Conceitos de Engenharia de Software

Dejair José Pereira Junior

- Software
- Sistema de Software
- Característica do Software
- Objetivo de Software
- Ética
- Evolução Histórica
- A Crise de Software
- Desafios
- Elementos da Engenharia de Software
  - Princípios
  - Métodos
  - Ferramentas
  - Qualidade

1. O que é Software?

- Conjunto
  - Instruções /programa
  - Documentação associada
- Papel do Software
  - Produto
  - Veiculo de entrega desse produto (Processo)

2. O que é Sistema de Software?

Conjunto de elementos que interagem coesivamente, onde cada elemento pode ser um sistema.

- Fronteira conceitual para delimitar o sistema
- Relação dos elementos dentro do sistema devem ser fortes
- Relação com os elementos de fora da fronteira devem ser fracas

4. Natureza da Aplicação

- Software de sistemas
  - Serve outros programas.
  - Estrutura de Informação determinada- compiladores, editores
  - Estrutura de informação indeterminada (interação com HW)- sistemas operacionais e processadores de telecomunicações
- Sistema de tempo real
  - Monitora, analisa e controla eventos do mundo real á medida que eles ocorrem. Ex. sistema de segurança predial.
- Software comercial
  - Ou sistemas discretos como folha de pagamento, contas a receber. Hoje são integrados em sistemas de gerenciamento.

Natureza da Aplicação

- Software cientifico e de engenharia
  - Algoritmos que processam números como astronomia, vulcanologia, análise automotiva, biologia de molecular, manufatura automatizada
- Software embutido
  - Fica embutido e reside na memória principal. Serve para funções bem específicas como controle de teclado para um forno microondas.
- Software para computadores pessoais
  - Processador de texto, planilhas, multimídia, diversões.
- Software para a WEB
- Software para Inteligência artificial
  - Algoritmos não numéricos para problemas que não são passíveis de computação ou analise discreta. Sistema especialista, reconhecimento de padrões, jogos.

4. Características do Software

Outros Produtos	Software
Concreto e tangível	Abstrato e intangível
Restrito por materiais e regido por leis físicas	Não restrito por materiais ou regido por leis físicas
Desgasta	Deteriora
Manufaturado	Desenvolvido
Falhas no início do uso e quando se desgasta	Falhas no início e durante todo o ciclo devido as modificações
Componentes prontos	Reuso está no início

5. Objetivo da Engenharia de Software

Desenvolvimento de sistemas de software com uma boa relação custo- benefício, aumentando a qualidade, reduzindo prazos e custos

- á Qualidade
- â Prazo
- â Custo

Questoes abordadas pelo Engenheiro de Software

- Por que leva tanto tempo para concluir um software ?
- Por que os custos sao tao altos ?
- Por que os erros nao podem aparecer antes da entrega do software ao cliente ?
- Por que existe tanta dificuldade de controlar o processo enquanto o sw e desenvolvido ?.

6. Evolução Histórica da ES

- Década de 60
  - Software era uma arte secundária
  - Software específico para cada aplicação
  - Inexistência de documentação
  - 1968- encontro para discutir a crise do software
- Década de 70
  - Multiprogramação e multiusuários
  - Sistema de tempo real
  - 1a. Geração de SGBD
  - Biblioteca de Software
  - Aumenta número de sistemas

Evolução Histórica da ES

- Década de 80
  - Sistemas distribuídos
  - Redes locais e globais
  - Uso generalizado de microprocessadores - microeletrônica reduziu custo de hardware
  - Com a disseminação do uso de sistemas, os mesmo tornaram-se maiores e mais complexos. Um programador não conseguia mais trabalhar sozinho, era necessário uma equipe. Assim, a abordagem informal não sustentava mais a industria – atrasos nos prazos e custos muito maiores dos planejados
- 1990
  - Tecnologia orientada a objetos
  - Sistemas especialistas e software
  - Software de rede neural artificial
  - Computação paralela
  - Internet

7. A Crise de Software

- Estimativas de prazo e de custo imprecisas
  - “ Não dedicamos tempo para coletar dados sobre o processo de desenvolvimento de software’’
  - ‘‘ Sem nenhuma indicação sólida de produtividade, não podemos avaliar com precisão a eficácia de novas ferramentas, métodos ou padrões.”
- Produtividade das pessoas da área de software não acompanha demanda dos serviços
  - Os projetos de desenvolvimento de software normalmente são efetuados apenas com um vago indício das exigências do cliente”

Crise de Software

- Qualidade de software ás vezes é menos que adequada
  - Só recentemente começam a surgir conceitos quantitativos sólidos de garantia de qualidade de software
- Software existente é muito difícil de manter
  - Tarefa de manutenção nem sempre é enfatizada como um critério importante

Causas da Crise de Software

- Mitos do software – da gerencia
  - Mito 1: Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso não oferece tudo o que os desenvolvedores precisam saber?
  - Realidade: O manual é usado? Ele reflete a pratica moderna de desenvolvimento?
  - Mito 2: O pessoal tem ferramentas de desenvolvimento de software e ultima geração, afinal compramos os mais novos computadores
  - Realidade: É preciso muito mais do que os recentes computadores para se desenvolver com qualidade

Causas da Crise de Software

- Mitos do software
  - Mito 3: Se estamos atrasados nos prazos, podemos adicionar mais programadores e recuperar o tempo
  - Realidade: Desenvolvimento de software não é um processo igual à manufatura. Acrescentar pessoas em um projeto torna-lo ainda mais atrasado. Pode-se acrescentar pessoas, mas com planejamento.
  - Mito 4: Se decidir terceirizar vou poder relaxar e deixar que a firma o elabore.
  - Realidade: - Precisa saber gerir o processo internamente para coordenar o externo.

Causas da Crise do Software

- Mitos do Software – do Cliente
  - Mito 5 – Uma declaração geral dos objetivos é suficiente para se começar a escrever programas, pode-se preencher os detalhes mais tarde
  - Realidade – Uma definição inicial ruim é a principal causa de fracassos. É fundamental uma descrição formal e detalhada do domínio da informação
  - Mito 6: Os requisitos de software modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, por que o software é flexível.
  - Realidade: Uma mudança tardia em um projeto é muito mais dispendiosa do que feita no inicio do projeto.

Causas da Crise de Software

- Mitos do Software – do profissional
  - Mitos 7: Assim que escrevermos o programa e o colocarmos em funcionamento, nosso trabalho está completo.
  - Realidade: Os dados da industria indicam que os esforços feitos depois da implantação do sistema entre 50% e 70% do total
  - Mito 8: Enquanto não estiver o programa funcionando, não teremos como avaliar sua qualidade.
  - Realidade: Uma das formas mais efetivas para se avaliar a qualidade pode ser aplicada desde o inicio do projeto.

Causa da Crise de Software

- **Mitos do Software – do profissional**
  - **Mito 9: A única coisa a ser entregue em um projeto é o programa funcionando.**
  - **Realidade: O programa é somente uma parte de uma configuração de software que inclui todos os itens de informação produzidos durante a construção e manutenção do software.**
  - **Mito 10: A Engenharia de Software vai nos fazer criar documentação volumosa e desnecessária que certamente nos atrasará.**
  - **Realidade – As vezes não se relaciona a criação de documentos, mas a criação de qualidade. Melhor qualidade reduz trabalho. Menor trabalho, tem prazo de entrega mais curto**.

Usuário x Desenvolvedores x Executivos

- **O que os usuários desejam**
  - Que seu problema seja resolvido
  - Que seu problema seja resolvido rápido
  - Que a informação seja correta e esteja a disposição
  - Que o sistema não tenha erros
  - Que o sistema não se transforme em um monstro
- **Executivos**
  - Que o orçamento e os prazos não estourem
  - Querem mais pelo menor preço
  - Gerenciamento completo do processo
- **O que os desenvolvedores desejam**
  - Metodologia simples e Automatizadas
  - Linguagem visuais
  - Mais distância do meio físico
  - Que o problema do usuário não muda

8. Desafios

- Sistemas legados
- Pode fazer a manutenção e atualização desses software antigos e mal escritos
- Heterogeneidade
- Sistemas distribuídos em redes com diferentes tipos de computadores e sistemas de apoio.
- Tempo de fornecimento
- Empresas exigem respostas rápidas e se modificam rapidamente. O software de apoio deve se modificar com a mesma rapidez

9. Elementos da ES

Disciplina que reúne metodologias, métodos, técnicas e ferramentas a ser utilizados desde a percepção do problema até o momento em que o sistema desenvolvido deixa de ser operacional, visando resolver problemas inerentes ao processo de desenvolvimento e ao produto de software.

Elementos da ES

10. Qualidade - Atributos

- Refletem comportamento, estrutura, organização do programa e da documentação
  - Facilidade de manutenção
  - Nível de confiança
  - Eficiência
  - Facilidade de uso
- Atributos diferem
  - Software bancário – confiança
  - Jogo - velocidade

Qualidade

- Não é sinônimo de perfeição
- Não é absoluta
  - Muda em relação ao tempo
  - Muda conforme as pessoas
- Busca da qualidade
  - Melhorar processos- produto é conseqüência
  - Melhorar forma de encontrar e corrigir erros
  - Identificar causa dos erros

Qualidade

- Internas
  - Desenvolvedores
  - Sustentam as externas
- Externa
  - Usuário
- Processo
  - Refletem no produto
- Produto

Qualidade – ISO 9126

- Funcionalidade- Satisfaz as necessidades?
  - Adequação- Faz o que deveria fazer?
  - Acúracia- faz o que deveria fazer corretamente?
  - Interoperabilidade- Interage com outros sistemas?
  - Conformidade- está de acordo com normas, leis?
  - Segurança de acesso- evita acesso não autorizado?

Qualidade – ISO 9126

- Confiabilidade- é imune a falhas?
  - Maturidade- com que freqüência apresenta falhas?
  - Tolerância falhas – com o sw reage as falhas/
  - Recuperabilidade- Recupera os dados em caso de falhas?

Qualidades – ISO 9126

- Usabilidade – é fácil de usar?
  - Inteligibilidade – é fácil entender o conceito e a aplicação?
  - Apreensibilidade - é fácil aprendera usar?
  - Operacionalidade – é fácil operar e controlar?

Qualidades – ISO 9126

- Eficiência- é rápido e enxuto?
  - Tempo- qual o tempo de resposta e velocidade de execução?
  - Recursos – Quantos recursos usa? Por quanto tempo?

Qualidades - ISO 9126

- Manutenibilidade – é fácil modificar?
  - Analisabilidade - é fácil encontrar as falhas?
  - Modificabilidade - é fácil modificar e adaptar?
  - Estabilidade – há riscos ao se fazer alterações?
  - Testabilidade – é fácil testar as mudanças feitas?

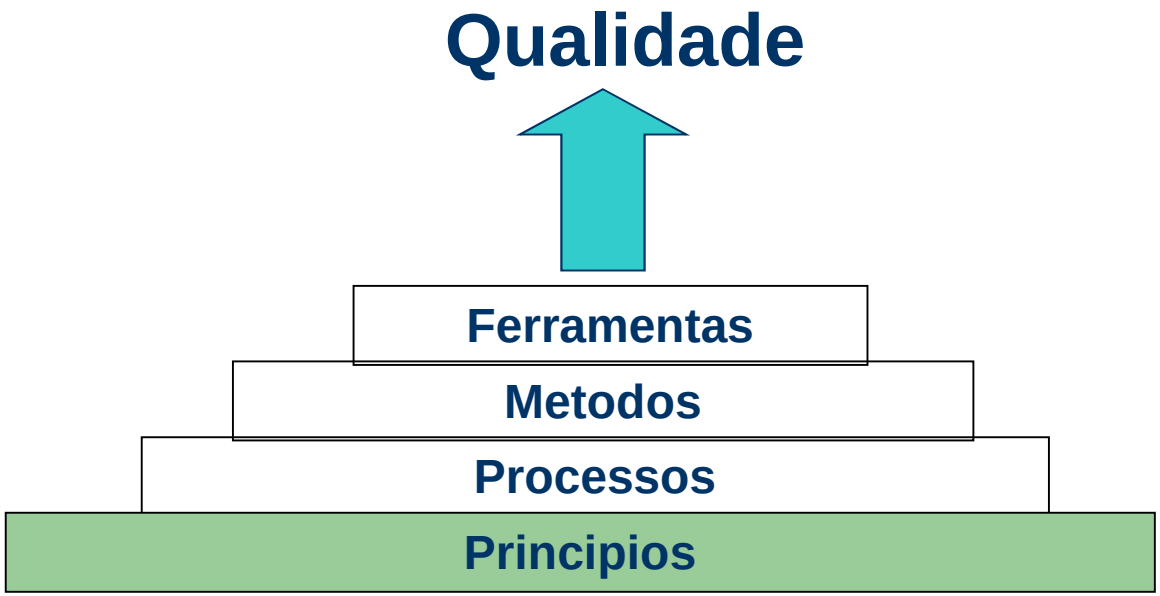
Qualidades – ISO 9126

- Portabilidade- é fácil usar em outro ambiente?
  - Adaptabilidade- é fácil adaptar a outros ambientes?
  - Capacidade de instalação – é fácil instalar em outros ambientes?
  - Conformidade - está de acordo com padrões de portabilidade?
  - Capacidade de substituição - é fácil substituir por outro sistema?

Qualidades

Qualidade	Interna	Externa	Processo	Produto

11. Elementos da ES



Princípios

- Fundamentam a Engenharia de Software
- Guiam os processos e métodos

Princípios Rigor e Formalismo

- Unir criatividade x formalidade
  - Não restringir criatividade
  - Melhorar criatividade através da confiança
- Cada passo do projeto guiado por metodologia
  - Tradicionalmente – usado na codificação
  - Necessidade: toda fase de desenvolvimento

Princípio Rigor e Formalismo

- Nível de Formalidade
  - Formal – embarcação para águas turbulentas- navio
  - Informal – embarcação para um riacho –jangada
  - Semi-formal

Princípio Rigor e Formalismo

- Qualidades atingidas no produto
  - Manutenibilidade, reusabilidade, portabilidade, entendabilidade, interoperabilidade
- No processo
  - Mesma documentação aproveitada em projetos similares
  - Auxilia na manutenção do próprio sistema
  - Facilita o monitoramento- timiliness melhorando produtividade

Princípio da Abstração

- Identificar aspectos importantes e ignorar detalhes
- Visões diferentes da realidade
- Exemplo: telefone sem fio
  - Usuário – manual com descrição dos efeitos de apertar vários botões e entrar nas fncionalidades
  - Alguém que conserta – informações sobre a caixa que deve ser aberta para substituir a pilha.

Princípio da Abstração

- Exemplo – Linguagens de programação – abstraem detalhes como números de bits, mecanismo de endereçamento de memória

Princípio de Decomposição

- Subdividir processo em atividades especificas/ separação de preocupações
- Resolver pequenas preocupações
  - Se estiver interligada a outra, tratar apenas aspecto relacionado
- Exemplos real
  - Tempo- médico separa atividades de cirurgia para manhã e consultório para tarde
- Decomposição de decisões em relação ao produto: funcionalidades a resolver, confiabilidade esperada, eficiência, portabilidade

Princípio da Decomposição

- Decomposição de relação ao processo: ambiente de desenvolvimento, estrutura da equipe, planejamento, estratégia de projeto.
- Decomposição de decisões em relação aos critérios de qualidade: tratar a corretude e depois a eficiência.
- Decomposição do produto
  - Modularização

Modularização

- Dividir sistemas em módulos
  - Tratar características de cada módulo
  - Integrar os módulos no todo
- Tipos
  - Top-down
    - Decompor o problema original em problemas menores
    - “Dividir para conquistar”
    - Análise estruturada
  - Botton-up
    - Criar um software através da composição de partes menores
    - Ideal – reutilização de código
    - Análise orientada a objetos
    - Industria Automobilística

Modularização

- Controlar coesão e acoplamento entre módulos
  - Coesão – elementos no módulo fortemente relacionados
    - Declarações, procedimentos e dados
  - Acoplamento – relacionamento entre o módulos
    - Muita dependência dificulta a análise, entendimento, testes e reutilização
  - Ideal
    - Alta coesão
    - Baixo acoplamento

Princípio da Generalização

- Focar no problema mais geral no que focado
- Solução do problema pode ser reutilizado
- Mais custosa pelo tempo de desenvolvimento
- Exemplo
  - Necessidade específica – catalogar livros de uma biblioteca e fazer pesquisas por palavra-chave
  - Generalizando – acrescentar funcionalidades como empréstimos, devoluções, aquisições

Princípio da Generalização

- Situações
  - Já existe componente para a solução generalizada
  - Será utilizado no futuro

Princípios da Flexibilidade

- Possibilidade de modificar o produto com facilidade
  - Adaptação a novos ambientes
  - Evolução de funções

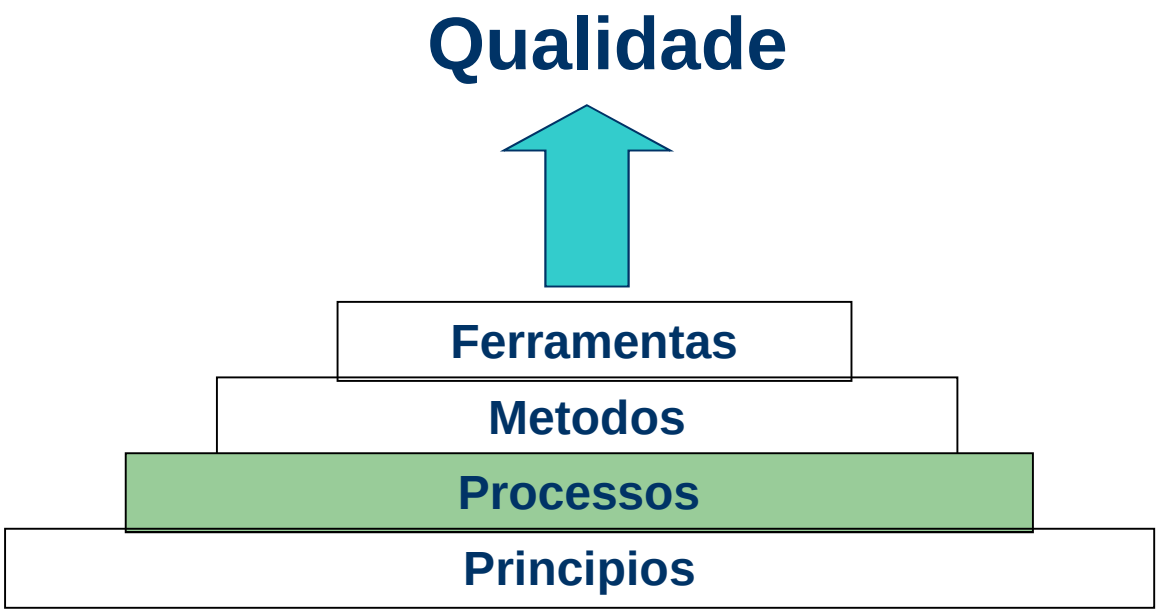
Princípio da Antecipação de mudanças

- Software modificável
  - Correção
  - Adaptação
  - Evolução funcional
- Identificar as prováveis mudanças e projetar de acordo
- Qualidades envolvidas
  - Evolutanabilidade, reusabilidade
- Envolve
  - Controle de versões, e revisões
- Processo
  - Mudança de pessoas participantes do processo

Princípio da Incrementalidade

- Utilizam passos que evoluem ao longo do tempo
  - Cada passo é um incremento do passo anterior
- Versões Múltiplas
  - Identificar subconjuntos de funções mais úteis, desenvolver e liberar
  - Obter feedback do cliente
  - Evoluir
- Garante maior compreensão dos requisitos
- Estágios intermediários são protótipos
- Exige
  - Gerenciamento de programas e documentação – muitas versões

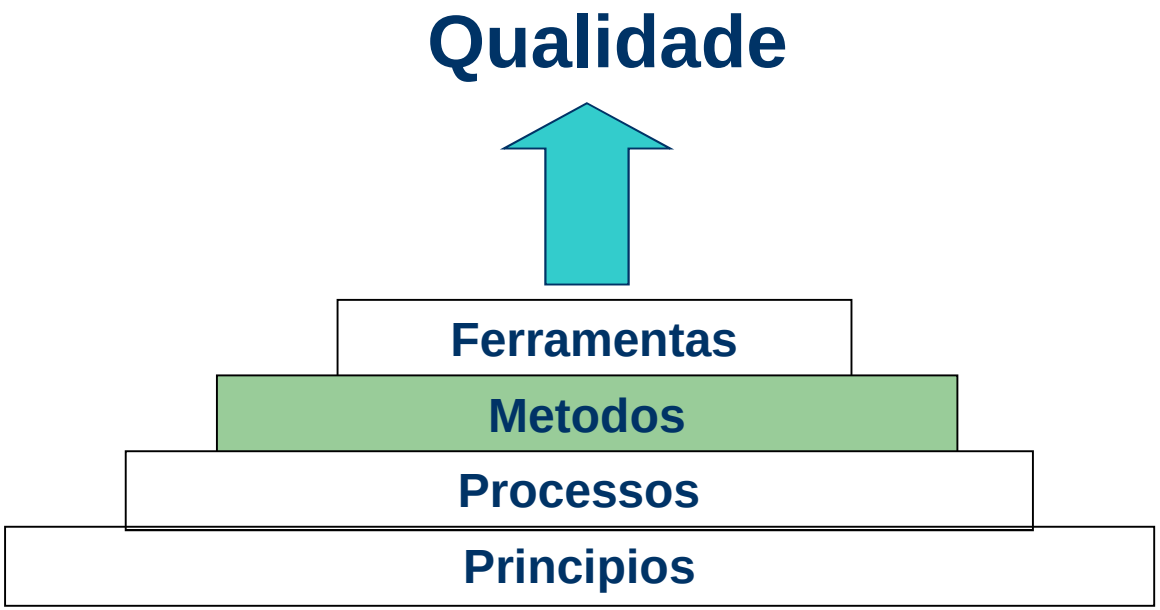
12.Elementos da ES



Processos

- Define uma estrutura para um conjunto de areas chaves de processo.
- As areas chaves do processo formam a base de controle gerencial de projetos de software

12.Elementos da ES



Métodos

- Linhas gerais que regulam uma série de operações para construir um software
  - Método de análise estruturada
  - Método orientado a objetos

Composição dos Métodos

- Descrição dos modelos e da notação
  - Fluxo de estado
  - Máquina de estado
- Regras
  - Restrições do modelo
- Recomendações
  - heurísticas da boa prática do método
- Diretrizes do processo
  - Descrição das atividades que podem ser seguidas

13. Elementos da ES

Ferramentas

Mecanismos automatizados de apoio a aplicação de métodos

- Ferramentas integradas
- CASE: Computer Aided Software Engineering

Questoes para discussao

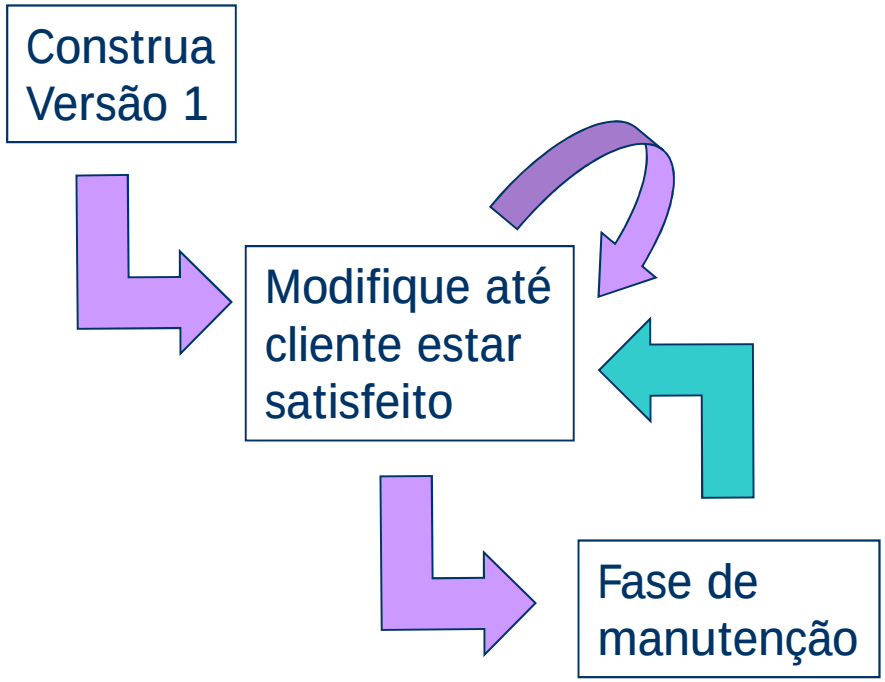
- Atualmente que tipo de ferramentas CASE sao mais utilizadas
- Como esta o mercado regional

Modelos de ciclo de vida

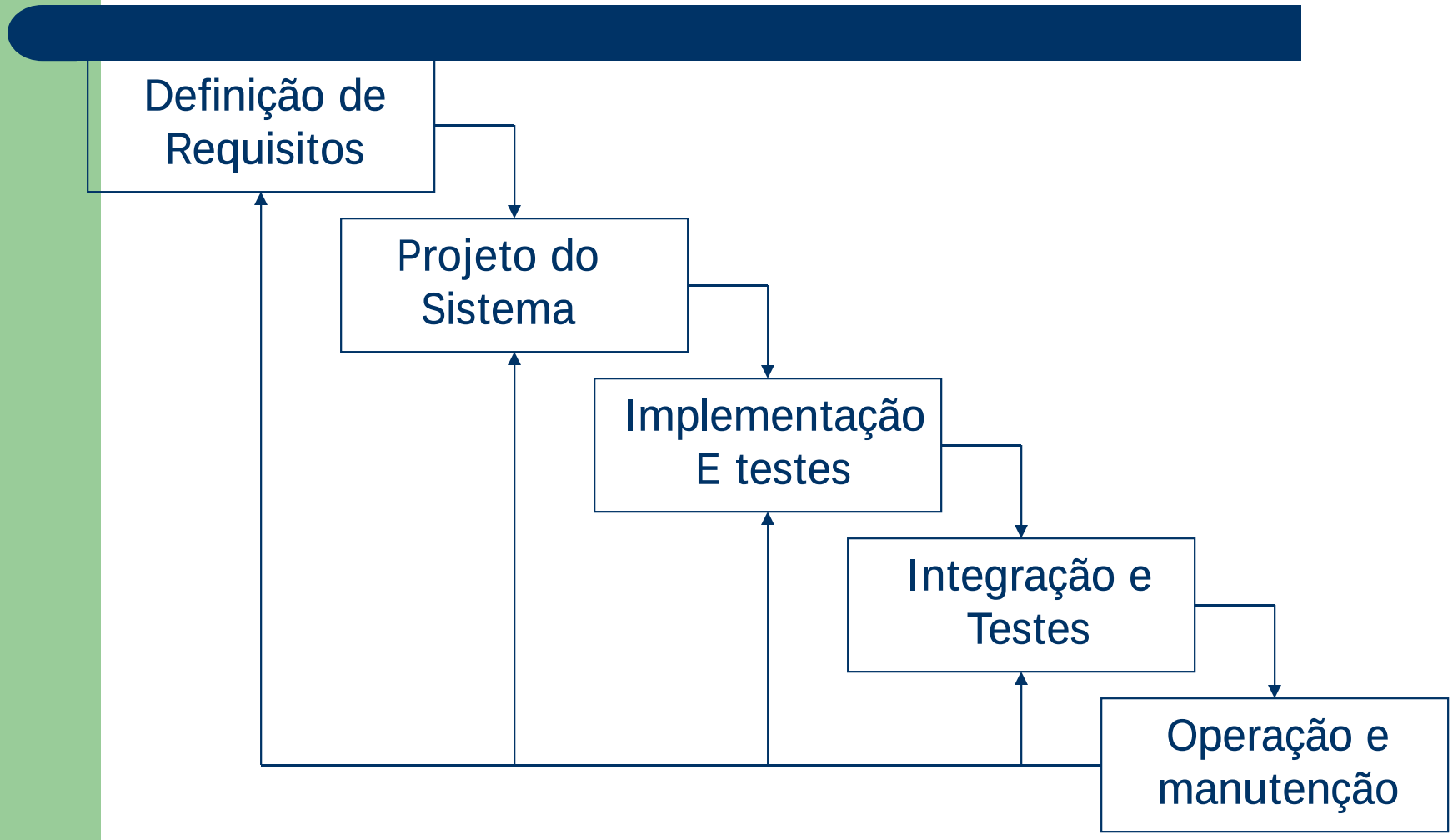
- Construa e Conserte (Code-and-Fix, Build-and-Fix)
- Cascata (“Waterfall”)
  - Cascata Modificado
- Prototipação
- Espiral
- Formal
- Baseado em reuso

Construa e Conserte

- Desvantagens
  - Sem especificação
  - Sem projeto
  - Não gerencia riscos
- Vantagens
  - Baixa sobrecarga de desenvolvimento
  - Não precisa de especialização
- Para sistemas muito pequenos



Modelo Cascata



Características do Modelo Cascata

- Desvantagens
  - Partição inflexível do projeto em estágios distintos
  - Dificuldade para lidar com as mudanças nos requisitos do sistema
  - Não gerencia riscos
- Vantagens
  - Orientado a documentação
  - Manutenção é mais simples

Características do Modelo Cascata*

- Desvantagens
  - Milestones podem tornar-se ambíguos
  - Atividades em paralelo podem levar a problema de comunicação, suposições errôneas e ineficiência
- Vantagens
  - Atividades podem ser sobrepostas
  - Mais fácil de lidar com mudanças nos requisitos
  - Capacidade para gerenciar riscos

Prototipação de sistema

- Prototipação é o desenvolvimento rápido de um sistema.
- No passado, protótipo tinha a finalidade exclusiva de avaliar os requisitos, assim o desenvolvimento tradicional era necessário.
- Atualmente , os limites entre a prototipação e o desenvolvimento normal do sistema, muitas vezes, sã indefinidos e muitos sistemas são desenvolvidos usando uma abordagem evolucionária.

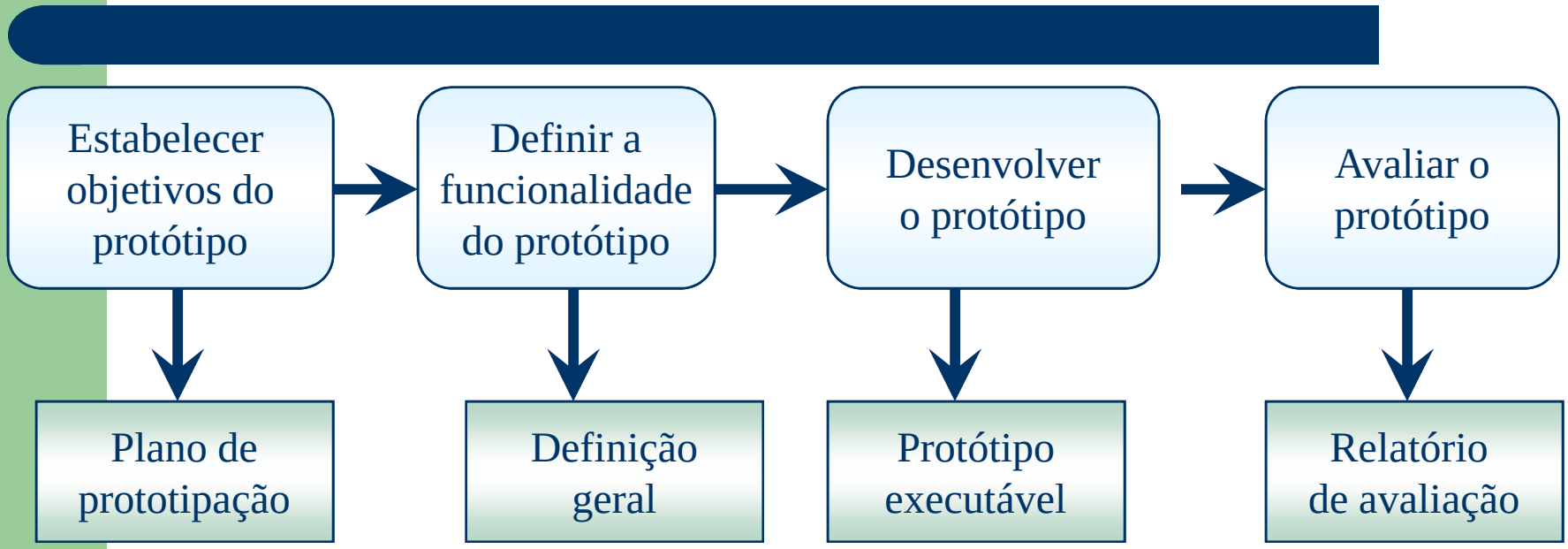
Usos de protótipos de sistemas

- O principal uso é ajudar os clientes e desenvolvedores entender os requisitos para o sistema.
  - Levantamento de requisitos. Usuários podem experimentar o protótipo para ver como o sistema pode apoiar o seu trabalho
  - Validação de requisitos. O protótipo pode revelar erros e omissões nos requisitos.
- A prototipação pode ser considerada como uma atividade de redução de riscos que reduz os riscos nos requisitos.

Benefícios da prototipação

- Equívocos entre os usuários de software e desenvolvedores são expostos.
- Serviços esquecidos podem ser detectados e serviços confusos podem ser identificados.
- Um sistema funcionando está disponível nos primeiros estágios no processo de desenvolvimento.
- O protótipo pode servir como uma base para derivar uma especificação do sistema com qualidade de produção.
- O protótipo pode ser usado para treinamento do usuário e teste de sistema.

Processo de desenvolvimento de protótipo



Benefícios da prototipação

- Melhoria na facilidade de uso do sistema;
- Maior aproximação do sistema com as necessidades dos usuários;
- Melhoria da qualidade do projeto;
- Melhoria na facilidade de manutenção, e
- Redução no esforço de desenvolvimento

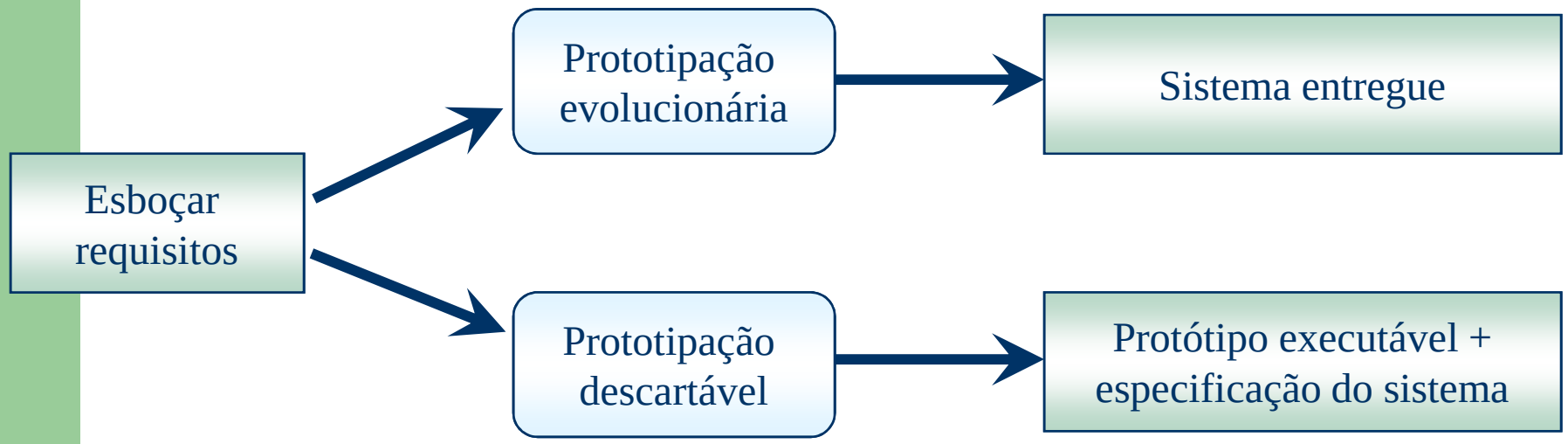
Prototipação no processo de software

- Prototipação evolucionária
  - Uma abordagem para o desenvolvimento do sistema onde um protótipo inicial é produzido e refinado através de vários estágios até atingir o sistema final.
- Prototipação descartável
  - Um protótipo o qual é usualmente uma implementação prática do sistema é produzida para ajudar a levantar os problemas com os requisitos e depois descartado. O sistema é então desenvolvido usando algum outro processo de desenvolvimento.

Objetivos da prototipação

- O objetivo da *prototipação evolucionária* é fornecer aos usuários finais um sistema funcinando. O desenvolvimento começa com aqueles requisitos que são melhores compreendidos.
- O objetivo da *prototipação descartável* é validar ou derivar os requisitos do sistema. O processo de prototipação começa com aqueles requisitos que não são bem compreendidos.

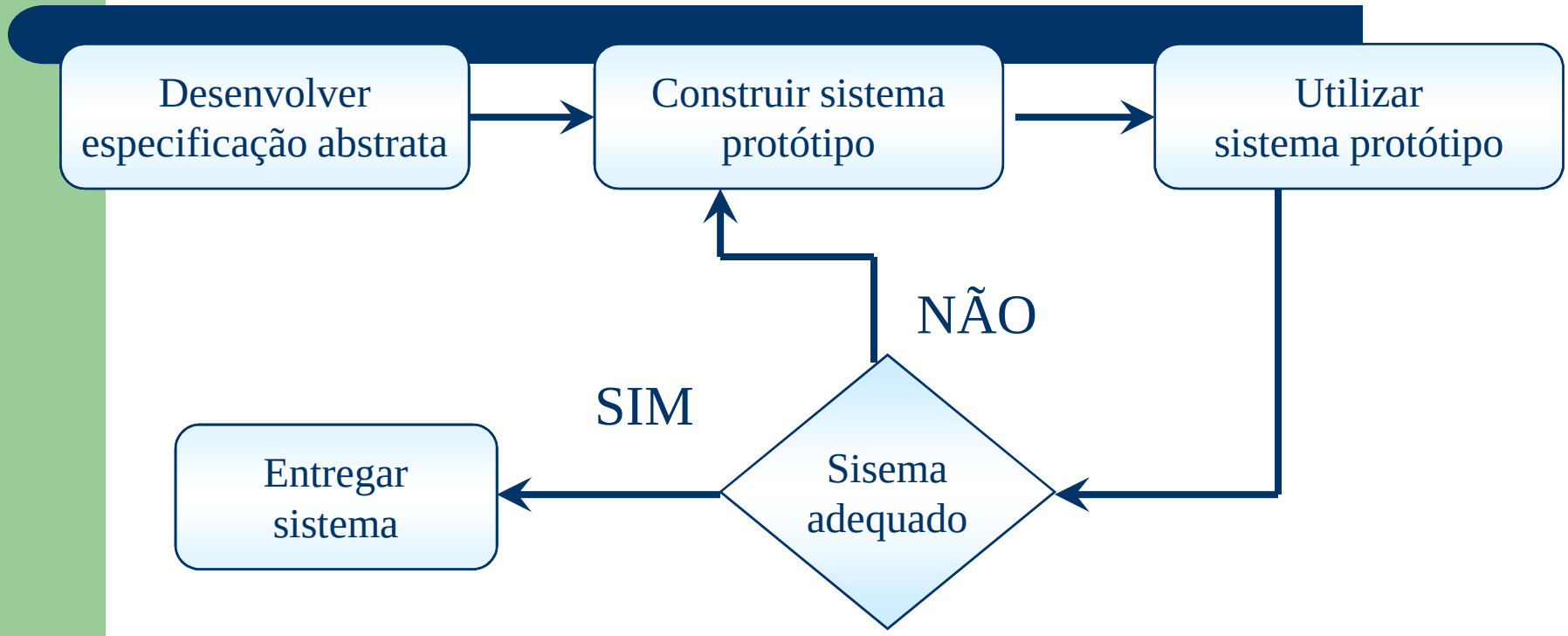
Abordagens de prototipação



Prototipação evolucionária

- Deve ser usada para sistemas onde a especificação não pode ser desenvolvida à priori, como por exemplo, os sistema de IA e os sistemas de interface com o usuário
- Baseada em técnicas que permitem interações rápidas para o desenvolvimento de aplicações.
- Verificação é impossível uma vez que não exise especificação. A validação significa demonstrar a adequação do sistema.

Prototipação evolucionária



Vantagens da prototipação evolucionária

- Rápido fornecimento do sistema
  - Em alguns casos, o rápido fornecimento e a facilidade de uso são mais importantes do que os detalhes de funcionalidade ou a facilidade de manutenção de software a longo prazo.
- Compromisso do usuário com o sistema
  - O envolvimento do usuário com o sistema significa maior possibilidade de atender aos seus requisitos e um maior empenho para que o sistema funcione de acordo.

Prototipação Evolucionária

- [redacted]ão são intercalados.
- O sistema é desenvolvido em uma série de estágios que são entregues ao cliente.
- Técnicas para o desenvolvimento rápido de sistemas, tais como ferramentas CASE e linguagens de 4a. Geração, são utilizadas.
- As interfaces com o usuário do sistema são usualmente desenvolvidas utilizando-se um sistema de desenvolvimento interativo

Problemas com prototipação evolucionária

- Problemas de gerenciamento
  - Processos de gerenciamento existentes assumem o modelo de desenvolvimento cascata.
  - Habilidades especialistas são necessárias e podem não estar disponível na equipe de desenvolvimento
  - Specialist skills are required which may not be available in all development teams
- Problemas de manutenção
  - A continuidade de mudanças tende a corromper a estrutura do protótipo do sistema, assim a manutenção a longo prazo pode ser cara.
- Problemas contratuais
  - Os contratos são, geralmente, estabelecidos baseados em uma especificação completa do software.

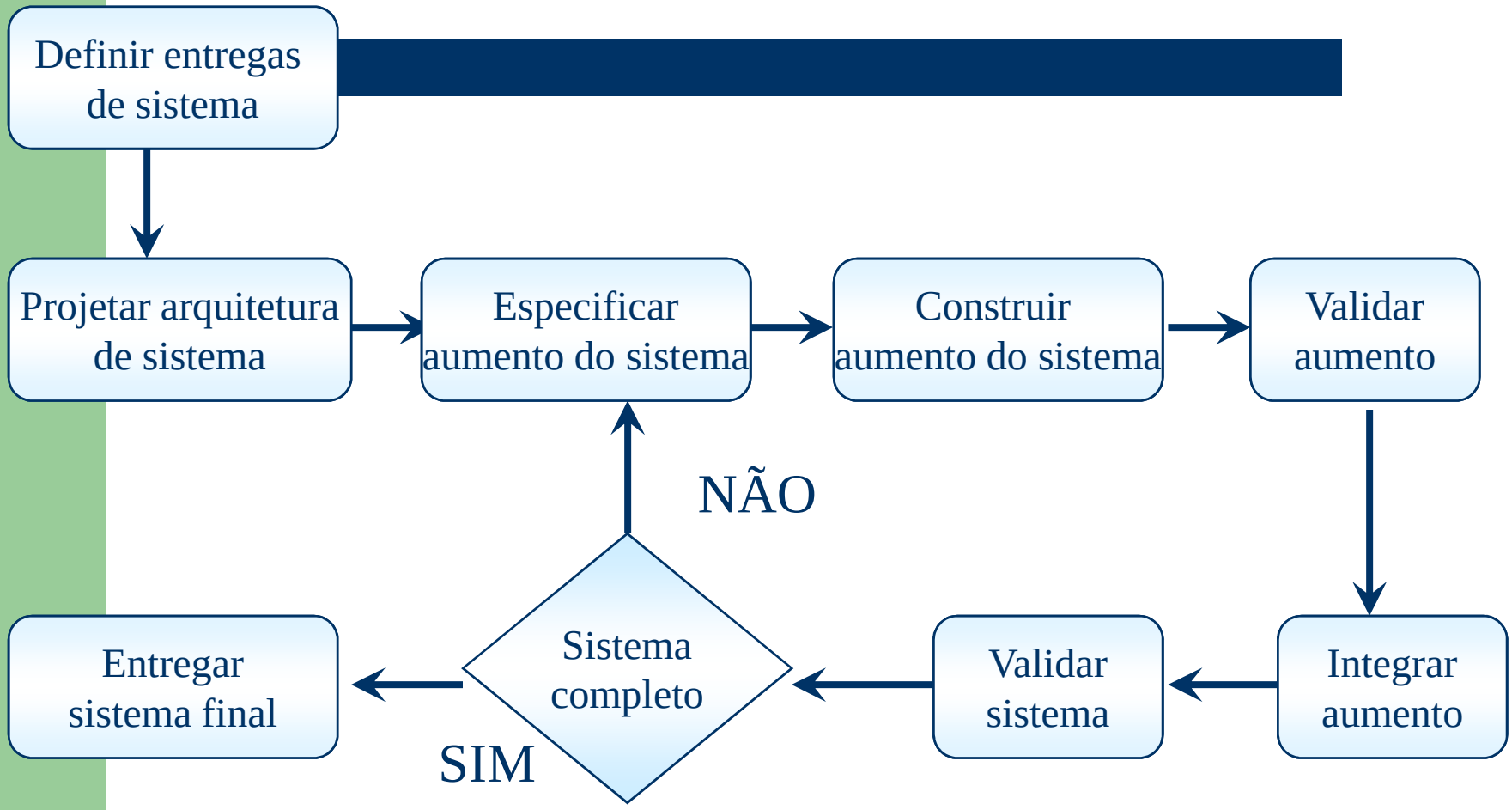
Protótipos como especificações

- Algumas partes dos requisitos (por ex. funções críticas com relação à segurança) são difíceis de aparecerem em protótipos, assim acabam não aparecendo na especificação.
- Uma implementação não tem valor legal de contrato.
- Requisitos não funcionais não podem ser testados adequadamente em um protótipo do sistema.

Desenvolvimento incremental

- O sistema é desenvolvido e liberado em incrementos após estabelecer uma arquitetura global.
- Requisitos e especificações para cada incremento podem ser desenvolvidos.
- Usuários podem avaliar os incrementos liberados enquanto outros estão sendo desenvolvidos. Portanto, esse serve como uma forma de sistema protótipo.
- O desenvolvimento incremental combina as vantagens da prototipação evolucionária com um processo de desenvolvimento mais fácil de ser gerenciado e uma melhor estruturação do sistema.

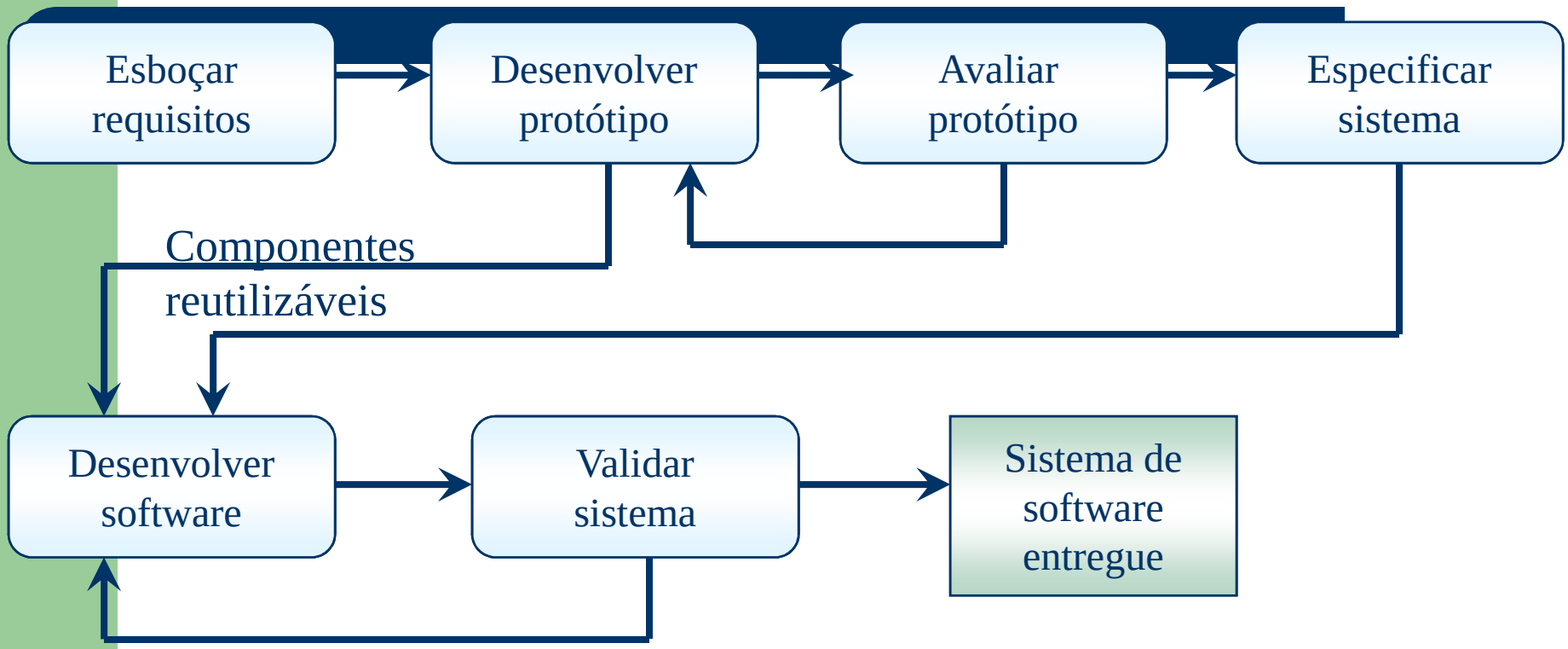
Um processo de desenvolvimento incremental



Prototipação descartável

- Usada para reduzir os riscos com os requisitos.
- O protótipo é desenvolvido de uma especificação inicial, entregue para avaliação e então descartado.
- O protótipo descartável NÃO deve ser considerado como um sistema final.
  - Características importantes podem ter sido excluídas do protótipo.
  - Não existe especificação para manutenção futura
  - O sistema será mal estruturado e difícil de manter.

Processo de software com prototipação descartável



Protótipos descartáveis liberáveis

- Desenvolvedores podem ser pressionados a entregar um protótipo descartável como um produto final
- Isso não é recomendado
  - Pode ser impossível ajustar o protótipo para atender os requisitos não funcionais.
  - O protótipo é inevitavelmente não documentado e isso é ruim para a manutenção a longo prazo.
  - A s mudanças feitas durante o desenvolvimento do protótipo provavelmente terão degradado a estrutura do sistema.
  - Os padrões de qualidade organizacional são, normalmente, deixados de lado no desenvolvimento do protótipo.

Técnicas de prototipação rápida

- Várias técnicas podem ser usadas para o desenvolvimento de protótipos
  - Desenvolvimento com linguagem dinâmica de alto nível
  - Programação de banco de dados
  - Montagem de componentes e aplicações
- Essas técnicas não são exclusivas - são muitas vezes utilizadas em conjunto.
- Programação visual é uma parte inerente da maioria dos sistemas de desenvolvimento de protótipos.

Escolha da linguagem de prototipação

- Qual é o domínio de aplicação do problema?
- Que tipo de interação com o usuário é necessário?
- Qual ambiente de suporte vem com a linguagem?
- Diferentes partes do sistema podem ser programados em diferentes linguagens. Contudo, pode haver problemas com a comunicação entre as linguagens.

Pontos-chave

- Um protótipo pode ser usado para dar aos usuários finais uma impressão concreta das capacidades desse sistema.
- A prototipação está se tornando cada vez mais comum para o desenvolvimento de sistema onde o desenvolvimento rápido é essencial.
- Protótipos descartáveis são usados para a compreensão dos requisitos do sistema.
- Na prototipação evolucionária, o sistema é desenvolvido pela evolução de uma versão inicial em uma versão final do sistema.

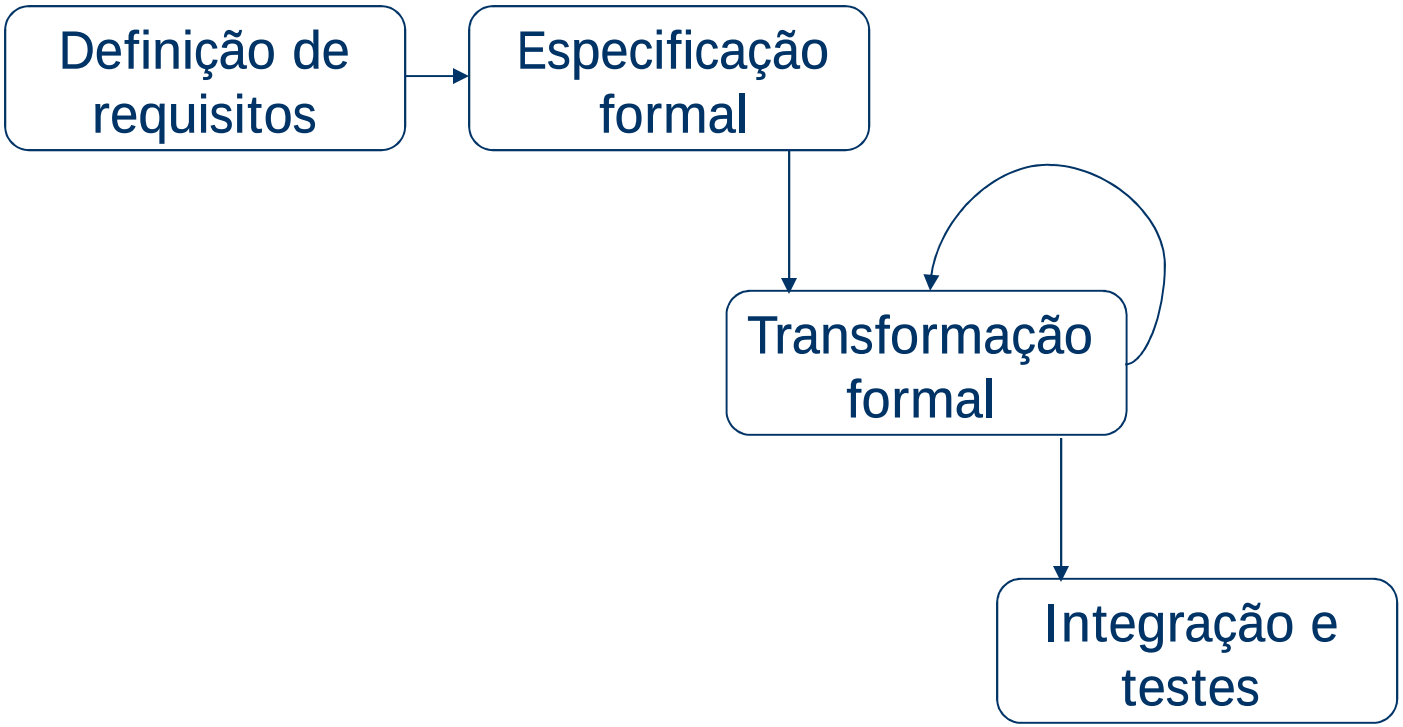
Pontos-chave

- O desenvolvimento rápido é importante na prototipação de sistemas. Isso pode levar à exclusão de algumas funcionalidades do sistema ou na diminuição dos requisitos não funcionais.
- Entre as técnicas de prototipação estão o uso de linguagens de nível muito elevado, a programação de bando de dados e a construção de protótipos a partir de componentes reutilizáveis.
- A prototipação é essencial para o desenvolvimento de interfaces com o usuário, as quais são difíceis de serem especificadas usando um modelo estático. Os usuários deveriam estar envolvidos na avaliação e na evolução do protótipo.

Processo Formal

- Transformação de uma especificação formal em um programa executável
- Pode ser empregado de duas formas:
  - Através de um cálculo de refinamentos
    - Implementação é derivada por construção.
  - Através de passos de refinamento
    - Versão mais concreta do sistema é proposta e depois deve-se verificá-la em relação a anterior

Processo Formal



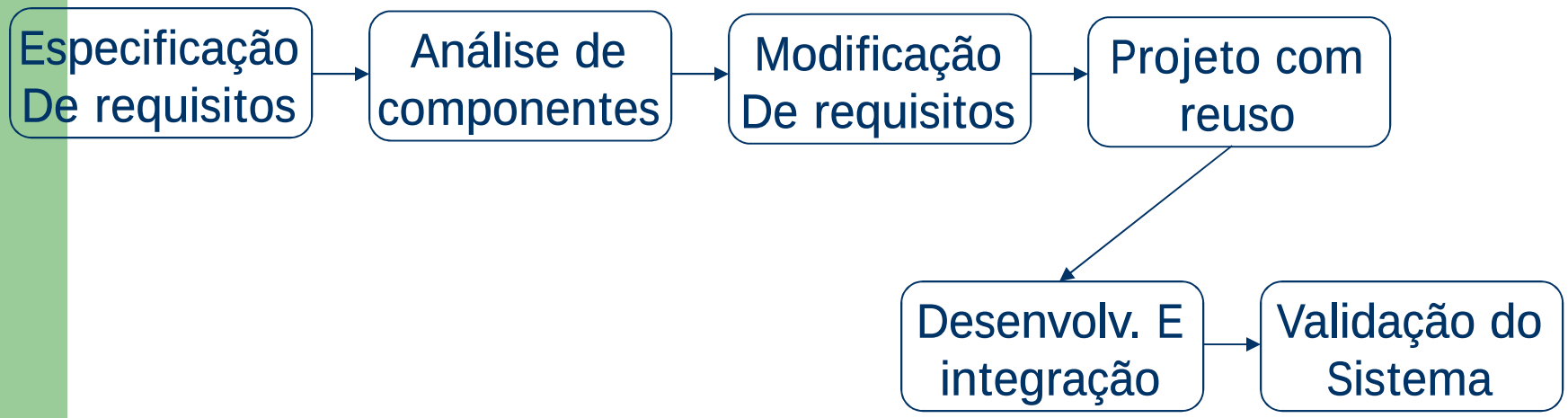
Perspectivas

- Desvantagens
  - Necessita de profissionais altamente qualificados para aplicar a técnica
  - Alguns aspectos ainda são difíceis de especificar (GUI)
- Vantagens
  - Garantia de segurança e confiabilidade
- Aplicações
  - Sistemas críticos e complexos

Processo baseado em Reuso

- componentes existentes na própria empresa ou no mercado
- Estágios do processo
  - Análise dos componentes
  - Modificação dos requisitos
  - Projeto do sistema com reuso
  - Desenvolvimento e integração
  - Validação
- Essa abordagem está se tornando cada vez mais importante, mas ainda faltam casos experimentais bem-sucedidos

Processo baseado em Reuso



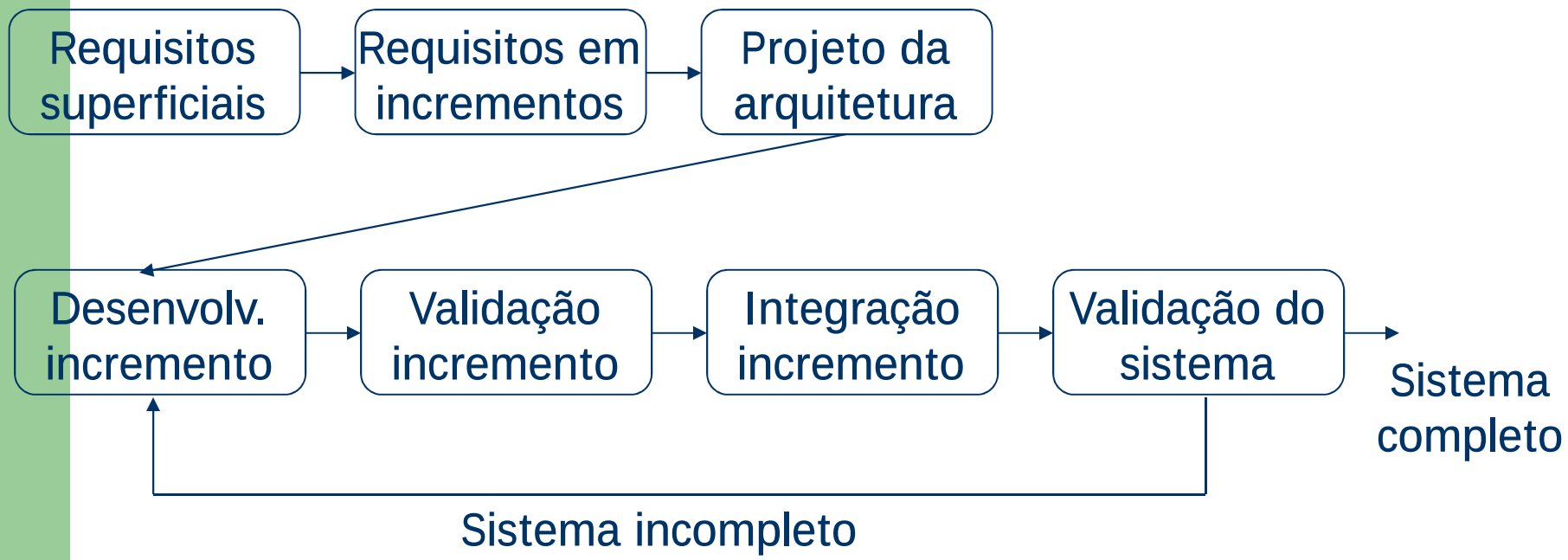
Processo Iterativo

- Os requisitos do sistema SEMPRE mudam durante o desenvolvimento
- Iteração pode ser aplicado a qualquer um dos processos de desenvolvimento
- Duas abordagens são destacadas:
  - Desenvolvimento incremental
  - Desenvolvimento em espiral

Desenvolvimento Incremental

- Ao invés de entregar o sistema completo, divide-se o sistema em partes de tal forma a cada entrega corresponder a alguma funcionalidade do sistema
- Requisitos do usuário são priorizados e os quanto maior a prioridade mais cedo tal funcionalidade deve ser entregue
- Uma vez que o desenvolvimento de um incremento seja iniciado, esses requisitos são fixados enquanto que os posteriores são deixados serem modificados

Desenvolvimento Incremental



Vantagens

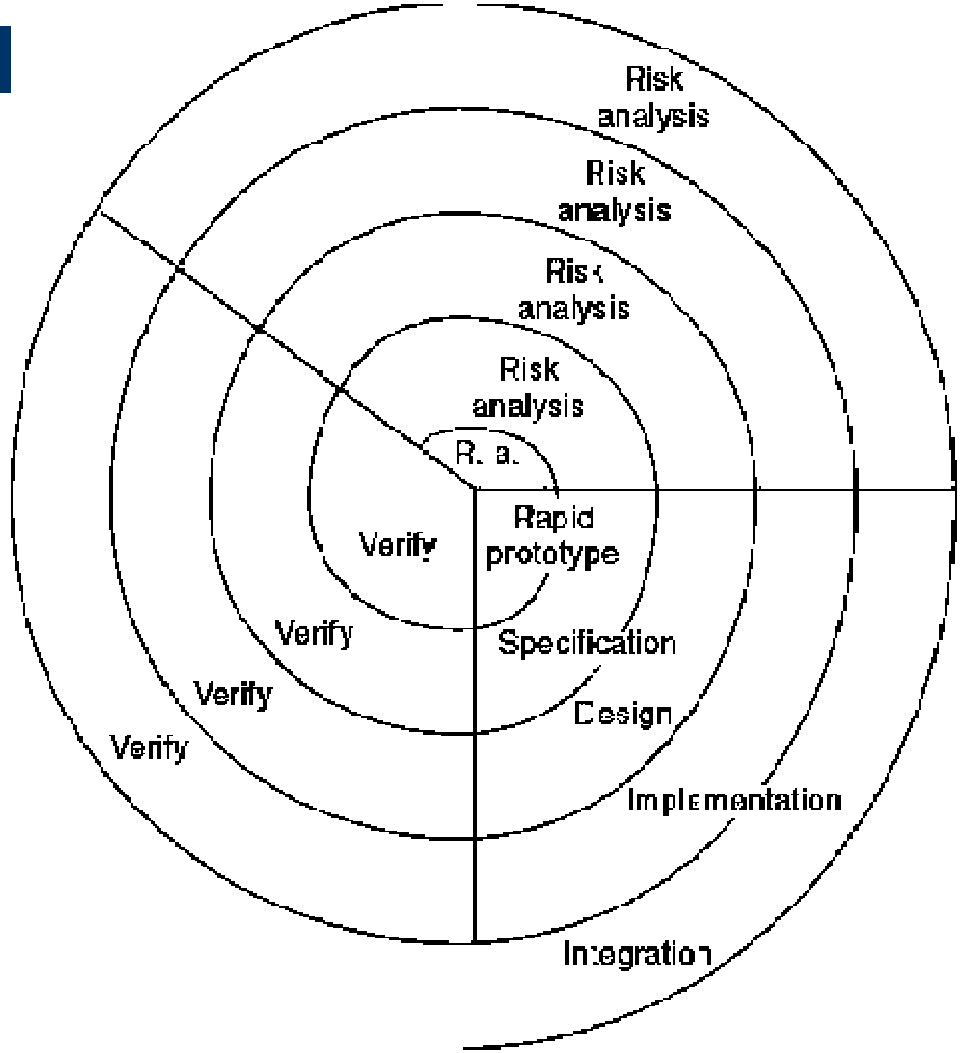
- Solicitações dos clientes são entregues a cada incremento. Assim, as funcionalidades são entregues o mais cedo possível
- Incrementos iniciais servem de protótipo para obter requisitos para incrementos posteriores
- Diminuição do risco de falha de todo o projeto
- Serviços de maior prioridade tendem a receber maior ênfase em testes

Modelo Espiral

- Forma Simplificada
  - Modelo cascata mais análise de riscos
- Precede cada fase por
  - Alternativas
  - Análise de riscos
- Procede cada fase por
  - Avaliação
  - Planejamento da fase seguinte

Modelo Espiral Simplificado

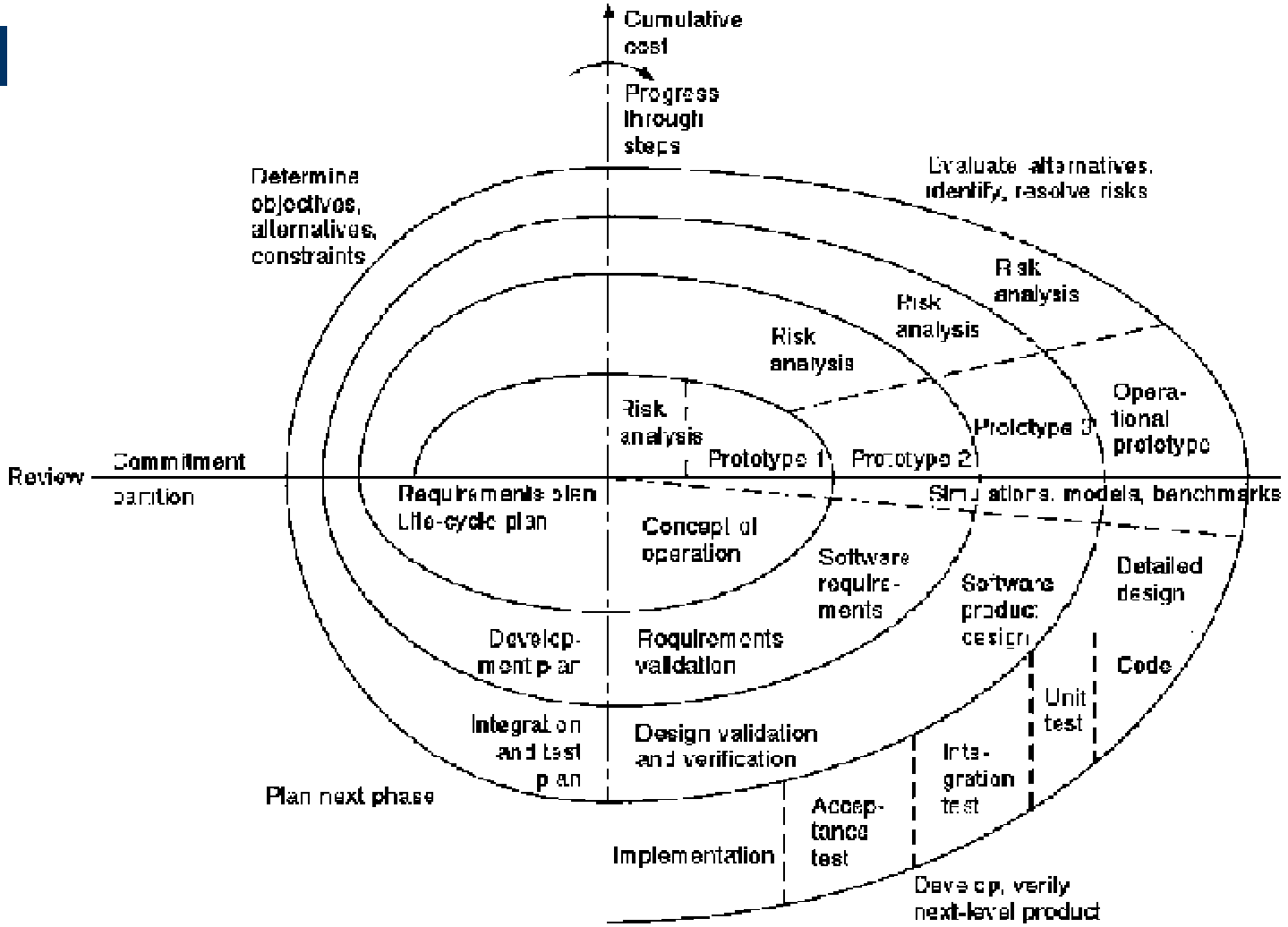
- Se os riscos não podem ser resolvidos, projeto é terminado imediatamente



Modelo Espiral Completo

- Dimensão Radial
  - Custo acumulado atualizado
- Dimensão Angular
  - Progresso através da espiral

Modelo Espiral Completo



Características do Modelo Espiral

- Desvantagens
  - Bem aplicado somente a sistemas de larga escala
  - Sistemas devem ser produtos internos da empresa
- Vantagens
  - Fácil de decidir o quanto testar
  - Não faz distinção entre desenvolvimento e manutenção

Gerência de Projeto de Software

Gerência de Projetos

Envolve planejamento, acompanhamento e controle de pessoas, processos e eventos que ocorrem á medida que o software evolui de um conceito inicial a uma implementação operacional

Gerência

- Todas as atividades e tarefas realizadas por uma pessoa ou mais pessoas com o objetivo de planejar e controlar as atividades de outros para atingir um objetivo ou complementar uma atividade que não pode ser atingido ou complementadas pelos outros de forma independente

Gerência de Projetos

- Gerente de Projetos
  - Planeja, acompanha e controla o trabalho de uma equipe de Engenheiros de Software
- Gerente Senior
  - Coordena a interface entre o negócio e os profissionais de software

Foco na Gerência

- Pessoas
- Produto
- Processo
- Projeto

Gerência de Projetos

- **Pessoas** precisam ser organizadas para realizar o trabalho de forma efetiva
- É necessária comunicação com o usuário para se entender o escopo de **produto** e os requisitos
- Um **processo** adequado ao produto e ás pessoas precisa ser selecionado
- O **projeto** deve ser planejado, estimando-se esforço e cronograma, definindo-se os produtos a serem gerados e os pontos de avaliação da qualidade, e, estabelecendo-se os mecanismos de acompanhamento e controle.

Pessoas

- Recrutamento
- Seleção
- Gerência de desempenho
- Treinamento
- Compensação (MOI)
- Organização
- Desenvolvimento da equipe

Pessoas

- A equipe de um projeto deve estar organizada de forma a maximizar as características positivas de cada pessoa

- Está é uma função do gerente do projeto

O Produto

No inicio do projeto, um dilema para o gerente do projeto:

- Estimativas quantitativas em um plano de projeto organizado são necessárias, mas não se dispõe de informações sólidas
- A análise detalhada dos requisitos forneceria as informações necessárias para as estimativas, mas leva muito tempo
- Os requisitos podem não ser precisos e mudarem muito
- **O plano do Projeto deve ser feito neste momento**

Antes de planejar é necessário

- Definir os objetivos e o escopo do projeto
- Considerar as alternativas de solução
- Identificar as restrições técnicas e gerenciais

A definicao do escopo do projeto deve ser precisa, sem ambiguidades e compreensivel para a gerencia e a equipe tecnica

Posteriormente, para estimativas mais precisas

- Decomposição do problema
  - Baseada na funcionalidade que deve ser entregue
  - Baseada no progresso usado no desenvolvimento

O Processo

- “ Um processo de software fornece o estrutura a partir do qual pode ser estabelecido um plano abrangente para o desenvolvimento de software” (PRESSMAN)

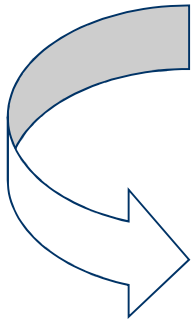
Problema:

- Selecionar o modelo de processo adequado ao software que será desenvolvido e a equipe
- O gerente do projeto deve solucionar o modelo de processo mais adequado:
  - Ao cliente que solicitou o produto e ás pessoas que farão o trabalho
  - Ás características do produto
  - Ao ambiente aonde os desenvolvedores trabalham

- Após a escolha do modelo de processo faz-se o mapeamento das atividades e tarefas do Processo Padrão da empresa para o modelo.

O Projeto

- Para gerenciar com sucesso é necessário entender o que pode dar errado
- 10 sinais de perigo



Pelo menos 7 destes 10 sinais se manifestam antes do final da fase de projeto, antes de escrever qualquer linha de código

Gerência de Projetos

1. A equipe do desenvolvimento não entende as necessidades do cliente
2. O escopo do produto não está bem definido
3. As alterações não são gerenciadas de adequadamente
4. A tecnologia escolhida muda
5. As necessidades do negócio mudam ou são mal definidas
6. O cronograma não é realista
7. Os usuários são resistentes
8. O patrocínio para o projeto é perdido ou nunca existiu
9. A equipe do projeto perde pessoas com as habilidades necessárias
10. Gerentes e desenvolvedores não utilizam boas práticas de Engenharia de Software

Função da Gerência

- Planejamento: elaboração de um curso de ação para atingir objetivos organizacionais
- Organização: organização das unidades de trabalho e responsabilidades
- Gerência de Pessoal: seleção e treinamento de pessoal
- Direcionamento: criação de um clima que motive a equipe a atingir os objetivos
- Controle: medição e avaliação de desempenho das atividades com relação aos objetivos planejados

Gerência de Projetos de Software

- Medições e Métricas
- Estimativas
- Análise de Riscos
- Elaboração de Cronogramas
- Acompanhamento e Controle

Medições e Métricas

- Métricas do Processo
  - Métricas coletadas durante o processo: para medir a eficácia das atividades de garantia da qualidade, gerência de mudança e gerência de projetos.
  - Métricas coletadas depois do fim do projeto: examinam a qualidade e a produtividade

Como os dados de diferentes projetos podem ser comparados e alisados de uma maneira consciente

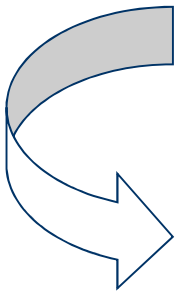
- Métricas do Produto
  - Medem características técnicas do software que são indicadores da qualidade do software

Estimativas

- Muitas vezes o cronograma e orçamento são estabelecidos por razões de negócio
- O papel das estimativas muitas vezes é verificar a viabilidade do que foi estabelecidos
- Para estimar é necessário ter-se:
  - Escopo do projeto
  - Visão de alto nível das funcionalidades
  - Avaliação da dificuldade e complexidade
  - Decomposição do problema em problema menores
  - Dados históricos
  - Modelos
    - COCOMO
    - Pontos por fusão

Aspectos que Afetam a Confiabilidade das Estimativas

- Complexidade
- Tamanho do Projeto
- Grau de incerteza estrutural



Estimativa de Riscos

- Estimativa de custos e esforços nunca será uma ciência exata
- Entretanto pode ser transformada num conjunto sistemático de passos que produzem estimativas dentro de um risco aceitável
- Opções para se ter estimativas confiáveis
  - Adiar as estimativas
  - Basear as estimativas em projetos similares já concluído
  - Usar técnicas de decomposição para gerar estimativas de custo e esforço
  - Usar um ou mais modelos para estimativas de custos

“É característica de uma mente instruída, ficar satisfeita com o grau de precisão quem a natureza do problema permite, e não buscar exatidão onde só é possível uma aproximação de verdade.”

Aristóteles

Gerência e Análise de Riscos

- Elemento chave da gerência de projetos
- Muitas coisas podem dar errado ao longo do desenvolvimento de um projeto
- É necessário estar preparado, entender os riscos e coletar dados de forma a evitar problemas ou gerenciá-los

Estratégia Reativa ou Pró - Ativa

- Estratégia reativa – Escola Indiana Jones
- Estratégia pró-ativa
  - O primeiro objetivo seria evitar riscos
  - Como nem todos os riscos podem ser evitados a equipe elabora um plano de contingência em que permitirá responder aos problemas de maneira controlada e efetiva.

Análise de Riscos

- Quando problemas acontecem os gerentes e a equipe, em geral, não estão preparados
- Necessidade de realizar Análise de Riscos
- Etapas
  - Identificação dos Riscos
  - Avaliação de Riscos
  - Hierarquização dos Riscos
  - Gerência dos Riscos

Elaboração de Cronograma

Realizada a partir de:

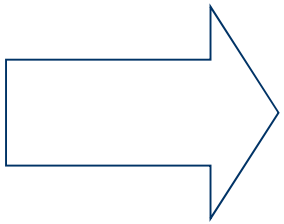
- Processo de software
- Estimativas
- Análise de riscos
- Recursos humanos disponíveis

  Identificação de:

  - Independência e paralelismo de tarefas
  - tempo

Acompanhamento e Controle

- Foco em Controle



Qualidade

Alterações

Acompanhamento e Controle são mais efetivos quando integrados ao processo de software

- Processo fornece marcos e pontos de controle que devem ser usados para acompanhamento do projeto

Planejamento de Projetos

- Planejar é decidir antecipadamente o que fazer, como fazer, quando fazer e quem deve fazer
- Todo projeto deve começar com um plano de projeto
- O plano do Projeto tem como foco os objetivos do projeto, as ações necessária para atingir estes objetivos e os potenciais riscos e problema que podem afetar se atingir estes objetivos

Planejamento de Projetos

Por que planejar?

- Evitar o fracasso
- Prever custos, recursos, prazos e riscos
- Analisar alternativas
- Organizar
- Preparar-se para alterações
- Poder acompanhar o andamento do projeto
- Planejar melhor da próxima vez

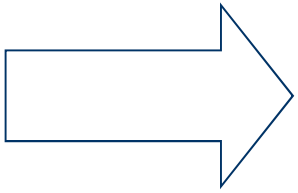
Planejamento de Projetos

Quando planejar?

- O planejamento começa de forma macroscópica no início do projeto
- O planejamento é revisto e detalhado ao longo do projeto

Planejamento de Projetos

O que planejar?



Processo de Software
Controle da qualidade
Testes
Recursos Humanos
Acompanhamento
Cronograma
Documentação
Riscos
Custo
Gerência de Configuração
Medição e Análise

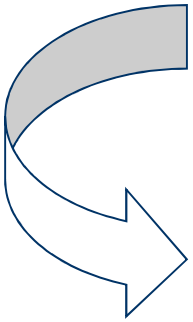
Planejamento de Projetos

Plano do Processo

- Ciclo de Vida do Projeto
- Instanciação do Processo da Organização para o Projeto

Planejamento de Projetos

Processo de software



f(x)

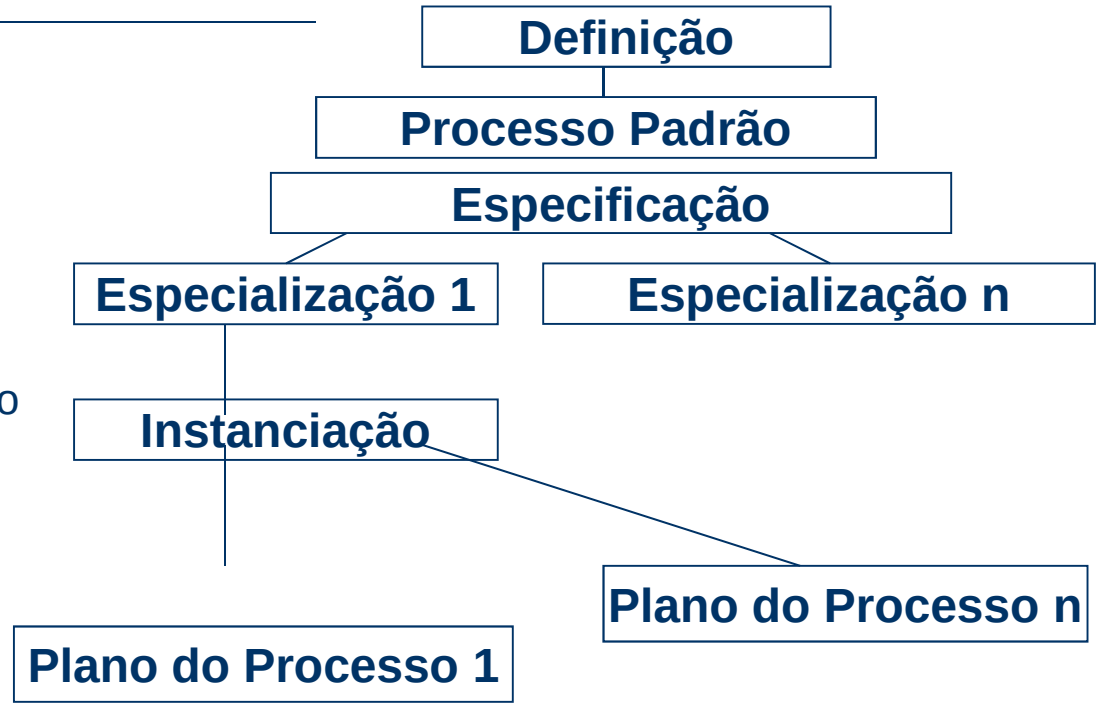
- **Tipos de sistemas**
- **Domínio de aplicação**
- **Organizações e suas equipes**
- **Restrições de negócio (cronograma, custo, qualidade**

Enfoque para Definição e Avaliação de Processo de Software

ISO 12207
CMMI
Praticas E. de Software
Cultura Organizacional

Paradigma de Desenvolvimento
Tipos de Software

Particularidades do Projeto



Planejamento de Projetos

Plano de Organização

- Equipe de Gerência
- Equipe de Desenvolvimento
- Equipe de Controle da Qualidade
- Acessorias

Planejamento de Projetos

Plano de Documentação

Deve definir

- Que documentos devem ser gerados
- Em que momento devem ser gerados
- O roteiro para elaboração do documento
- Responsabilidade
- Destinatários
- Ferramentas de apoio

Planejamento de Projetos

Plano de Acompanhamento

- Marcos
- Pontos de Controle
- Procedimentos para Acompanhamento do Projeto

Planejamento de Projetos

Plano de Controle de Qualidade

- Controle da Qualidade ao longo do Desenvolvimento
- Avaliação do Produto Final
- Plano de Testes

Planejamento de Projetos

Plano de Recursos e Produtos

- Recursos Humanos
- Recursos de Hardware
- Recursos de Software
- Recursos Financeiros
- Análise de Riscos
- Cronograma
- Produtos

Custo de um Projeto

Inclui:

- Facilidade: hardware, espaço, mobiliários, telefones, modem, ar condicionado, cabos, discos, papel, xerox, etc.
- Métodos
- Ferramentas
- Equipe
  - Maior componente dos custos
  - É necessário determinar quantos homens-dia serão necessários para completar o projeto => ESFORÇÕ

Recursos Humanos

No desenvolvimento de software o principal custo é com o pessoal

estimar custo = estimar pessoal

- à Inicialmente define-se as características desejáveis para o pessoal
- à Após estimar-se o esforço de desenvolvimento, estima-se o número de pessoas

  CMMI -> seleção de pessoal para o projeto baseado no conhecimento de habilidades

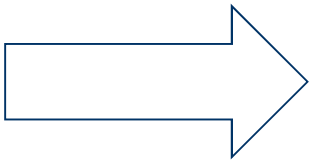
Planejamento de Projetos

Duas pessoas coma mesma função podem ser muito diferentes quanto a:

- Capacidade para realizar trabalho
- Interesse pelo trabalho
- Experiência com aplicações similares
- Experiência com linguagens e ferramentas similares
- Treinamento
- Capacidade de comunicação com outros
- Capacidade de dividir responsabilidades
- Capacidade gerencial

Esforço

- Componente do custo que envolve mais incertezas
- Afetado pelo estilo de trabalho, organização de projeto, capacidade, interesse, experiência, treinamento e outras características pessoais
- Afetado pela necessidade de comunicação, reuniões, documentos e treinamento



Técnicas para estimar Esforço

Julgamento de Especialista

- Técnicas informais baseadas na experiência do gerente com projetos similares
  - Precisão da estimativa depende da competência, experiência, objetividade e percepção de quem faz a estimativa
  - Estimativas top-down e bottom-up
  - Estimativa por analogia
  - Tendem a esquecer fatores que incidem no esforço necessário para um projeto
  - Técnica Delphi
  - Dados históricos de projetos de organização (um modelo que se mostra adequado numa organização pode não ser em outra)

Métodos Algoritmos

- Modelos que expressam a relação entre esforços e os fatores que tem influência sobre ele.
- Geralmente descritos através de equações onde o esforço é uma variável dependente e vários fatores (experiência, tamanho, tipo de aplicação e etc) são variáveis independentes
- A maior parte dos modelos considera tamanho como sendo o fator que tem maior influência -> é um problema destes modelos
- As estimativas são feitas no início do projeto antes de se ter estimativas precisas do tamanho
  - Modelos transferem o problema de estimar esforços para o de estimar tamanho

Causas de Estimativas Imprecisas

- Pedidos freqüentes de alterações
- Negligência de tarefas
- Falta de entendimento dos usuários sobre os requisitos
- Análise insuficiente ao desenvolver a estimativa
- Falta de coordenação do desenvolvimento do sistema, serviços técnicos, operações, administração de dados e outras funções durante o desenvolvimento
- Falta de um método adequado ou de diretrizes para estimar

Aspectos do Projeto que mais Influenciam as Estimativas

- Complexidade da aplicação
- Necessidade de integração com sistemas existentes
- Complexidade dos programas do sistema
- Tamanho do sistema
- Capacidade da equipe
- Experiência da equipe com a aplicação
- Potencial de faturas alterações nos requisitos
- Experiência da equipe com a linguagem de programação
- Banco de dados
- Tamanho da equipe
- Volume de padrões de documentação e de programação
- Disponibilidade de ferramentas
- Experiência da equipe com o hardware

Proposta para reflexao

- A decisao de fazer ou comprar

- O processo de tercerizacao

Gerência de Riscos

- Riscos “ possibilidade de sofrer dano ou perda; perigo’’
- Para projetos de desenvolvimento de software o dano ou perda pode er:
  - diminuição da qualidade do produto
  - aumento de custos
  - atraso o cronograma
  - Falha total do programa

Atividades da Gerência de Risco

- ¾Identificar riscos
- ¾Avaliar ricos
- ¾Classificar riscos
- ÃHierarquizar riscos
- ÄPlanejar como mitigar riscos
- ÄAcompanhar riscos e Plano
- ÄRever e ajustar Plano

Atividades da Gerência de Riscos

- Identificar riscos
  - O primeiro passo é listar os riscos e torná-los visíveis para todos
  - Nada tem mais sucesso em tornar gerentes interessados nos riscos do que uma lista de 100 ou mais riscos, analisados e hierarquizados.
  - Quando está no papel os gerentes se sentem obrigados a tomar medidas.

Atividades da Gerência de riscos

- Avaliar riscos
  - Avaliação qualitativa ou quantitativa?
  - Há casos onde se pode acessar a probabilidade de um evento futuro mais não raros
  - Em geral uma quantificação prematura não é necessária.
  - Cuidado para não gastar mais tempo avaliando o risco, do que o tempo seria gasto se este ocorrer

- Classificar riscos
  - Um único risco pode dar origem a varias declarações de risco
  - Classificar/ agrupar declarações de risco em categorias baseado-se em características comuns ajuda a encontrar riscos globais que podem ser resolvidos juntos

Atividades da Gerência de Riscos

- Hierarquizar riscos
  - Deve-se tratar primeiro os ricos mais importantes e procurar ver para quais destes se tem recursos para mitigar
  - Em geral é difícil lidar com mais de 10 erros

Atividades da Gerência de Riscos

- Planejar como mitigar riscos
  - Nem todos os riscos podem ser mitigados
  - Deve-se identificar:
    - Riscos com o quais se pode conviver caso se tornem realmente problemas
    - Riscos para os quais deve-se designar um responsável mais capaz de gerenciá-lo

Atividades da Gerência de Riscos

- Acompanhar riscos e Planos
  - Com um objetivo claro para mitigar riscos, pode-se determinar se o esforço para mitigar riscos esta sendo feito
  - Podem ser necessário relatórios e dados para acompanhar o status de riscos críticos
  - Relatórios devem ser escritos evitando-se a comunicação real

Atividades da Gerência de Riscos

- Rever e ajustar Planos
  - Controlar riscos significa:
    - Alterar a estratégia para mitigar erros se ela não se mostrar efetiva
    - Tomar medidas quando um risco se tornar importante a ponto de requerer ações para mitigá-lo
    - Seguir um planejamento de contingências
    - Encerrar um risco quando este não mais existir

Taxonomia de Riscos

- Riscos de Cronograma

  Os problemas ao se tentar desenvolver um produto difícil ou com menos pessoal do que o desejado são seriamente exacerbados quando não existe tempo suficiente

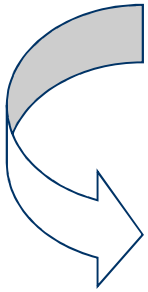
Conseqüência

- custos mais altos
- Atraso no cronograma
- Entrega de um produto não adequado

Taxonomia de Riscos

- Risco do produto

  O desenvolvimento de produtos complexos ou com muita necessidade de confiabilidade é uma tarefa difícil



- Necessidade de sólida formação experiência em análise e programação por parte da equipe para evitar retrabalho
- Necessidade de se alocar o tempo adequado o tempo pois haverão mais atividades de V&V

Taxonomia de Riscos

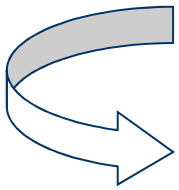
- Riscos da Plataforma
  - Problemas relacionados a sistema operacional, hardware e ambiente de desenvolvimento imaturos
  - Erros devido *a upgrades*
- Riscos de Pessoal
  - Principal fonte de riscos em projetos
- Riscos de Estimativa de Custos
  - Planejamento baseado em estimativas inadequadas

Taxonomia de Riscos

- Riscos do Processo
  - Problemas relacionados a métodos e ferramentas
    - Os riscos aumentam consideravelmente se ferramentas e métodos comprovadamente adequados não são usados
  - Problemas decorrentes de não detectar erros no inicio do projetam afetam custo e tempo das fases finais
  - Problemas de comunicação quando o projeto é desenvolvido em vários locais
  - Processo de desenvolvimento inicial

Importante:

- Guardar informações sobre os riscos de um projeto
- Documentar o que se aprendeu



Banco de Dados de Riscos

Referência Bibliográfica

Pressman, Roger S. ; Engenharia de Software, Ed. Makron Books

Sommervile, I; Engenharia de Software, , Ed. Pearson