INSTITUTO FEDERAL
Sul de Minas Gerais

2018

MEDIOTEC - REDE - E TEC

APOSTILA DE
Informática
Engenharia de Software

Prof. : Straus Michalsky

MEDIOtec

Sumário

Apresentação da disciplina 3

1. Introdução 4

2. Processo desenvolvimento de software 8

3. Modelos de processos de desenvolvimento de software 12

   3.1. Modelo em cascata 12

   3.2. Modelo Incremental 15

4. Gerência de projetos 18

   4.1. Requisitos 21

   4.2. Restrições 22

   4.3. Riscos 23

   4.4. Contratos 28

Apresentação da disciplina

Esta disciplina tem como objetivo ensinar técnicas utilizadas para o desenvolvimento de softwares de forma profissional. Processos de desenvolvimento de software, ferramentas para gerência de projetos, entre outras atividades, são apresentadas no decorrer deste curso.

A ementa da disciplina é a seguinte: Gerência de projetos; Histórico e fundamentos; Avaliação e gerenciamento de riscos de projetos; Organização, negociação e planejamento de projetos. Modelos clássicos de processos de software. Atividades comuns nos principais modelos de processos de software.

O livro base será ***Engenharia de Software***: Uma Abordagem Profissional, de Roger S. Pressman.

O conteúdo ministrado será dividido em quatro partes:

1 Introdução - onde os problemas comuns no desenvolvimento de software serão apresentados junto com o cenário atual do desenvolvimento de software.

2 Processo desenvolvimento de software - apresentará a definição de processo e de processo de software e as atividades fundamentais presentes no desenvolvimento de um software.

3 Modelos de processo - os modelos de processo de software são simplificações de processos reais, assim, nesta seção serão estudados os modelos em Cascata e Incremental.

4 Gerência de projetos - algumas ferramentas de gerenciamento de projetos serão apresentadas, trabalharemos também com os diferentes tipos de contratos de software e com o gerenciamento de riscos.

Bons estudos!

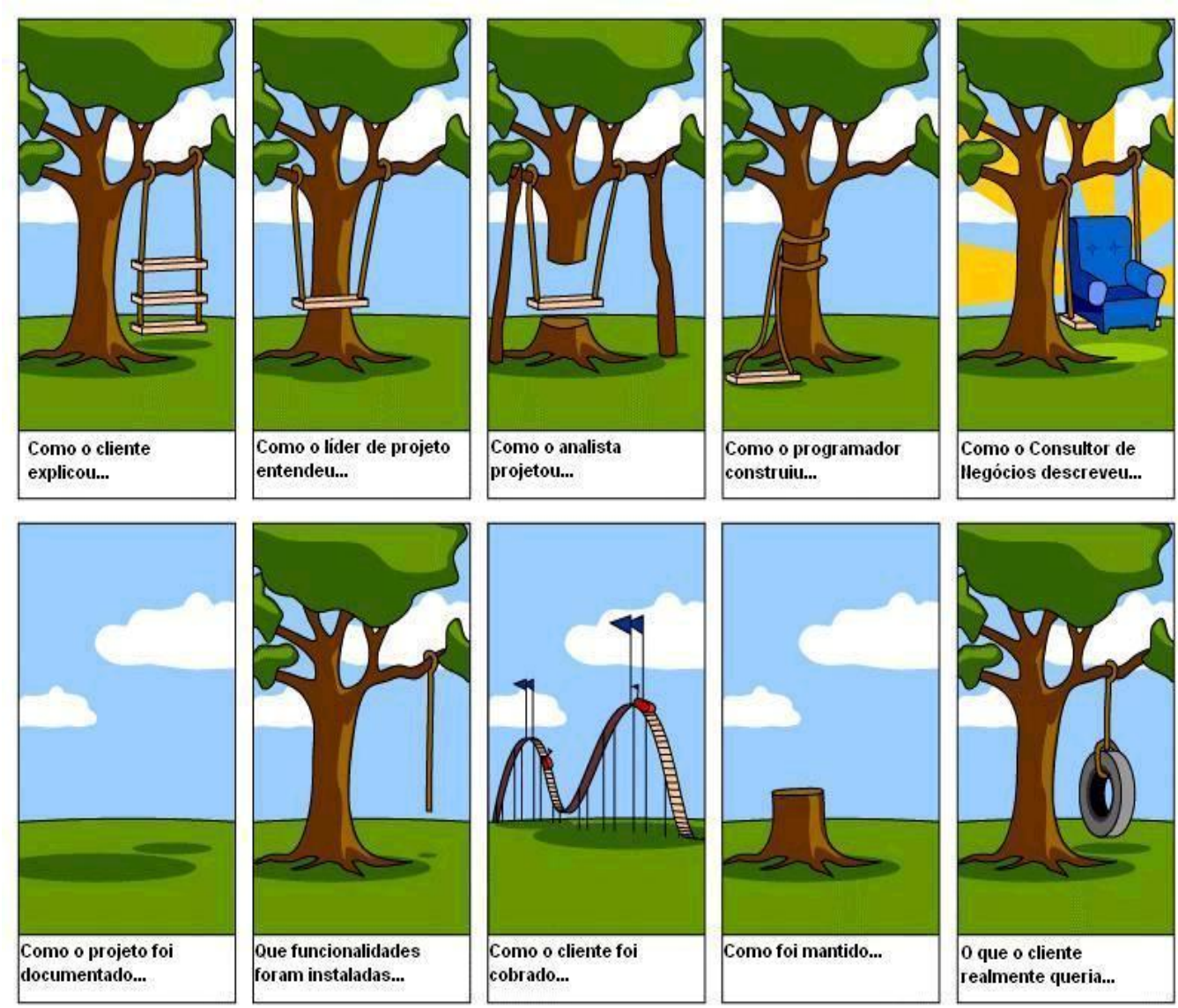
1. Introdução

Engenharia de software é uma área da computação voltada à especificação, desenvolvimento, manutenção e criação de software, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade.

Atualmente, essas tecnologias e práticas englobam linguagens de programação, banco de dados, ferramentas, plataformas, bibliotecas, padrões de projeto de software, processo de software e qualidade de software. Além disso, a engenharia de software deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento de um sistema computacional de qualidade e que atenda as necessidades de um requisitante de software.

Desta forma, durante o curso, iremos discutir as diversas atividades que abrangem a criação de um software. Essa criação passa a ser feita por grupos de pessoas, uma vez que se tornam grandes demais para serem desenvolvidos por uma só pessoa.

Assim, quando começamos a criar software de forma profissional com um grupo de pessoas, teremos diferentes pessoas desempenhando diferentes papéis durante o desenvolvimento do software. A seguir temos um exemplo de como isso pode ser feito.



Essa tirinha mostra alguns dos problemas que uma equipe de desenvolvimento de software encontra no mercado de trabalho. Nela é representada também a falha de comunicação entre os diversos profissionais envolvidos. Durante o curso, falaremos dessas dificuldades, que muitas vezes começam pelo próprio entendimento do que o cliente quer e passa pela análise, projeto, desenvolvimento e manutenção do sistema.

Para ajudar a nos situar, precisamos entender também o histórico do desenvolvimento de software, nele temos uma crise do software, que mostra que essa área precisa de uma atenção muito grande para não apresentar resultados desanimadores ao final do processo de desenvolvimento de um software.

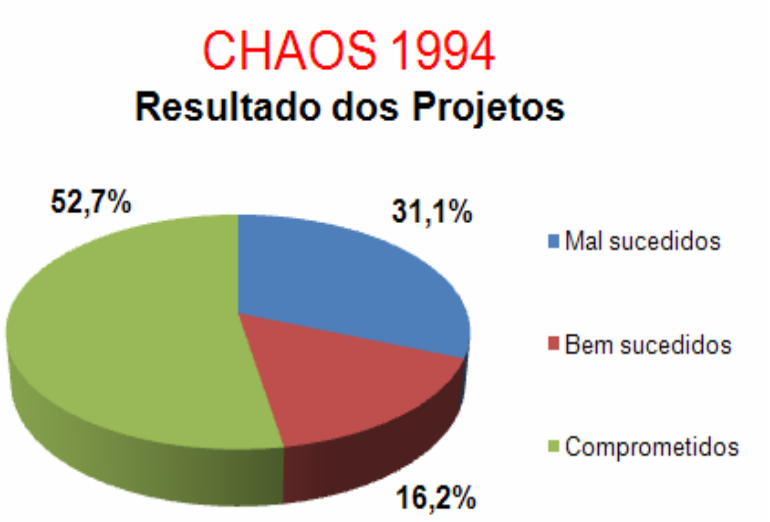
Custo hardware x software
1970 = 8:2
1991 = 2:8
Hoje = 1:9

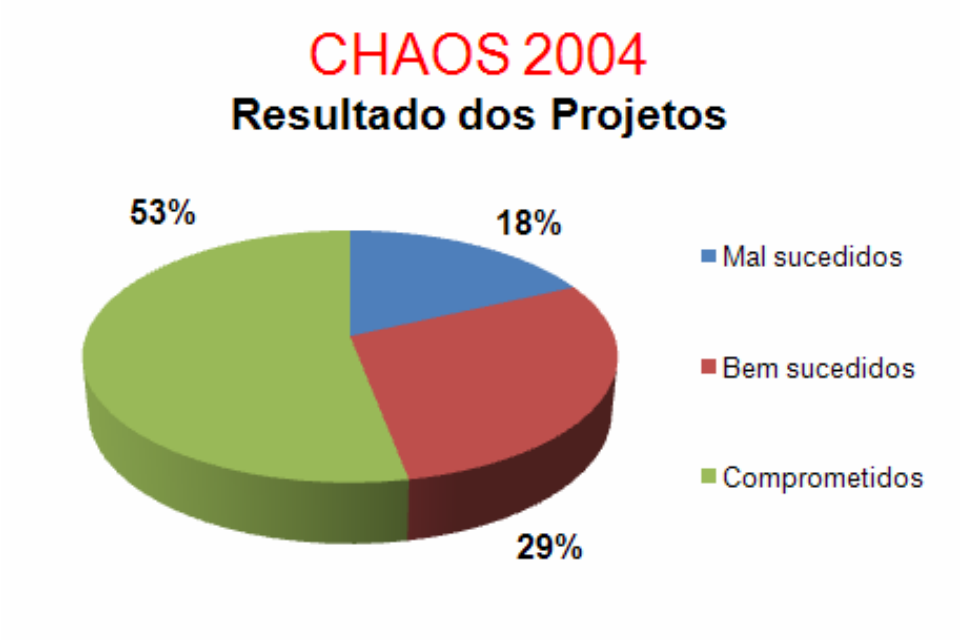
Aqui temos a proporção do custo de hardware e software durante o passar dos anos. Podemos perceber que o software passou a ser a parte mais cara na construção de uma solução computacional, pois o hardware foi ficando mais barato durante os anos e o software foi se tornando cada vez mais complexo.

No relatório Chaos, os projetos são enquadrados em três categorias distintas:

- **Mal sucedido** – o projeto é cancelado em algum momento do desenvolvimento por uma ou mais razões;
- **Bem sucedido** – o projeto é concluído dentro do prazo previsto e do orçamento estimado;
- **Comprometido** – o projeto é concluído. Porém, é entregue com atraso, com orçamento além do estimado e, em alguns casos, o software não possui todas as funcionalidades especificadas.

O resultado dessa pesquisa pode ser verificado nos gráficos a seguir:





Do ponto de vista do cliente esse não é um resultado animador, visto que, em 2004, 71% dos projetos tiveram algum tipo de problema. A evolução dessa pesquisa do Chaos não é animadora, como pode ser observado na tabela a seguir:

	2011	2012	2013	2014	2015
Bem sucedidos	29%	27%	31%	28%	29%
Comprometidos	49%	56%	50%	55%	52%
Mal sucedidos	22%	17%	19%	17%	19%

Podemos ver que, apesar do avanço dos anos, as porcentagens se mantêm bem próximas dos resultados obtidos em 2004. Por isso esta disciplina se torna tão importante, precisamos entender os problemas encontrados no mercado para evitarmos repetir as mesmas falhas e para que os projetos em que trabalharmos estejam dentro da porcentagem dos projetos bem sucedidos.

Abaixo listo os problemas mais comuns no desenvolvimento de software:

- Estimativas de prazo e de custo imprecisas.
- Custos acima do previsto.
- A facilidade de manutenção não era enfatizada como um critério importante, gerando, assim, custos de manutenção elevados.
- Não atendimento dos requisitos do usuário.

Para que possamos exemplificar melhor um caso de projeto mal sucedido, apresento aqui o projeto Ariane 5 - um Projeto Espacial da agência Europeia que custou US$ 8 bilhões. Ele demorou um total de 10 anos para ser desenvolvido e, por uma falha, aconteceu uma explosão 40 segundos após a decolagem, acarretando a destruição do foguete com carga avaliada em mais de US$ 480 milhões.

Exercícios

1- Explique os resultados que o relatório Chaos apresenta nos anos de 1994 e 2004.

2 - Cite os problemas mais comuns no desenvolvimento de software.

3 - Descreva um software de seu interesse, apresente um cronograma para seu desenvolvimento e um orçamento para seu desenvolvimento.

2. Processo desenvolvimento de software

Processo (do latim *procedere*) é um verbo que indica a ação de avançar, ir para frente (pro+cedere) e é um conjunto sequencial e particular de ações com objetivo comum. Pode ter os mais variados propósitos: criar, inventar, projetar, transformar, produzir, controlar, manter e usar produtos ou sistemas.

Um processo de software é um conjunto de atividades que levam à produção de um produto de software. Essas atividades podem envolver seu desenvolvimento a partir do zero em qualquer linguagem de programação.

Existem vários processos de software diferentes, mas todos devem incluir quatro atividades fundamentais.

- **Especificação de software;**
- **Projeto e implementação de software;**
- **Validação de software;**
- **Evolução de software.**

Especificação de software: detalha as funcionalidades do software, suas restrições e o seu funcionamento.

Projeto e implementação: o software deve, necessariamente, ser produzido para atender às especificações.

Validação: o software deve ser validado para garantir que atenda às demandas do cliente.

Evolução: o software deve evoluir para atender as necessidades de mudança do cliente.

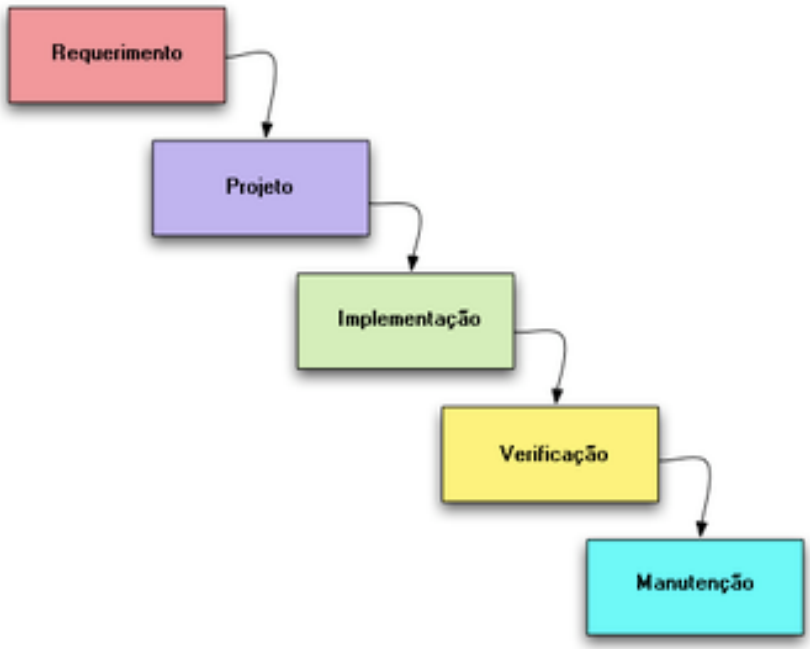
Para que tudo funcione de maneira correta, é necessário que todas as atividades sejam muito bem organizadas e, para tanto, se fazem necessárias algumas definições:

Produtos: resultado de uma ou mais atividades de processos.

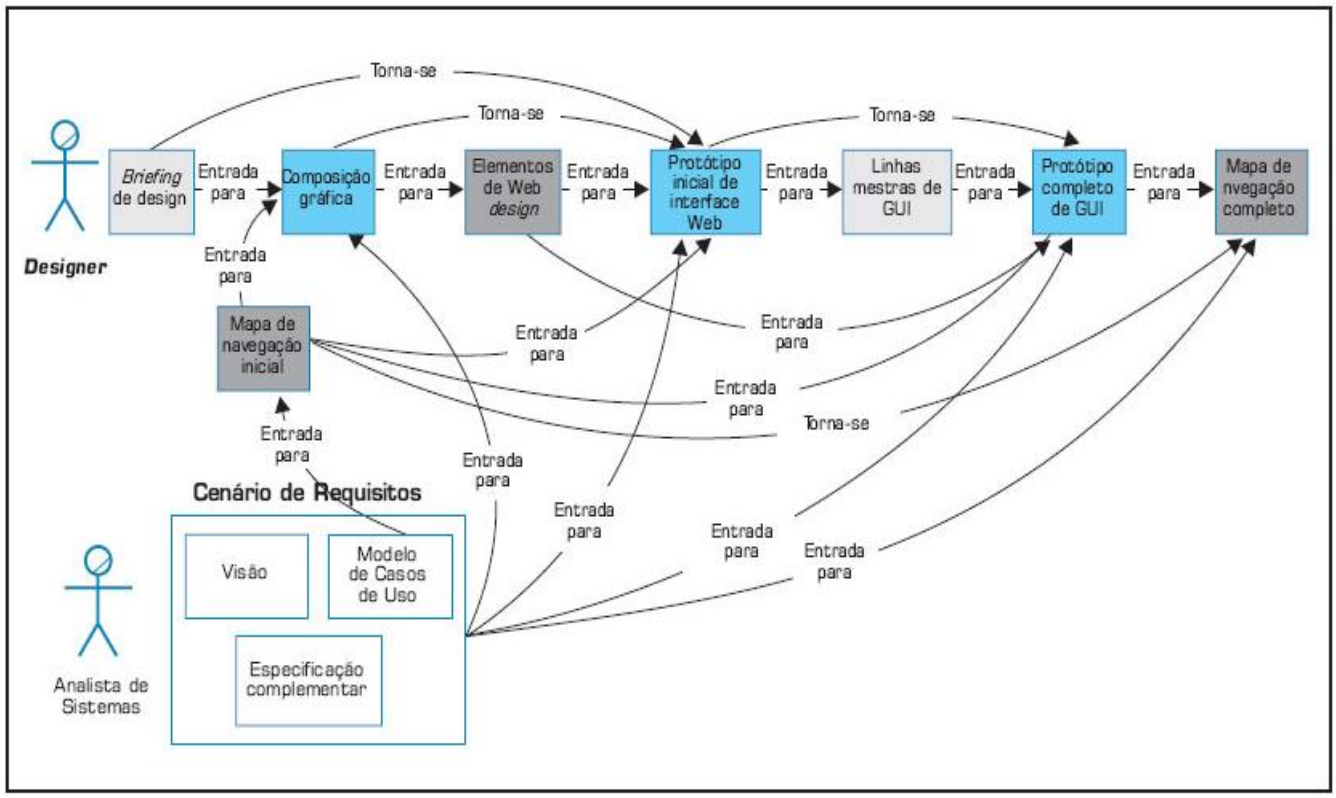
Papéis: são as responsabilidades das pessoas envolvidas nos processos.

Pré e pós-condições: declarações (requisitos) verdadeiras de antes e depois de uma atividade.

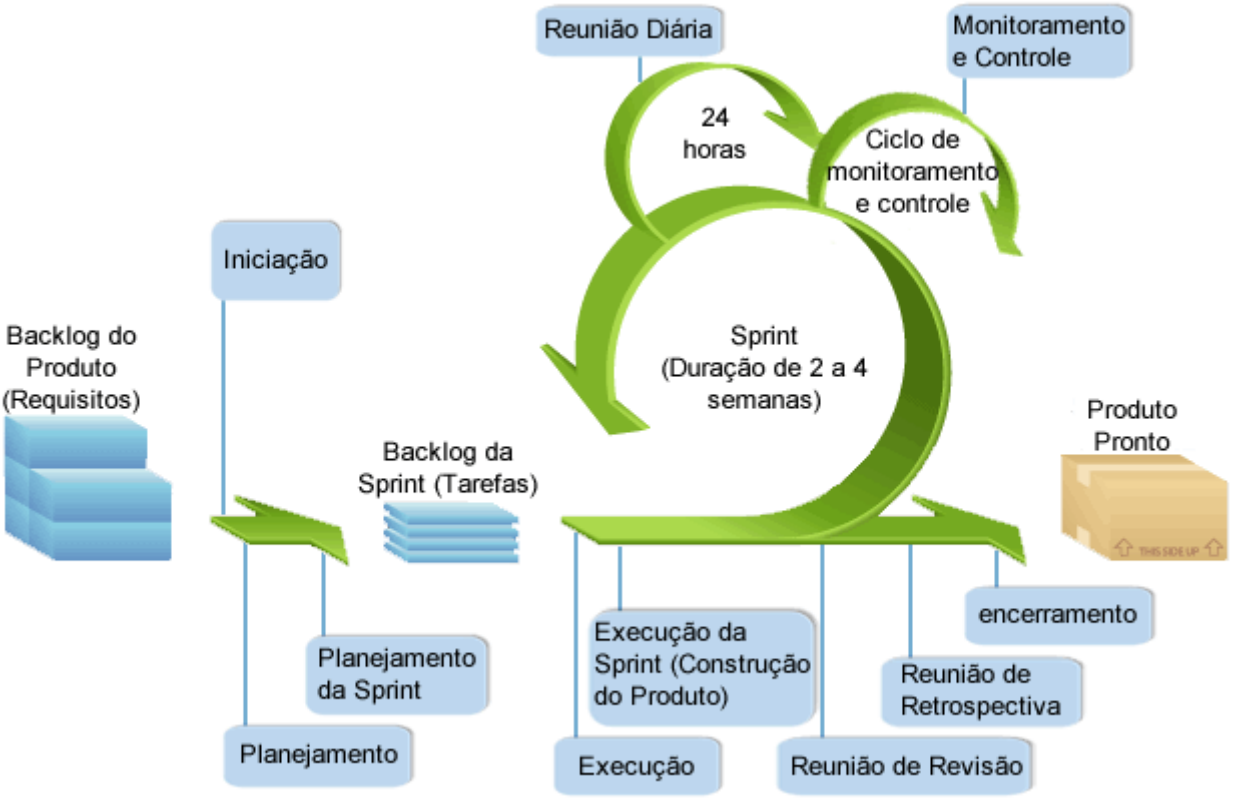
A seguir são apresentados diferentes processos de softwares utilizados no mercado. Cada um tem sua peculiaridade e são utilizados em situações diferentes por empresas diferentes.



Esse processo segue bem à risca as atividades fundamentais apresentadas anteriormente. Nele pode-se perceber que cada etapa se encaixa diretamente em uma das atividades fundamentais do desenvolvimento de software. Estas atividades são complexas e muitas das vezes existem outras sub-atividades dentro das principais.



Esse é o processo utilizado no *Rational Unified Process* (RUP) para a integração do processo criativo de design. Ele representa apenas uma pequena parte das atividades gerais do processo. Gosto de usá-lo para mostrar que um processo pode ter etapas minuciosamente detalhadas, como representado nessa imagem, além de envolver mais de uma pessoa, que são representadas pelos bonecos, e, conforme definimos anteriormente, são representados pelos seus papéis.



Essa é a representação do processo utilizado pelo Scrum, nele podemos ver que vários ciclos estão presentes. Estes ciclos representam atividades e etapas que são executadas repetidamente.

Vendo esses diferentes processos, podemos chegar às seguintes conclusões:

- Não existe processo ideal;
- Cada organização tem o seu processo;
- Processos evoluem;
- Necessitam de pessoas;
- Padronização é importante;

Exercícios:

1 - Escreva um processo para alguma atividade de seu interesse.

2 - Defina “produtos”, “papéis”, “pré” e “pós-condições” para o processo que você definiu na atividade anterior.

3 - Explique as atividades fundamentais no desenvolvimento de software.

3. Modelos de processos de desenvolvimento de software

Modelos são representações simplificadas de um processo de software. Cada modelo representa uma perspectiva particular de um processo e, portanto, fornece informações parciais. Vamos conhecer dois modelos genéricos e amplamente conhecidos na área de Engenharia de Software.

- Modelo em Cascata;
- Modelo incremental;

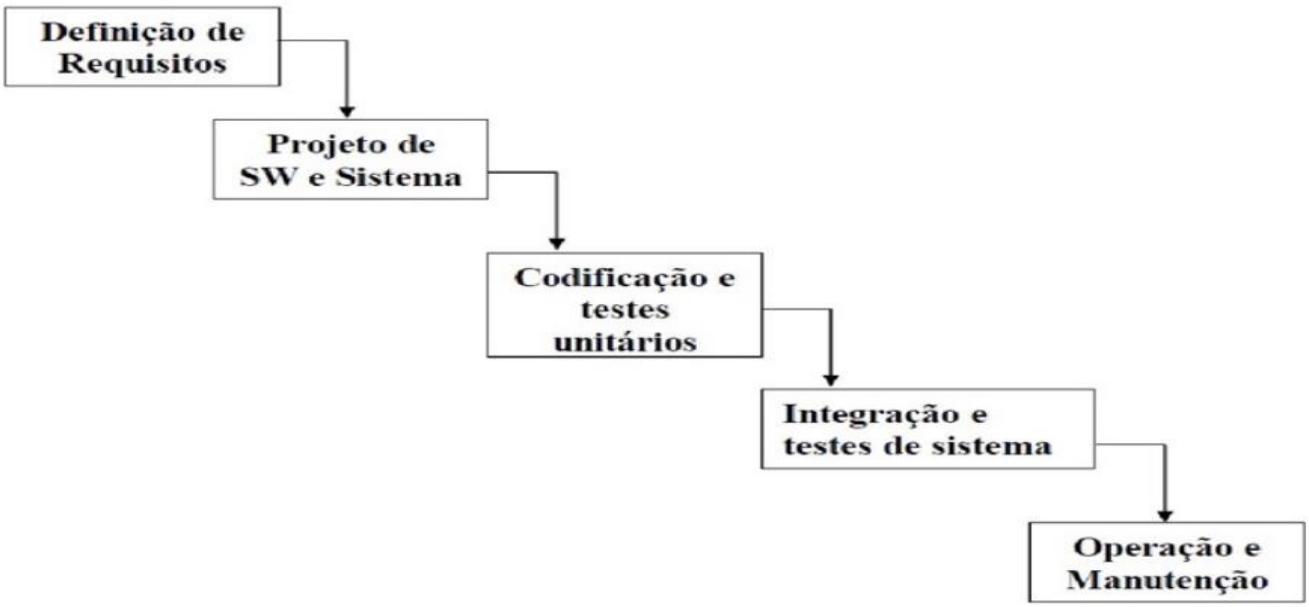
3.1. Modelo em cascata

O Modelo em Cascata é um modelo sequencial de desenvolvimento de software, no qual o processo é visto como um fluir constante para frente, da mesma forma que uma cascata, através das fases de análise de requisitos, projeto, implementação, validação, integração, e manutenção de software.

Para seguir o modelo em cascata, o progresso de uma fase para a próxima se dá de forma puramente sequencial. Por exemplo, inicialmente completa-se a especificação de requisitos. O software em questão é projetado, esse projeto deve ser um plano para implementação dos requisitos dados. Quando o projeto está terminado, uma implementação para este projeto é feita pelos codificadores. Encaminhando-se para o próximo estágio da fase de implementação, inicia-se a integração dos componentes de software construídos por diferentes pessoas do projeto. Após as fases de implementação e integração estarem completas, o produto de software é testado e qualquer problema introduzido nas fases anteriores é removido

aqui. Com isso o produto de software é instalado e, mais tarde, mantido pela introdução de novas funcionalidades e remoção de defeitos.

O modelo em cascata move-se para a próxima fase somente quando a fase anterior está completa e perfeita. Desenvolvimento de fases no modelo em cascata são discretas, não há pulo para frente, para trás ou sobreposição entre elas.



No modelo em cascata as fases são encadeadas. Trata-se de um modelo dirigido a planos - em princípio, cada etapa deve ser cuidadosamente planejada antes de se começar a trabalhar nelas. Cada estágio deve ser finalizado para que o seguinte possa iniciar. Todas as etapas são muito bem definidas e distintas entre si.

Etapas

Definição de requisitos: as necessidades, restrições e metas do sistema são estabelecidas por meio de uma consulta aos usuários. Em seguida, são definidos os detalhes e uma especificação do sistema.

Projeto de sistema e software: o processo de projeto de sistemas aloca os requisitos, tanto para sistemas de hardware como para sistemas de software, por meio da definição de uma arquitetura geral do sistema.

Codificação e testes unitários: durante esse estágio, o projeto do software é desenvolvido como um conjunto de programas ou unidades de programa. O teste unitário envolve a verificação de que cada unidade atenda a sua especificação.

Integração e testes de sistema: as unidades individuais do programa são integradas e testadas como um sistema completo para assegurar que os requisitos do software tenham sido atendidos. Após os testes o sistema é entregue ao cliente.

Operação e Manutenção: uma das fases mais longas de todo o processo de desenvolvimento de um software. O sistema é instalado e colocado em uso. A manutenção envolve correção de erros que não foram descobertos nos estágios iniciais, a melhora da implementação e a ampliação dos serviços.

Conclusões

- ✓ Cada etapa gera um ou mais documentos;
- ✓ A etapa seguinte não deve ser iniciada até que a anterior esteja concluída;
- ✓ Cada etapa alimenta a seguinte com informações;
- ✓ Apesar de ser um modelo linear, o feedback de uma fase para outra é necessário;
- ✓ Altos custos de produção;
- ✓ Burocrático;
- ✓ Iterações constantes podem gerar falhas;
- ✓ O software só é colocado em uso no final do processo e alguns problemas podem ser incontornáveis;
- ✓ Deve ser utilizado somente quando os requisitos são bem definidos.

3.2. Modelo Incremental

O desenvolvimento incremental é baseado na ideia de desenvolver uma implementação inicial, expô-la aos interessados e continuar por meio da criação de várias versões, até que um sistema adequado seja desenvolvido.

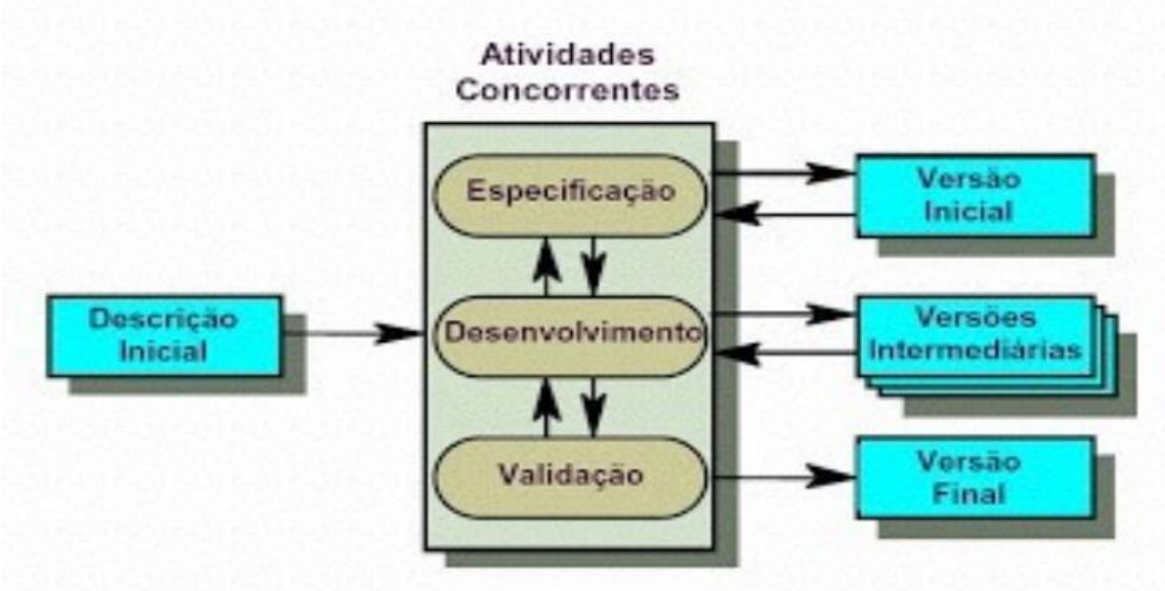
O Desenvolvimento Incremental é um dos clássicos modelos de processo de desenvolvimento de software, criado em resposta às fraquezas do modelo em cascata, que é o mais tradicional. Os dois padrões

mais conhecidos de sistemas iterativos de desenvolvimento são o RUP (*Processo Unificado da Rational*) e o Desenvolvimento Ágil de Software. Nesta disciplina não abordaremos o desenvolvimento ágil de software, mas, é importante ressaltar que o desenvolvimento incremental é parte essencial do desenvolvimento ágil.

Atividades:

- ✓ Especificação;
- ✓ Desenvolvimento;
- ✓ Validação.

São intercaladas, não separadas, e com rápido feedback entre todas as atividades.



Para a maioria dos sistemas, o Desenvolvimento Incremental é melhor que uma abordagem em cascata, pois reflete a maneira como resolvemos problemas em nossas vidas, principalmente por ter entregas constantes e permitir mudanças.

Cada incremento gera uma versão do sistema e incorpora funções necessárias para o cliente. Frequentemente, os incrementos iniciais incluem as funcionalidades mais importantes para o cliente. Isso significa que as avaliações iniciais podem identificar possíveis problemas e, caso algo incoerente seja detectado, a funcionalidade deverá ser redefinida, ou seja, volta-se para o estágio inicial.

Vantagens

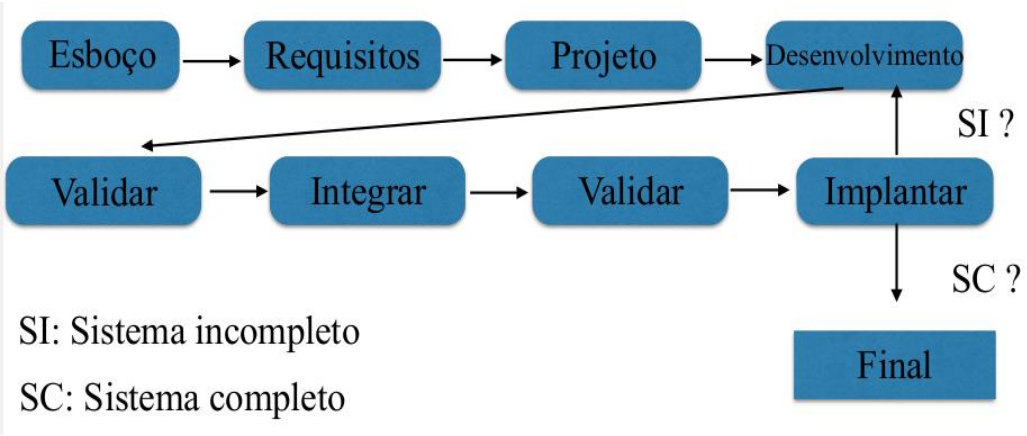
O desenvolvimento incremental tem três vantagens quando comparado ao modelo em cascata:

- O custo de acomodar as mudanças nos requisitos do cliente é reduzido. A quantidade de análise e documentação a se feita é menor.
- É mais fácil obter o feedback dos clientes sobre o desenvolvimento do que foi feito.
- É possível obter entrega e implementação rápida de um software útil ao cliente.

Desvantagens

Apesar dos grandes benefícios, o desenvolvimento incremental não é isento de problemas, veja a seguir:

- O processo não é visível;
- Dificuldades em mensurar o projeto;
- Estrutura do sistema tende a se degradar;
- Refatoração e melhorias são necessárias.



Uma vez que os incrementos do sistema tenham sido identificados, os requisitos dos serviços a serem entregues são definidos em detalhes, e esse incremento é desenvolvido. Durante o desenvolvimento, mais análises podem ser realizadas, contudo, os requisitos atuais não mudam. Quando um incremento é concluído e entregue, os clientes podem colocá-lo em operação. Isso nada mais é que a entrega de uma parte do sistema. Os clientes podem experimentar o sistema e isso os ajuda a compreender suas necessidades para incrementos posteriores.

Vantagens

- Os clientes podem usar os incrementos iniciais;
- Não é necessário esperar a conclusão do sistema;
- Facilita a incorporação de recursos;
- As partes importantes são entregues primeiro.

Desvantagens

- Quando está sendo desenvolvido um sistema substituto;
- São necessários recursos básicos;
- Dificuldades no contrato;

Exercícios:

1 - Crie um processo que utilize o modelo cascata para executar alguma atividade de seu interesse.

2 - Crie um processo que utilize o modelo incremental para executar alguma atividade de seu interesse.

3 - Escolha algum origami e monte um processo para poder fazer as dobraduras. Qual o modelo que ele segue?

4 - Cite as vantagens e desvantagens do modelo cascata.

5 - Cite as vantagens e desvantagens do modelo incremental.

4. Gerência de projetos

Gerência de projetos é a área da Administração que aplica os conhecimentos, as habilidades e as técnicas para elaboração de atividades relacionadas a um conjunto de objetivos pré-definidos, num certo prazo, com um certo custo e qualidade, através da mobilização de recursos técnicos e humanos.

A humanidade realiza projetos desde a antiguidade, a gerência de projeto é um assunto antigo, porém, é considerado como ciência há pouco tempo. Somente no Século XX as técnicas de gestão passaram a ser utilizadas como ferramenta formal. Foi em 1969 que a Gestão de Projetos tornou-se uma ciência, com a fundação do PMI (*Project Management Institute* – Instituto de Gerenciamento de Projetos).

Quando abordamos o gerenciamento de projetos, ouvimos falar de algumas siglas como PMI, PMBOK e PMP, veja a descrição de cada uma delas a seguir:

- PMI (*Project Management Institute*):
  - ✓ Entidade mundial que tem como objetivo desenvolver e divulgar o gerenciamento de projetos como ciência.

- PMBOK (*Project Management Body of Knowledge*):
  - ✓ Trata-se de uma publicação emitida pelo PMI;
  - ✓ Manual de boas práticas, reconhecido mundialmente como a principal publicação a respeito de Gerenciamento de Projetos;
  - ✓ É um guia genérico e pode ser aplicado em projetos de todas as áreas;
  - ✓ Atualmente está na 6ª edição, lançada em janeiro de 2018.

- PMP (*Project Management Professional*):
  - ✓ Certificação profissional concedida pelo PMI;
  - ✓ Comprovante de conhecimento formal nas práticas difundidas pelo PMI;
  - ✓ Comprovante de experiência prática em gerenciamento de projetos.

Existe uma preocupação diferente quando se fala de gerência de projeto de software, por isso temos o cuidado de discutir a gerência de projetos de software dentro de um curso como este. Isso acontece porque os softwares têm algumas peculiaridades, por serem abstratos e intangíveis. Os softwares não são restringidos por:

- ✓ Propriedades materiais;
- ✓ Leis da física; ou
- ✓ Processos de manufatura.

Devido à falta de restrições, os softwares podem se tornar complexos rapidamente, difícil de entender e caros para alterar.

Na gerência de projetos de software, temos que levar em conta que eles são muito diversificados, há vários tipos de software: Sistema de informações corporativo; Controlador para um instrumento específico; Jogo computacional; entre outros. Desta forma, precisamos ter a preocupação que diferentes tipos de software necessitam de abordagens diferentes de gerenciamento. Essa abordagem diferente começa pela escolha do processo de desenvolvimento, envolve a escolha da equipe e se preocupa até com as tecnologias envolvidas em seu desenvolvimento.

Existe uma cultura em que softwares amadores ou pouco trabalhados são entregues no mercado. Essa cultura leva muitos programadores a escreverem softwares e a se aventurarem no mercado, vendendo softwares com soluções problemáticas. Isso acontece devido ao despreparo das pessoas que procuram soluções de software. Nós, programadores responsáveis, devemos garantir que alguns cuidados sejam tomados para que o software obtenha sucesso e possa ser bem aproveitado pelo cliente.

Vamos diferenciar dois tipos de software, o software comum feito sem muita preocupação, e software profissional, que é o software que todos serão capazes de produzir ao final deste curso.

- Software Comum
  - ✓ Criado para simplificar seu trabalho;
  - ✓ Escrito apenas por hobby;
  - ✓ Usado para diversão ou interesse próprio.

- Software Profissional
  - ✓ Possui um propósito específico de negócio;
  - ✓ Usado por alguém além do seu desenvolvedor;
  - ✓ Criado por equipes;
  - ✓ É mantido e alterado durante sua vida.

O software profissional é melhor recebido pelo mercado, pois foi produzido com objetivo e propósito específico, normalmente foi criado por uma equipe e recebe manutenção durante o tempo em que é utilizado. Um software profissional deve contemplar:

- O programa em si;
- Documentação associada;
- Dados de configurações;
- Possivelmente um site sobre o produto;
- Diagramas e esquemas usados durante o desenvolvimento.

Em resumo, podemos dizer que a gestão de projeto é a aplicação de técnicas, conhecimento e habilidades para garantir que um projeto tenha sucesso. Ela é especialmente necessária na gestão de projetos de software, pois o software é algo abstrato e intangível. Podemos falar que a Gestão de Projetos é importante, pois, auxilia na entrega das seguintes características:

- Entrega no prazo definido, respeitando começo e fim;
- Planejamento, execução e controle do projeto;
- Entrega de produtos, serviços ou resultados exclusivos;
- Desenvolvimento por etapas e com evolução progressiva;
- Gerência da equipe de profissionais;
- Gerência de recursos disponíveis.

4.1. Requisitos

Requisitos são funções, objetivos, propriedades e/ou restrições que o sistema deve possuir para satisfazer contratos, padrões ou especificações, de acordo com os usuários. Podemos dividir os requisitos em duas categorias: os requisitos funcionais e os requisitos não funcionais.

- **Requisitos funcionais**: os requisitos funcionais referem-se ao que o sistema deve fazer, ou seja, suas funções e informações. Ele é o conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados.

- **Requisitos não funcionais**: os requisitos não funcionais referem-se aos critérios que qualificam os requisitos funcionais. Esses critérios podem ser de qualidade para o software, ou seja, os requisitos de performance, usabilidade, confiabilidade, robustez, etc.

Além de requisitos de software, outra palavra que ouviremos bastante é “escopo”. Escopo é a finalidade, o alvo ou o intento que foi estabelecido como meta final. O escopo é o objetivo que se pretende atingir, é sinônimo de fim, propósito ou desígnio. Em gerenciamento de projetos, escopo é a soma total de todos os produtos do projeto e seus requisitos ou características.

4.2. Restrições

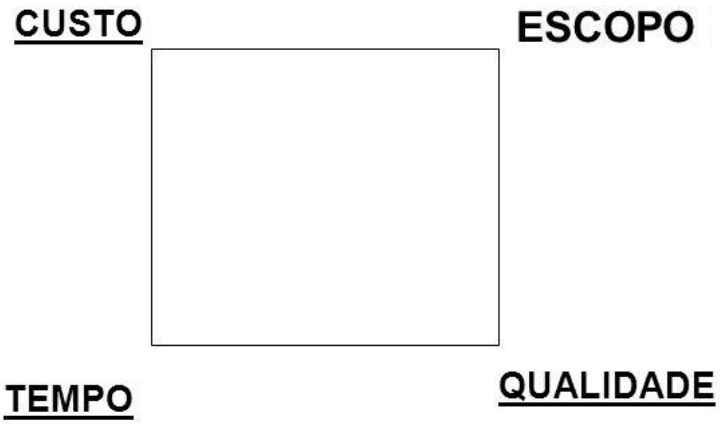
Triângulo de restrições

Uma grande preocupação do gerente e da equipe do projeto deve ser com o gerenciamento da restrição tripla - escopo, tempo e custo do projeto - de necessidades conflitantes do projeto. Considera-se ainda a qualidade do projeto como uma quarta variável, a qual é afetada pelo balanceamento das três demais.

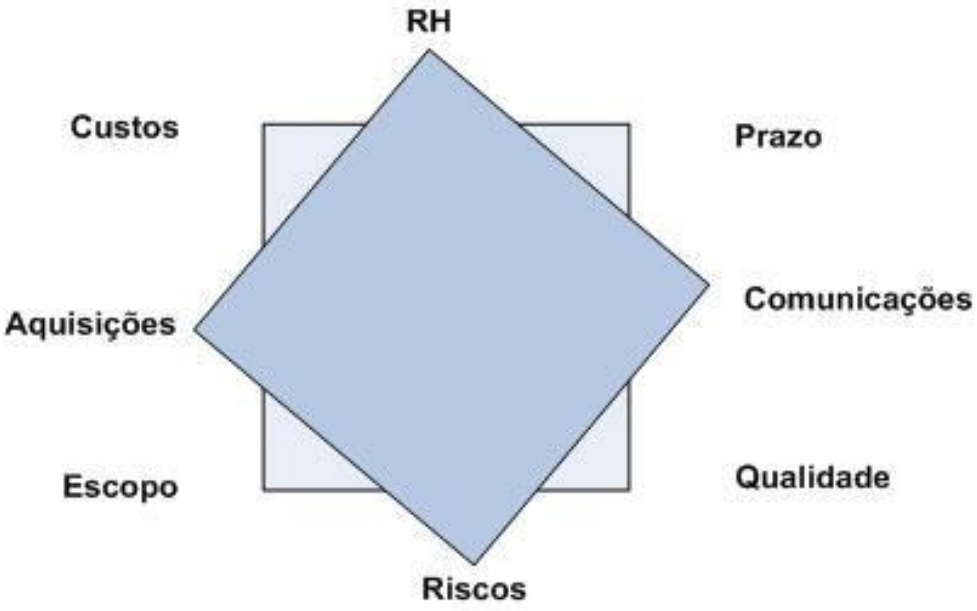
Dentro do projeto existe um equilíbrio natural entre essas quatro forças, o qual é estabelecido no momento em que as linhas de base de escopo, tempo e custo são acordadas entre as partes envolvidas na realização do projeto. A partir desse ponto, mudanças em uma dimensão serão refletidas em uma ou mais entre as três outras.

Por exemplo, para atender a uma solicitação do cliente por um aumento no escopo, os custos e/ou o prazo do projeto deverão ser aumentados, caso contrário, a qualidade final do projeto poderá ser afetada pela necessidade de reduzir a quantidade de testes, de utilizar materiais mais baratos e, possivelmente, de menor qualidade, contratação de equipes mais baratas e menos experientes, etc.

Muitos pesquisadores acreditam que a qualidade também pode ser uma variável nesta equação. E o triângulo de restrições pode, na verdade, ser apresentado no formato de um quadrado, onde a qualidade se torna uma força dentro do projeto e pode ser redimensionada para se ter resultados de acordo com o que foi acordado entre as partes interessadas.



Apesar dessa representação do quadrado ser mais completa, existem várias outras forças atuando e tanto o gerente de projetos quanto a equipe devem acompanhar. Essas forças são várias e uma representação mais completa pode ser vista na imagem abaixo, onde oito características são apresentadas.



Assim, para fins didáticos, o triângulo de restrições é bastante difundido e utilizado, mas é muito importante entender que são várias as forças e fatores que influenciam no resultado de um projeto.

4.3. Riscos

Os riscos de projeto são um conjunto de eventos que podem ocorrer sob a forma de ameaças ou de oportunidades que, caso se concretizem, influenciam o objetivo do projeto, negativamente ou positivamente.

O gerenciamento dos riscos é uma tarefa importante atribuída ao gerente de projetos e envolve a antecipação de riscos e a precauções para evitá-los.

Um risco pode ser pensado como algo que você preferiria que não acontecesse. E os riscos podem comprometer: o projeto; o software; ou a organização.

Eles podem ser subdivididos nas seguintes categorias e podem, ainda, se sobrepor:

- Riscos de projeto:
  - ✓ Afetam cronograma e recursos de projeto.
  - ✓ Perda de um projetista experiente.
- Riscos de produto:
  - ✓ Afetam a qualidade ou desempenho do software que está sendo desenvolvido.
  - ✓ Falha em componente comprado, que afeta o desempenho de todo o sistema.
- Riscos de negócio:
  - ✓ Afetam a organização que desenvolve ou adquire o software.
  - ✓ Concorrente que introduz um novo produto.
  - ✓ Queda nas vendas do novo produto.

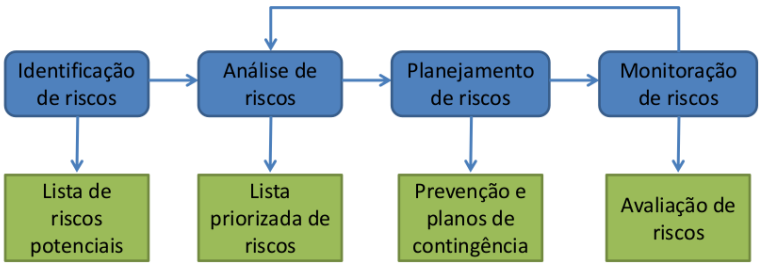
A tabela abaixo mostra alguns exemplos de riscos:

Risco	Afeta	Descrição
Rotatividade de pessoal	Projeto	Funcionários experientes deixam a equipe
Indisponibilidade de hardware	Projeto	Hardware não será entregue no prazo
Mudança nos requisitos	Projeto e produto	Número maior de alterações do que o previsto
Tamanho subestimado	Projeto e produto	O tamanho do sistema foi subestimado
Componente de terceiros com problemas	Produto	Componente adquirido de terceiros apresenta falhas
Mudança de tecnologia	Negócio	A tecnologia usada na construção do sistema é substituída
Concorrência de produto	Negócio	Produto concorrente aparece no mercado

Podemos dividir o processo de gerenciamento de riscos em 4 estágios:

1. Identificação de riscos: identificar riscos de projeto, de produto e de negócios.
2. Análise de riscos: avaliar probabilidade e consequências dos riscos.
3. Planejamento de riscos: como enfrentar, seja evitando ou minimizando seus efeitos.
4. Monitoramento de riscos: avaliar regularmente os riscos e planos para mitigação, atualizando caso necessário.

Este o processo de gerenciamento de riscos é algo iterativo e deve ocorrer ao longo do projeto. Primeiro, elabora-se um plano de gerenciamento; depois, monitora-se esse plano; em seguida, reavaliam-se os riscos do plano; e, por último, mudam-se os planos de prevenção e gerenciamento de contingência.



Os riscos podem ser analisados em equipe. O gerente pode usar sua experiência, mas, com todos membros da equipe, a chance de contemplar uma quantidade maior de riscos é maior.

Para facilitar este levantamento, pode ser usado um checklist de riscos:

- Riscos de tecnologia;
- Riscos de pessoas;
- Riscos organizacionais;
- Riscos de ferramentas;
- Riscos de requisitos;
- Riscos de estimativas.

Tipo de risco	Possíveis riscos
Tecnologia	O banco não suporta tantas transações quanto esperado. (1) Componentes reusáveis contém defeitos e não poderão ser usados, conforme planejado. (2)
Pessoas	Impossível recrutar pessoas com habilidades necessárias. (3) Principais pessoas estão doentes e não estarão presentes em momentos críticos. (4) Treinamento para o pessoal não está disponível. (5)
Organizacional	Organização é reestruturada para que gerentes coordenem vários projetos. (6) Problemas financeiros da organização forçam reduções no orçamento. (7)
Ferramentas	Código gerado pela ferramenta automatizada é ineficiente. (8) Ferramentas de software não podem trabalhar de forma integrada. (9)
Requisitos	Propostas de mudanças de requisitos exigem retrabalho de projeto. (10) Clientes não conseguem compreender impacto nas mudanças de requisitos. (11)
Estimativa	Tempo necessário para desenvolver é subestimado. (12) Tamanho do software é subestimado. (13)

Como considerar a probabilidade de ocorrência de cada risco?

- Não há regras nem fórmulas para atribuir a um risco uma probabilidade de que ele ocorra;
- É uma atribuição subjetiva, baseada em julgamento;
- A experiência advinda de projetos anteriores e de problemas que surgiram ajuda na análise.

Uma sugestão de como avaliar a probabilidade é:

Probabilidade de Ocorrência	
Muito baixa	< 10%
Baixa	10 a 25%
Moderada	25 a 50%
Alta	50 a 75%
Muito alta	>75%

Gravidade	
Catastrófico	Ameaçam o projeto
Grave	Causariam grandes atrasos
Tolerável	Atrasos dentro da margem
Insignificante	Não causam danos

Risco	Probabilidade	Efeito
7	Baixa	Catastrófico
3	Alta	Catastrófico
4	Moderada	Grave
2	Moderada	Grave
10	Moderada	Grave
6	Alta	Grave
1	Moderada	Grave
12	Alta	Grave
9	Alta	Tolerável
11	Moderada	Tolerável
5	Moderada	Tolerável
13	Alta	Tolerável
8	Moderada	Insignificante

O processo de planejamento considera os riscos identificados e elabora estratégias para gerenciá-los.

São estratégias de gerenciamento de riscos:

- Estratégias de prevenção;

- Tenta reduzir a probabilidade de ocorrência do risco;
- Estratégias de minimização;
- Diminui o impacto do risco;
- Planos de contingência;
- Estratégia para lidar com o pior.

Risco	Estratégia
Problemas financeiros *Contingência*	Crie um documento e apresente à gerência, mostrando a contribuição do projeto para os objetivos do negócio, apresentando motivos para evitar cortes de orçamento.
Doença de pessoal *Minimização*	Reorganize a equipe para que as pessoas compreendam o trabalho umas das outras.
Componentes defeituosos *Prevenção*	Substitua os componentes falhos por outros de confiabilidade conhecida.
Desempenho do banco de dados	Analise a compra de um banco de dados de alto desempenho.

Os riscos devem ser verificados a cada etapa do projeto. Sempre verificando se os efeitos do risco ainda prevalecem ou se novos riscos surgem durante o desenvolvimento do projeto.

4.4. Contratos

A dúvida de qual modelo de contratação é mais adequada para seu projeto é sempre algo que acompanha os gerentes de projetos, desde os contratos fechados e tradicionais, com escopo, prazo e custos bem definidos, aos mais abertos, que permitem a prestação de serviços com foco na entrega de valor para o projeto e com escopo flexível. Nesta seção iremos estudar três modelos e detalhar o que é importante você saber no momento da negociação.

Os tipos de contratos que iremos ver são:

- Preço fixo;
- Escopo negociável;
- Tempo e material.

Preço fixo

Também conhecido como Contrato de Preço Global ou Preço Fixo Garantido, em que o comprador paga ao fornecedor um valor determinado (definido em contrato), independentemente dos custos do fornecedor. Trata-se do tipo mais comum de contrato e é mais utilizado para adquirir produtos ou serviços com requisitos bem definidos. Isso permite reduzir os riscos do fornecedor e criar estimativas de tempo e custos mais precisas, sendo determinante para a obtenção de um preço final justo e razoável.

Permite ao comprador oferecer incentivos ao fornecedor por metas alcançadas no projeto, como, por exemplo, antecipação de entregas. Nesse tipo de contrato, o comprador não tem conhecimento do lucro do fornecedor, assim como exige um menor esforço de gerenciamento do trabalho a ser realizado.

- Estrutura: Acordados os entregáveis, entregue-os. Mande a fatura. Os clientes gostam de projetos de preço fixo porque lhes dá segurança.

- Risco: Claramente o risco está do lado do fornecedor. Se as estimativas estiverem erradas, ele irá perder o dinheiro. Riscos menos óbvios são as solicitações de mudança, pelo qual o fornecedor irá negociar receita adicional pelas mudanças de escopo. Se o fornecedor tiver subestimado o esforço em um preço não realista, as perdas ameaçam a existência do fornecedor, o que representa um problema para o cliente.

- Relacionamento: Competitivo a indiferente. Clientes, normalmente, querem receber mais e o fornecedor quer fazer menos. O fornecedor quer que o cliente fique satisfeito, então, normalmente produz. A palavra “et cetera” é muito perigosa em uma especificação com preço fixo.

Escopo negociável

No processo de desenvolvimento tradicional, que usa como base o modelo em cascata, é comum fechar o escopo do que será entregue logo no início do projeto, antes que se tenha uma ideia clara de tudo que será feito, ou pior, antes de saber qual o esforço necessário para realizar a entrega. Pode-se usar o histórico de projetos passados similares e quanto tempo foi usado para a resolução daquele problema, mas, mesmo assim, sempre existe a possibilidade de desvios.

Quando esses desvios são para mais, a empresa que desenvolveu o software sairá no lucro, mas provavelmente não fechará novos projetos com o cliente, pois este se sentirá lesado por pagar um valor bem maior que o correto. Se o desvio foi para menos, houve um erro de cálculos e a empresa que desenvolveu o sistema sairá no prejuízo. Dependendo do tamanho dele, pode não existir uma segunda chance para recuperar o dinheiro perdido.

Essa tentativa de acertar quanto será gasto logo nos primeiros encontros, para colocar em contrato as responsabilidades de cada um, gera grandes níveis de estresse de ambos os lados e acaba, muitas vezes, criando situações políticas para se conseguir um novo cliente, mantê-lo ou simplesmente baixar um pouco mais o preço do serviço que se está comprando.

Usando o modelo incremental, o escopo dos projetos pode ser variável, ou seja, o cliente, em comum acordo com a equipe de desenvolvimento e os responsáveis pelo projeto, estabelecem quais são as

prioridades na construção do sistema para que a equipe comece a trabalhar. A equipe de desenvolvimento se compromete a realizar o trabalho no tempo correto para finalizar a funcionalidade, nem mais, nem menos, numa relação de transparência.

A equipe que controla o tempo e esforços alocados ao projeto fica responsável por dosar o que pode ser feito e por replanejar tudo que não couber dentro do tempo acordado inicialmente. Caso uma funcionalidade não possa ser feita, ele tem que ter a coragem para explicar a situação para o cliente, que, devido ao seu histórico, deverá descartar tudo aquilo que não lhe seja de extrema urgência naquele momento.

Tudo isso só pode ser feito suportando numa relação de confiança e transparência. Problemas acontecerão de qualquer forma. O que muda é a forma de encará-los e resolvê-los. Ao ter o escopo variável, uma das quatro variáveis presentes em todos os projetos (relembrando: custo, prazo, escopo e qualidade) torna-se flexível, ou seja, três delas não precisam ser mexidas, mantendo-se o cronograma inicialmente estabelecido, dentro do custo planejado e com o nível de qualidade inicialmente pensado.

Infelizmente, esse tipo de abordagem exige tanto uma equipe técnica madura, quanto um cliente aberto a esse tipo de comportamento, pois ele tem um papel ativo durante o ciclo de vida do projeto e não apenas no momento da contratação e estabelecimento de requisitos.

No início acontecerão conflitos, os interesses de um lado e de outro se chocarão, mas os ganhos decorrentes desta mudança de postura são perceptíveis em longo prazo, com maior comunicação entre cliente e empresa, menos níveis de pressão e consequente melhora na qualidade do sistema, que será entregue por uma equipe que sabe que tem seu valor reconhecido.

Tempo e material

Conhecido também como Contrato de Preço Unitário, representa um tipo de contrato com características pertencentes tanto aos Contratos de Preço Fixo como aos Contratos de Custos Reembolsáveis.

Neste tipo de contrato o comprador paga por homem/hora ou por item adquirido. Semelhantemente ao Contrato de Custos Reembolsável, o valor final da aquisição é desconhecido no momento em que o contrato é firmado. Por outro lado, o preço fixo do trabalho homem/hora é conhecido.

Apresenta ainda termos e condições mais simples que os tipos apresentados anteriormente, tornando a negociação mais célere e agilizando o início do trabalho. É importante ressaltar que esse tipo de contrato deve ser utilizado apenas para projetos pequenos (curta duração), pois o fornecedor não possui incentivos para trabalhar com maior agilidade, já que ele é pago por cada hora trabalhada.

Parabéns!

Agora que você terminou nossa disciplina, você pode desenvolver software profissionalmente. Várias das técnicas de gestão de projetos podem ser utilizadas nos seus projetos de vida, então, não perca tempo e coloque em prática o que aprendeu na disciplina.

A Engenharia de Software é algo bem amplo, nesta disciplina não conseguimos abordar as metodologias ágeis de desenvolvimento de software, então, se você gostou do assunto, faça uma busca sobre o tema e aprofunde ainda mais no mundo da engenharia de software.

rede e-Tec Brasil

MEDIOtec