando os componentes que tenham sido desenvolvidos separadamente. Essas linhas nos permitem coordenar as atividades de desenvolvimento [...]”.

Especificamente, durante a identificação e análise de requisitos, o gerenciamento de configurações detalha a correspondência entre os elementos da definição de requisitos e a especificação de requisitos de tal modo que a visão do cliente esteja associada à visão do desenvolvedor, de uma maneira que possa ser acompanhada.

## Requisitos funcionais e NÃO funcionais



### Diálogo com o Autor

Segundo Pfleeger (2004, p. 115):

“Os requisitos descrevem o comportamento de um sistema. À medida que o sistema atua nos dados ou nas instruções, objetos ou entidades se ‘movem’ de um estado para outro, por exemplo: de vazio para cheio; de ocupado para desocupado; de envio para recebimento. Isto é, em um determinado estado, o sistema satisfaz um conjunto de condições; quando o sistema atua, ele pode mudar seu estado como um todo, modificando-o de um objeto.

Os requisitos expressam os estados e as transições do sistema e do objeto, de um para outro. Em particular, os requisitos descrevem atividades do sistema, tais como uma reação à inserção de dados, e o estado de cada entidade antes e depois de a atividade ocorrer. Por exemplo, em um sistema de folha de pagamento, os funcionários podem existir em pelo menos dois estados: funcionários que ainda não foram pagos e funcionários que já foram pagos. Os requisitos descrevem, como, na emissão de contracheques, um funcionário pode se ‘mover’ do primeiro estado para o segundo”.

Para descrevermos os requisitos podemos classificá-los em funcionais e não funcionais.

### A. Requisitos Funcionais

Os requisitos funcionais descrevem uma interação entre o sistema e o ambiente, descrevem como o sistema deve funcionar, considerando determinado estado, declara quais serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações.

Tomemos como exemplo o sistema de emissão de contracheque (holerite de pagamento mensal): se esse sistema for utilizado semanalmente, os requisitos funcionais deverão responder às questões relacionadas aos contracheques emitidos. Para isso podemos pensar em algumas perguntas como: quais informações são necessárias para que um contracheque seja emitido? Sob quais condições o valor poderá ser alterado? O que causa a exclusão de um funcionário da folha de pagamento?

É importante lembrarmos que as respostas a essas perguntas independem da implementação da solução para o cliente. Estamos descrevendo o que o sistema fará e quais os requisitos adicionais necessários para que ele funcione.

Imagine um sistema de matrícula escolar: Sabemos que ele deve permitir inclusão, alteração, exclusão e consulta, mas quais as perguntas que seriam necessárias para definição dos requisitos? Quais perguntas você faria para o cliente que contratou seus serviços?



#### Pense

Neste momento, é importante que você pare, compare o parágrafo anterior com o que veio antes e tente traçar um paralelo entre eles. Tente ir além do exemplo do contracheque, transponha o conteúdo abordado até aqui para um novo sistema e defina os requisitos.

### B. Requisitos Não Funcionais

Em vez de informar o que o sistema fará, requisitos não funcionais colocam restrições no sistema, por isso podem ser chamados de restrições.

Informações como: o sistema deverá rodar numa plataforma XYZ, as consultas devem ser respondidas em X intervalo de tempo, o sistema deve permitir um compartilhamento de dados entre diferentes escolas do mesmo grupo etc.

Esses requisitos limitam nossa seleção com relação à linguagem a ser utilizada, modelagem a ser utilizada (estruturada, essencial, orientada a objeto etc.).

Os requisitos não funcionais podem afetar a estrutura do sistema assim como podem gerar inúmeros outros requisitos funcionais, como é o caso de um requisito de proteção que definirá, por exemplo, os serviços necessários ao sistema (SOMMERVILLE, 2011).

Os requisitos funcionais e não funcionais devem ser definidos e identificados com os clientes de uma maneira cuidadosa, pois definirão os próximos passos de implementação, testes e manutenção.

## Tipos de requisitos



O documento de especificação de requisitos descreve tudo sobre o sistema e a integração com o ambiente. Eles incluem os seguintes itens, segundo Pfleeger (2004) e Pressman (1996):

**Tabela 1** – Questões e tipos relacionados aos requisitos.

TIPOS	QUESTÕES
<b>Ambiente físico</b>	Onde o equipamento funcionará? Esse funcionamento se dará em um ou vários locais? Existe alguma restrição ambiental, tal como temperatura, umidade ou interferência magnética? (Softwares da área médica, de controladores de voos etc. devem considerar essas interferências de forma diferenciada).
<b>Interface</b>	A entrada tem origem em outros sistemas? A saída vai para outro sistema? Existe uma maneira pré-estabelecida pela qual os dados devem ser formatados? (CEP – sempre numérico) Existe alguma mídia definida que os dados devem utilizar?
<b>Usuários e fatores humanos</b>	Quem utilizará o sistema? Haverá diversos tipos de usuários? Qual o nível de habilidade de cada tipo de usuário? Que tipo de treinamento será necessário para cada tipo de usuário? Que facilidade o usuário terá para entender e utilizar o sistema? Qual será a dificuldade para que o usuário utilize adequadamente o sistema?
<b>Funcionalidade</b>	O que o sistema fará? Quando o sistema fará? Existem diversos modos de operação? Como e quando o sistema pode ser modificado ou aprimorado? Existem limitações quanto à velocidade de execução, ao tempo de resposta, ou à saída?
<b>Documentação</b>	Que documentação é necessária? Essa documentação deve ser on-line, no formato de livro, ou ambos? A que público se destina cada tipo de informação?
<b>Dados</b>	Qual deverá ser o formato dos dados de entrada e saída? Com que frequência eles serão enviados e recebidos? Que precisão devem ter os dados? Com que grau de precisão os cálculos deverão ser feitos? Existem dados que devem ser mantidos por determinado tempo?

<b>Recursos</b>	Que materiais, pessoal ou outros recursos são necessários para construir, utilizar e manter o sistema? Que habilidade os desenvolvedores devem ter? Quanto espaço físico será ocupado pelo sistema? Quais os requisitos quanto à energia, ao aquecimento ou condicionamento de ar? Existe um cronograma já definido para o desenvolvimento? Existe um limite de custo para o desenvolvimento ou para a aquisição de hardware ou de software?
<b>Segurança</b>	O acesso ao sistema ou às informações deve ser controlado? Como os dados de um usuário serão isolados dos outros usuários? Como os programas dos usuários serão isolados de outros programas e do sistema operacional? Com que frequência será feito backup? Onde serão armazenadas as cópias de backup? Devem ser tomadas precauções contra fogo, danos provocados pela água, ocorrência de roubo etc.?
<b>Garantia de qualidade</b>	Quais os requisitos quanto à confiabilidade, disponibilidade, manutenibilidade, segurança e outros atributos de qualidade? Como as características do sistema devem ser demonstradas para os outros? O sistema deve detectar e isolar defeitos? Qual o tempo médio entre falhas, que foi determinado? Existe um tempo máximo permitido para reiniciar o sistema depois de uma falha? Como o sistema pode incorporar modificações no projeto? A manutenção corrigirá os erros ou também incluirá o aprimoramento do sistema? Que medidas de eficiência serão aplicadas à utilização dos recursos e ao tempo de resposta? Com que facilidade o sistema se deslocará de um local para outro (acesso remoto) ou de um tipo de computador para outro?

Alguns passos podem ser seguidos para descobrir exatamente o que o cliente quer, como é o caso de várias entrevistas e reuniões. Basicamente, durante o processo de especificação de requisitos, devemos:

- Analisar a situação atual da empresa (se existe algum sistema, funcionalidades etc.)
- Fazer com que o usuário aprenda a entender o contexto, os problemas e os relacionamentos do sistema solicitado.
- Entrevistar usuários atuais e potenciais.
- Demonstrar como o novo sistema deve operar.
- Pesquisar documentação existente.
- Realizar um brainstorming com os usuários potenciais e usuais.
- Observar estruturas e padrões.
- Documentar todo o processo e todas as reuniões.

## Revisão dos Requisitos

Todo requisito permite ao desenvolvedor explicar seu entendimento de como os clientes querem que o sistema funcione. Eles devem informar aos projetistas quais funcionalidades e características o sistema resultante deve ter; e informar a equipe de teste sobre o que demonstrar para convencer o cliente de que o sistema será entregue de acordo com o que foi solicitado.

Para garantir que os itens especificados sejam alcançados, é importante verificarmos sempre se: os requisitos estão corretos, são consistentes, estão completos, são realistas, cada um descreve algo que é importante para o cliente, podem ser verificados, podem ser rastreados etc.

Com relação ao sistema, após definirmos os requisitos funcionais e não funcionais, é importante revisarmos para verificar se as metas e objetivos do software permanecem consistentes com as metas e os objetivos do projeto, se as interfaces importantes para todos os elementos do sistema foram descritas, se o fluxo e a estrutura da informação são adequadamente definidos para o domínio da informação, se a modelagem e diagrama estão claros, se há risco tecnológico do desenvolvimento, se os requisitos de software alternativos foram considerados, se existem inconsistências, redundâncias ou omissões, se o que o cliente desejava foi contemplado por completo.

## Documento de Definição de Requisito

A documentação da definição de requisitos é fundamental e deve sempre ser descrita em linguagem que os clientes entendam. Utilizaremos linguagem técnica apenas na modelagem do sistema.

Este documento é construído ao longo do processo de desenvolvimento. Alguns capítulos serão finalizados antes, outros durante e outros ao final do desenvolvimento do sistema.

Para tanto, Pfleeger (2004) e Sommerville (2011) definem os seguintes tópicos que deverão ser estruturados em capítulos, conforme a Tabela 2:

**Tabela 2** – Estrutura do documento de requisitos.

Capítulo	Descrição
<b>Prefácio</b>	Define possíveis leitores do documento assim como o histórico das versões (criação, alteração, exclusão etc.).
<b>Introdução</b>	Descreve a necessidade para o sistema. Explica as funções dos sistemas e como deverão ser integradas (funcionar com) a outros sistemas, além de como atender aos objetivos globais dos negócios do solicitante.
<b>Glossário</b>	Descreve termos técnicos utilizados.
<b>Definição dos requisitos de usuários</b>	Descreve o serviço a ser fornecido aos usuários. Os requisitos não funcionais também são descritos neste item. Esta descrição pode ser em linguagem natural, diagrama ou outra forma, desde que facilmente compreensível ao cliente; afinal, ele terá que aprovar este documento.
<b>Arquitetura do sistema</b>	Apresenta a visão geral e de alto nível da arquitetura do sistema, inclusive mostrando a distribuição de funções entre os módulos. Aqui devem ser descritos os objetos que serão reutilizados no desenvolvimento do projeto.

<b>Especificação dos requisitos do sistema</b>	Descreve em detalhes os requisitos funcionais e não funcionais.
<b>Modelos do sistema</b>	Inclui modelos gráficos que mostram os relacionamentos dos componentes do sistema e do ambiente em que operará. Aqui entra as descrições e documentação de modelagem (fluxo de dados, relacionamentos, objetos etc.).
<b>Evolução do sistema</b>	Descreve os pressupostos fundamentais em que o sistema se baseia e todas as suas mudanças previstas em decorrência da evolução do hardware. Esta função é útil ao projetista do sistema porque poderá dimensionar futuras versões e tecnologias a serem consideradas.
<b>Apêndice</b>	Neste item, informações detalhadas são incorporadas – hardware, banco de dados, modelagem, configurações mínimas exigidas, organização lógica dos dados, modelagens etc.
<b>Índice</b>	Vários índices podem existir: alfabético (normal), de diagramas, de figuras, funções etc.

De posse de todas as especificações de requisitos, podemos dar início ao processo de modelagem. Nesta disciplina, não abordaremos os conceitos de modelagem. Caso não tenha tido acesso a esse assunto, sugerimos uma pesquisa complementar sobre as técnicas existentes, pois a modelagem de um sistema junto com a definição e especificação de requisitos são documentos importantes para os desenvolvedores conseguirem realizar seu trabalho de forma eficiente e eficaz, cumprir os prazos e as estimativas.

## Gerenciamento de configurações

Alterar um software é inevitável quando o estamos desenvolvendo. Inúmeras ideias surgem para melhoria, solicitações por parte do cliente ou dos próprios gerentes de desenvolvimento, e a solução é mexer no projeto inicial para adequá-lo.

Geralmente, quando falamos em projeto de desenvolvimento de software, estamos nos referindo a sistemas complexos, feitos por módulos por diversos desenvolvedores que podem estar ou não fisicamente distantes e conhecer ou não todos os membros da equipe de desenvolvimento.

É nesse cenário que problemas em modificações em rotinas, sistema ou até mesmo no escopo do projeto podem ser catastróficos se não forem coordenados e gerenciados.

Imagine você desenvolvendo um módulo, vai para casa descansar no final do expediente e, no dia seguinte, percebe que algumas rotinas foram alteradas? E o pior: percebe que foram alteradas por alguém que não conhecia direito o quanto aquela rotina influenciava outras do próprio sistema e nem que era usada em outro sistema de outro usuário.

Com certeza, nessa cena, algum desconforto entre as equipes surgiria.

Mudanças devem ser analisadas antes da realização, registradas antes da implementação, relatadas aos que precisam tomar conhecimento delas ou controladas de modo que melhorem a qualidade geral do projeto e reduzam erros. (PRESSMAN, 1996).

É nesse cenário de controlar as alterações que surge o Gerenciamento de Configurações de Software (GCS) ou Software Configuration Management (SCM).

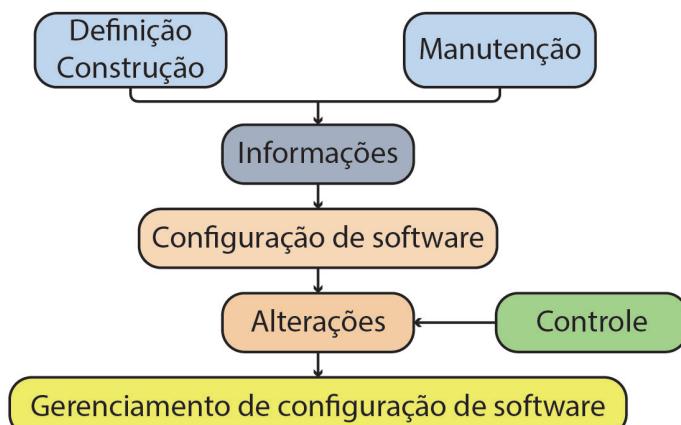
GCS é uma atividade abrangente que é aplicada em todo o processo de engenharia de software. Uma vez que uma mudança pode ocorrer a qualquer tempo, as atividades de GCS são desenvolvidas para:

1. Identificar mudanças.
  2. Controlar mudanças.
  3. Garantir que a mudança esteja sendo adequadamente implementada.
  4. Relatar mudanças a outras pessoas que possam ter interesse nelas.
- (PRESSMAN, 1996, p. 916)

GCS pode ser considerada como a arte de coordenar o desenvolvimento de software para minimizar os problemas de mudanças e alterações antecipadamente.

É importante salientar que GCS não é manutenção.

Segundo Pressman (2006, 1996), Sommerville (2011) e Pfleeger (2004), a manutenção é um conjunto de atividades de Engenharia de Software (ES) que acontece depois que o sistema foi entregue para o , enquanto GCS é um conjunto de atividades de ES que acontece no momento que o projeto começa a ser desenvolvido e vai até o momento de ser retirado do mercado. É um conjunto de atividade contínua (Figura 2).



**Figura 2** – Descrição ilustrativa sobre GCS.



## Diálogo com o Autor

Uma meta primordial da metodologia de Engenharia de Software é melhorar a facilidade com que as mudanças podem ser acomodadas e reduzir a quantidade de esforços despendidos quando as mudanças são feitas. (PRESSMAN, 1996, p. 917)

O resultado de todo processo de Engenharia de Software são informações que podem ser divididas em:

- 1 - Código fonte: executável e fonte, que na verdade é o resultado do projeto contratado.
- 2 - Documentos que apoiam e esclarecem todo o processo e itens relacionados a ele.
- 3 - Estrutura de dados: existente e integrada que apoia e é apoiada (tanto para o item 1, quanto para o 2).

Os itens gerados de todos esses componentes são chamados de **Itens de Configuração de Software (ICS) ou Software Configuration Items (SCI)**, e sua descrição e informações relacionadas a eles deverão fazer parte da documentação do Plano de Projeto.

Seria tranquilo se pudéssemos trabalhar por meio da garantia da não mudança do Plano de Projeto ou da Especificação de Software. Infelizmente, não é isso que acontece. Mudanças ocorrem o tempo todo, aliás, se você está desenvolvendo um projeto de software, não importa em qual momento do ciclo de desenvolvimento você esteja, com certeza, mudanças serão implementadas por você e solicitadas pelos membros da equipe de desenvolvimento e/ou projetista e/ou gerente e/ou qualquer pessoa que conheça e pense no projeto. Isso é tão certo quanto  $2+2 = 4$ !

Gerenciar as mudanças nos oferece um ambiente estável de desenvolvimento, pois alterações sem controle proporcionam um ambiente caótico. Na nossa unidade 1, vimos que a ES veio para colocar ordem numa atividade que era caótica: a de desenvolvimento de software sem ordem, padrão ou gerenciamento adequado.

Existem técnicas e conceitos que nos ajudam a coordenar essas mudanças e é isso que veremos a partir de agora.

## Baseline – linhas básicas

Mudanças: éis uma coisa constante num ambiente tão instável como é o de desenvolvimento de um projeto.

Clientes querem modificar requisitos, administração quer modificar a abordagem do projeto, desenvolvedores querem modificar abordagens técnicas etc.

Com o passar do tempo e a familiaridade com o desenvolvimento de cada projeto, novas ideias, sugestões e mudanças surgem e o engenheiro de software tem que enfrentar uma realidade difícil: muitas das mudanças solicitadas são justificáveis, ou seja, precisam ser feitas.

Uma Linha Básica ou Baseline é um conceito de GCS que ajuda a controlar as mudanças sem impedir seriamente as mudanças justificáveis. Ou seja: se uma mudança é justificável, ela poderá ser feita, mas por meio de uma linha básica.

Pressman (1996) nos descreve o mecanismo da linha básica por meio de uma analogia:

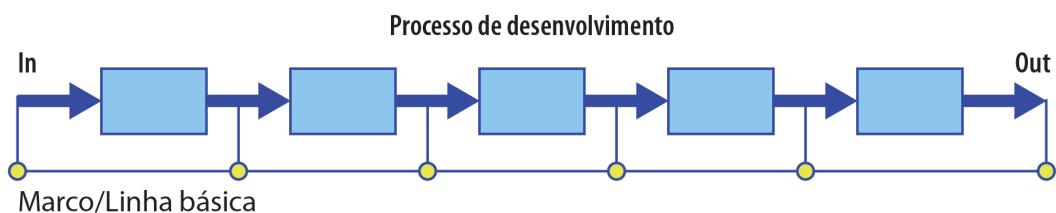
Consideremos as portas da cozinha de um grande restaurante. Para eliminar colisões, uma porta é marcada como entrada e outra como saída. As portas têm quatro paradas que permitem que elas sejam abertas somente na direção apropriada.

Se um garçom pegar um pedido na cozinha, colocá-lo numa bandeja e depois perceber que escolheu o prato errado, ele pode mudar para o prato certo rápida e informalmente antes de sair da cozinha.

Se, entretanto, ele sair da cozinha, entregar o prato ao cliente e depois for informado de seu erro, ele deve seguir uma série de procedimentos:

1. Olhar a conta para determinar se ocorreu o erro;
  2. Desculpar-se profundamente;
  3. Retornar à cozinha pela porta ENTRADA;
  4. Explicar o problema, e assim por diante.
- (PRESSMAN, 1996, p. 918)

Uma baseline é idêntica a portas da cozinha de um restaurante. Antes de um item de configuração de software se tornar uma linha básica, mudanças podem ser feitas de modo rápido e informalmente. Após esse período, mudanças devem ser feitas formalmente e por meio de procedimentos e etapas (Figura 3).



**Figura 3** – Definição das linhas básicas.

Uma linha básica é um marco de referência. Antes de um item se tornar uma linha básica, inúmeras alterações podem ocorrer, pessoas diferentes podem alterá-los (além do desenvolvedor), sugestões podem ocorrer; uma vez que o item foi considerado referência/marco, alterações só poderão ser realizadas por meio de solicitação e documentação formal.

Linhas básicas podem ocorrer em qualquer fase do projeto e, uma vez definidas informações sobre elas, são colocadas no que chamamos de repositório de software ou repositório de itens de configuração ou biblioteca de projeto.

Quando alguém da equipe de Engenharia de Software quer fazer uma mudança num SCI definido com linha básica, este item é copiado do da Base de Dados (BD) para a área particular do solicitante e um registro dessa solicitação é guardado num arquivo chamado contábil. O solicitante então poderá alterar o arquivo copiado até que todas as mudanças sejam completadas e só então ele voltará à base de dados inicial já atualizada. Em algumas circunstâncias, o arquivo original é bloqueado e ninguém pode utilizá-lo até que todas as mudanças sejam executadas e o arquivo seja atualizado, revisto, aprovado e liberado novamente para uso de todos da equipe.

As linhas básicas podem ocorrer ao final de cada fase do desenvolvimento ou quando gerentes, projetistas, engenheiros ou desenvolvedores decidirem. Quando um item passa, é definido como linha básica e, pelo baseline, ele é considerado baselined, ou é dito que o item tornou-se uma linha básica.

### **Características de um item baselined:**

- Foi revisto formalmente e teve acordo das partes.
- Serve como base para o trabalho futuro.
- É armazenado no repositório dos itens de configuração (RIC).
- Pode ser alterado somente através de procedimentos formais de controle de mudança.

## O processo de gerenciamento de configuração de software



Para controlarmos o processo de GCS, é necessário coordenarmos as alterações de algumas atividades, como veremos a seguir.

O processo consiste em:

- A. Selecionar o item a ser gerenciado.
- B. Identificar objetos.
- C. Controlar versão.
- D. Controlar mudanças.
- E. Fazer auditoria de configuração.
- F. Elaborar relatório de status do sistema.
- G. Controlar interface.
- H. Controlar fornecedores de informação.

### **A. Selecionar o item a ser gerenciado**

Selecionar o item a ser gerenciado é uma das tarefas mais difíceis, pois ele deve ser escolhido por meio de análise minuciosa. Esse é novamente um processo que engloba técnica e criatividade.

Se não selecionar qual item será gerenciado e não souber exatamente o porquê disso, poderá acontecer uma exagerada documentação e um atraso nas funções de desenvolvimento desnecessárias.

O processo para execução dessa atividade pode ser estruturado por:

- Selecione o item.
- Descreva como o item se relaciona a outros no BD.
- Planeje as linhas de referências tendo como base o ciclo de desenvolvimento do projeto.

- Descreva quando pode ser alterado (descrever quais as justificativas que serão aceitas e em que circunstâncias).
- Descreva como pode ser alterado (se haverá necessidade ou não de suspender sua execução e utilização durante o período de mudança).
- Crie um documento e anexe ao plano do projeto.
- Insira essas informações, de forma adequada, no repositório de itens de configuração.

## B. Identificação de objetos na configuração de software

Para controlar e administrar ICS, cada um deve ser nomeado separadamente e depois organizado, usando uma abordagem orientada a objeto.

Cada item é considerado um objeto, por isso a necessidade do nome, da descrição, da lista de recursos e identificação, conforme o seguinte:

- Nome: nome dado ao objeto.
- Descrição: é uma lista de itens de dados que identifica o tipo de ICS (documento, programa, dados) que é representado por objeto, identificador do projeto, informações sobre mudanças ou versões.
- Lista de recurso é a lista de entidades fornecidas, processadas, consultadas ou exigidas pelo objeto.
- A identificação do objeto de configuração também deve levar em consideração objetos relacionados ao objeto nomeado.

Exemplo de uma informação existente no ICS:

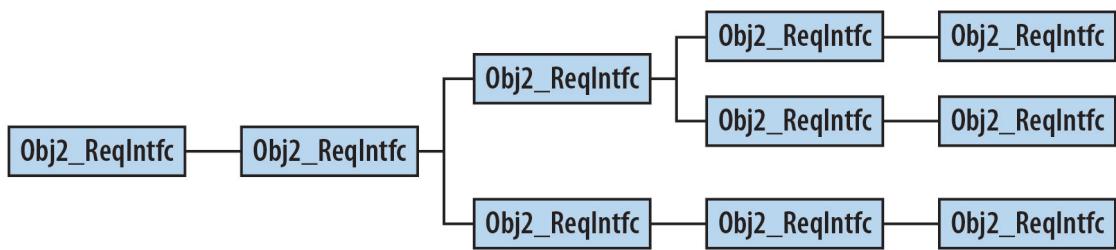
Item	Id	tipo	nome	versão	nome completo
Especificação do Projeto	ProjAA	EP	Obj1_escopo	V1.0	AAEP Obj1_escopo V10
Especificação de Requisito	ProjAA	ER	Obj2_ReqIntfc	V2.0	AAER Obj2_ReqIntfc V20

Os objetos podem ser:

**Básicos:** unidade de texto que foi criada por um engenheiro de software durante a análise, projeto, codificação ou teste. Um exemplo pode ser uma seção de especificação de requisitos ou uma listagem fonte para um módulo.

**Composto:** é uma coleção de objetos básicos e outros objetos compostos.

O controle das alterações desses objetos pode ser documentado por meio de gráficos como podemos ver na Figura 4:



**Figura 4** – Representação gráfica do controle de objetos nos ICS também chamado de Gráfico de Evolução.

A representação gráfica do controle de mudanças dos ICS tratados como objetos fará parte da documentação do Plano de Projeto.

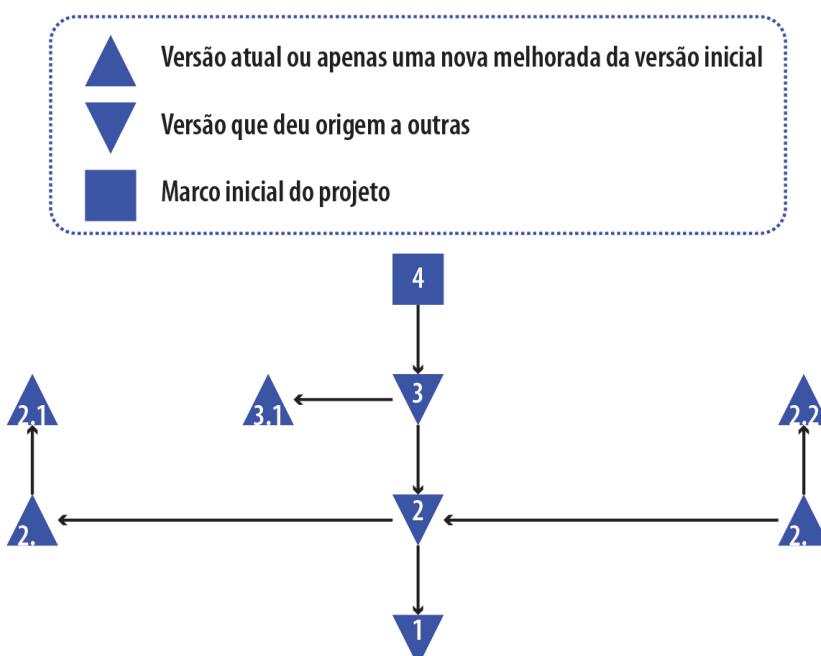
### C. Controle de versão

Esse controle combina procedimentos e ferramentas para gerenciar diferentes versões de objetos de configurações.

O gerenciamento de configurações permite que o usuário identifique configurações alternativas do sistema de software por meio da escolha de versões apropriadas. Isso é levado ao efeito ao associar atributos a cada versão de software e depois permitir que uma configuração seja especificada [e construída], descrevendo-se o conjunto de atributos desejados. (PRESSMAN, 1996, p. 927)

Como no item anterior, o controle de versões também pode ser representado graficamente, conforme veremos na Figura 5.

Uma das representações existentes para ser usada nesse caso é a chamada “Árvore Delta”. Nela, são inseridas figuras delta, de acordo com sua relação com a mudança, por exemplo:



**Figura 5** – Representação gráfica de Árvore Delta para o controle de versão.

A representação gráfica do controle de versões dos ICS fará parte da documentação do Plano de Projeto.

## D. Controle de mudanças

Não controlar as mudanças significa trabalhar num ambiente caótico e consequentemente realizar trabalhos com qualidade questionável.

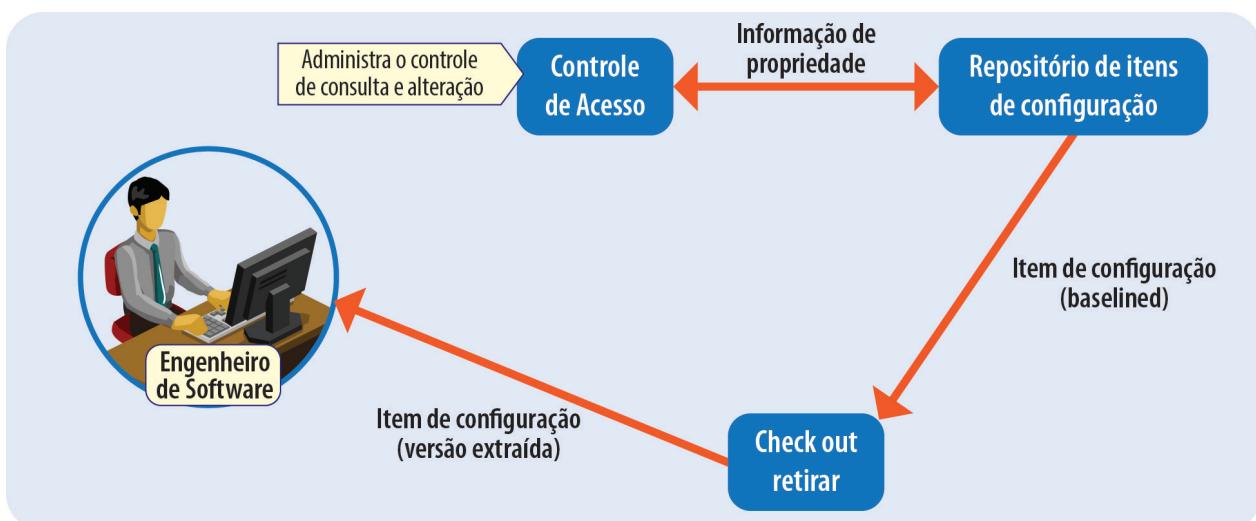
O controle de mudanças combina procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle das mudanças.

Um pedido de mudança sempre será submetido a avaliações técnicas. Devem ser analisados os potenciais efeitos colaterais, o impacto global sobre os outros objetos de configuração e funções do sistema e o custo projetado da mudança.

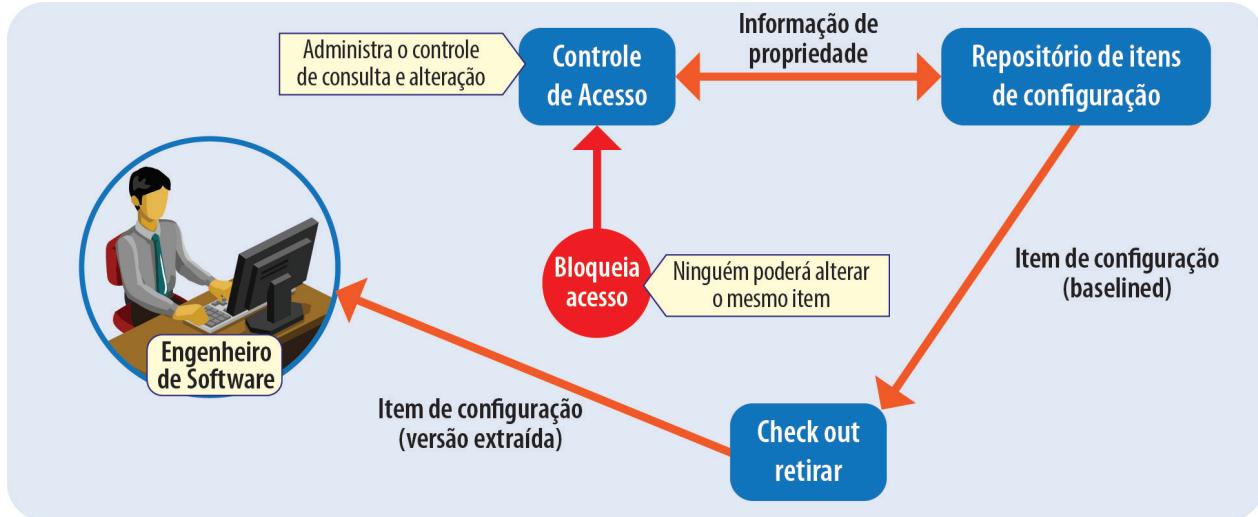
O resultado dessa avaliação é apresentado num documento chamado relatório de mudanças e uma equipe pertencente à Autoridade Controladora de Mudanças (Change Control Authority – CCA) toma a decisão final sobre o status e a prioridade da mudança.

Uma Ordem de Mudança de Engenharia (Engineering Change Order – ECO) é gerada para cada mudança aprovada. Na ECO, é descrita a mudança a ser feita, as restrições que devem ser respeitadas e os critérios de revisão e auditoria.

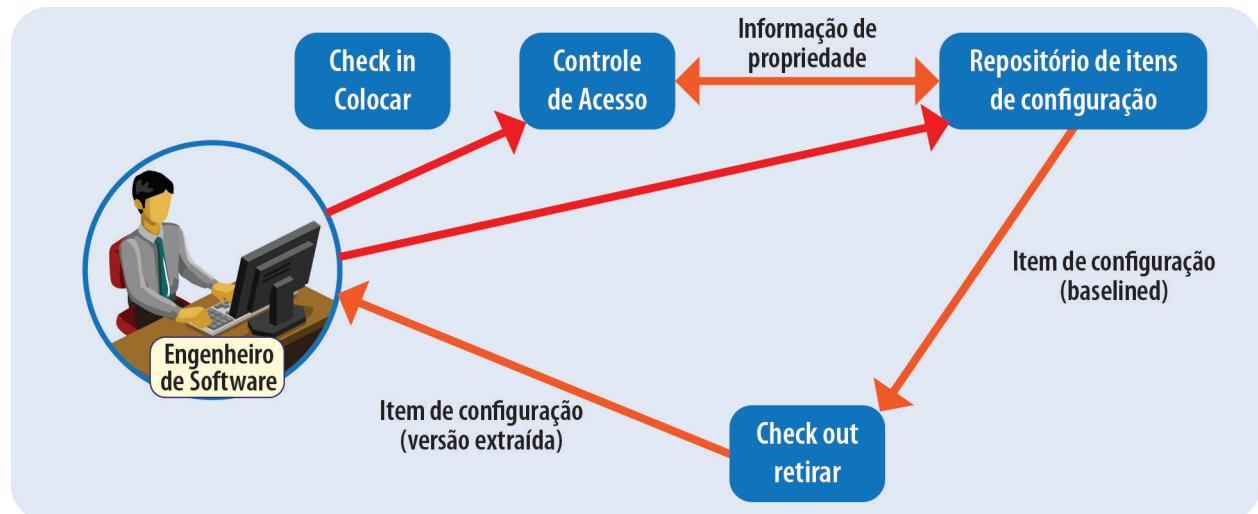
Nesse momento, o processo a ser alterado passa por um processo de check-out. Esse processo retira o item do repositório, cataloga-o no sentido de informar que ele está sendo alterado. Posteriormente, após o item ser alterado, ele passa por um período de check-in – inserção do objeto no repositório (figuras 6, 7 e 8).



**Figura 6** – Descrição ilustrativa sobre *check-out*



**Figura 7** – Descrição figurativa do bloqueio do objeto no repositório.



**Figura 8** – Item alterado e inserido no repositório.

## E. Auditoria de Configurações

A identificação, o controle de versão e o controle de mudanças ajudam o desenvolvedor de software a manter ordem daquilo que, de outro modo seria uma situação caótica e difícil de ser gerenciada.

Entretanto, a mais controlada ação de mudança é capaz de controlar e rastrear as mudanças até o ponto em que uma ECO é gerada.

Para garantirmos que a mudança seja adequadamente realizada é necessário:

4 - Revisões técnicas formais.

5 - Auditoria de configuração de software.

Uma **revisão técnica formal** deve ser realizada para todas as situações. Ela focaliza a exatidão técnica do objeto de configuração que foi alterado e é avaliada sua consistência com relação a outros ICS.

Uma **auditoria de configuração de software** complementa a revisão técnica formal ao avaliar um objeto de configuração quanto às características que geralmente não são consideradas durante a revisão.

[...] a auditoria pergunta e responde às seguintes questões:

A mudança especificada na ECO foi feita? Outras modificações adicionais foram incorporadas?

Uma revisão técnica formal foi realizada para avaliar a exatidão técnica?

Os padrões de ES foram adequadamente seguidos?

A mudança foi “realçada” no ICS? A data da mudança e o autor da mudança foram especificados? Os atributos do objeto de configuração refletem a mudança?

Os procedimentos de GCS para anotar a mudança, registrá-la e relatá-la foram seguidos?

Todos os ICS relacionados foram adequadamente atualizados?

(PRESSMAN, 1996, p. 934)

## F. Relato de status de configuração

A construção do Relato de Status de Configuração (RSC) – *Configuration Status Report* (CSR) – é uma tarefa que visa a responder às questões sobre o que aconteceu, quem fez, quando e o que mais é afetado.

Toda vez em que um ICS recebe uma identificação nova, é atualizado; toda vez em que uma auditoria é realizada, uma entrada de RSC é realizada.

Essas informações, geralmente são geradas online e podem ser acessadas pela administração e por profissionais para que eles fiquem a par das mudanças ocorridas.

## G. Controle de interface

Este item segue os mesmos padrões dos anteriores: coordena as mudanças de interfaces dos objetos que pertencem ao ICS.

As mudanças de interfaces solicitadas são coordenadas pelo processo de controle de mudanças e por meio delas é atualizada a base de dados que contém as informações sobre as interfaces/acesso a dados.

## H. Controle de fornecedores de informação

Nem sempre toda informação e base de dados acessadas por um projeto faz parte de uma BD pertencente ao cliente. Algumas vezes, informações e BD de terceiros são relacionadas e denominadas Base de Dados de Fornecedor.

Esse item controla mudanças de terceiros acoplados, mudanças essas coordenadas pelos itens anteriores (A-F).

As questões relacionadas à Gerência de Configuração de Software (GCS) têm como objetivo coordenar e auditar todas as mudanças ocorridas no sistema do seu desenvolvimento até o momento que é retirado do mercado.

É importante salientar que não é manutenção e nem procedimentos que ocorrem após a entrega do projeto ao cliente. GCS visa responder, por meio de um processo estruturado, às seguintes questões: o que mudou e quando? Por que mudou? Quem fez a mudança? Podemos reproduzir esta mudança?

Nesse contexto, podemos dizer que o controle de versão é capaz de dizer o que mudou e quando mudou, além de atribuir os motivos a cada uma das mudanças.

A auditoria, por sua vez, responde às duas últimas perguntas: quem fez a mudança? Podemos reproduzir a mudança? Cada uma dessas perguntas está diretamente relacionada ao processo de controle de atualização e mudanças do projeto que é de vital importância para a Engenharia de Software.

## Material Complementar



### Explore

O objetivo do material complementar é lhe ajudar a entender, sob uma ótica diferente daquela da professora confeudista, assuntos abordados nas unidades teóricas.

É fundamental a leitura deste material para o melhor entendimento sobre o assunto.

Como nesta unidade abordamos os conceitos gerais da Engenharia de Software, nossa sugestão de material complementar é o capítulo 4, intitulado Engenharia de requisitos, no seguinte livro:

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011. p. 57-81.

## Referências

### Bibliografia Fundamental

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

### Bibliografia Básica

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação**. 4. ed. Rio de Janeiro: LTC, 1999.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação Gerenciais**. Administrando a empresa digital. 5. ed. São Paulo: Pearson Education do Brasil, 2006.

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

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 2006.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Pearson Addison Wesley, 2005.

### Bibliografia Complementar

ALCADE, E.; GARCIA, M.; PENUELAS, S. **Informática Básica**. São Paulo: Makron Books, 1991.

FAIRLEY, R. E. **Software engineering concepts**. New York: McGraw-Hill, 1987.

IEEE. **Software Engineering Standards**. 2013. Disponível em: <[http://www.ieee.org/portal/innovate/products/standard/ieee\\_soft\\_eng.html](http://www.ieee.org/portal/innovate/products/standard/ieee_soft_eng.html)>. Acesso em: 10 dez. 2013.

LUKOSEVICIUS, A. P.; CAMPOS FILHO, A. N.; COSTA, H. G. **Maturidade em gerenciamento de projetos e desempenho dos projetos**. Disponível em: <[www.producao.uff.br/conteudo/rpep/.../RelPesq\\_V7\\_2007\\_07.doc](http://www.producao.uff.br/conteudo/rpep/.../RelPesq_V7_2007_07.doc)>. Acesso em: 12 nov. 2013.

MAFFEO, B. **Engenharia de Software e especialização de sistemas**. Rio de Janeiro: Campus, 1992.

MICHAELIS. **Moderno dicionário da língua portuguesa**. São Paulo: Cia. Melhoramentos, 1998.

PARREIRA JÚNIOR, W. M. **Apostila de Engenharia de Software**. Disponível em: <[http://www.waltenomartins.com.br/ap\\_es\\_v1.pdf](http://www.waltenomartins.com.br/ap_es_v1.pdf)>. Acesso em: 13 nov. 2013.

PAULA FILHO, W. P. **Engenharia de Software**: fundamentos, métodos e padrões. 2. ed. Rio de Janeiro: LTC, 2001.

**Revista Engenharia de Software.** Disponível em: <<http://www.devmedia.com.br/revista-engenharia-de-software-magazine>>. Acesso em: 12 nov. 2013.

VONSTA, A. **Engenharia de Programas.** Rio de Janeiro: LTC, 1983.

WIENNER, R.; SINCOVEC, R. **Software engineering with Modula 2 and ADA.** New York: Wiley, 1984.

WIKIPEDIA (2007a). **ISO 9000.** Disponível em: <[http://pt.wikipedia.org/wiki/ISO\\_9000](http://pt.wikipedia.org/wiki/ISO_9000)>. Acesso em: 22 jun. 2007.

WIKIPEDIA (2007b). **Melhoria de Processo de Software brasileiro.** Disponível em: <<http://pt.wikipedia.org/wiki/MPS.BR>>. Acesso em: 22 jun. 2007.

WIKIPEDIA (2007c). **CMMI.** Disponível em: <<http://pt.wikipedia.org/wiki/Cmmi>>. Acesso em: 22 jun. 2007.

WIKIPEDIA (2007d). **Engenharia de Software.** Disponível em: <[http://pt.wikipedia.org/wiki/Engenharia\\_de\\_software](http://pt.wikipedia.org/wiki/Engenharia_de_software)>. Acesso em: 01 fev. 2007.

# Anotações







**Educação a Distância**  
Cruzeiro do Sul Educacional  
*Campus Virtual*

[www.cruzeirodosulvirtual.com.br](http://www.cruzeirodosulvirtual.com.br)  
Campus Liberdade  
Rua Galvão Bueno, 868  
CEP 01506-000  
São Paulo SP Brasil  
Tel: (55 11) 3385-3000



Universidade  
**Cruzeiro do Sul**



**UNICID**  
Universidade  
Cidade de S. Paulo



**UNIFRAN**  
Universidade  
de Franca



**UDF**  
Centro  
Universitário



**Módulo**  
Centro  
Universitário