Ciclo de vida do projeto

[2]

Metodologias de desenvolvimento de software

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Prof. Felippe Scheidt – IFPR – Campus Foz do Iguaçu
2024

Tópicos

- Metodologias de desenvolvimento
- As fases de desenvolvimento de software
- Modelos de desenvolvimento

Ciclo de desenvolvimento

Na engenharia de software o desenvolvimento de um sistema é guiado por uma **metodologia** que organiza as atividades de acordo com as **fase** do ciclo de vida do projeto.

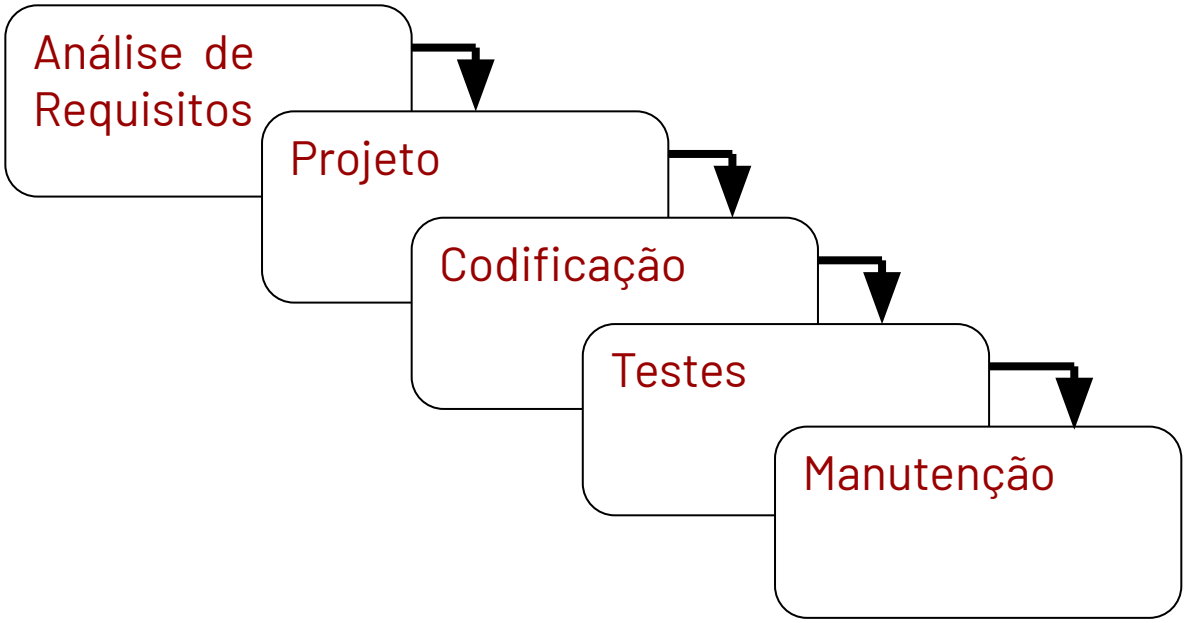
Existem diversos modelos de desenvolvimento, sendo atualmente a metodologia ágil a mais utilizada.

✅ Modelos

- Modelo cascata
- Modelo incremental
- Modelos evolucionários
- Modelo de prototipação
- Modelo RAD
- Modelo espiral
- Modelo de desenvolvimento ágil
  - Extreme programming (XP)
  - Scrum

O Modelo Cascata

Modelo mais antigo, segue uma abordagem, **seqüencial** de desenvolvimento de software, onde o resultado de uma fase constitui na entrada da outra.



Etapas

Análise de requisitos: coleta dos requisitos e compreensão das funcionalidades, refinados junto com o cliente

Projeto: tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade

Codificação: tradução das representações do projeto para linguagem executável pelo computador

Testes: validação dos aspectos lógicos do software, garantindo que todas as instruções tenham sido testadas.

Manutenção: software sofre mudanças depois de entregue ao cliente, seja correções de erros ou acréscimos de funcionalidades.

Problemas com o Modelo Cascata

- Projetos reais raramente seguem o fluxo seqüencial.
- No início do projeto é difícil estabelecer explicitamente todos os requisitos.
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento.

Contribuições do Modelo Cascata

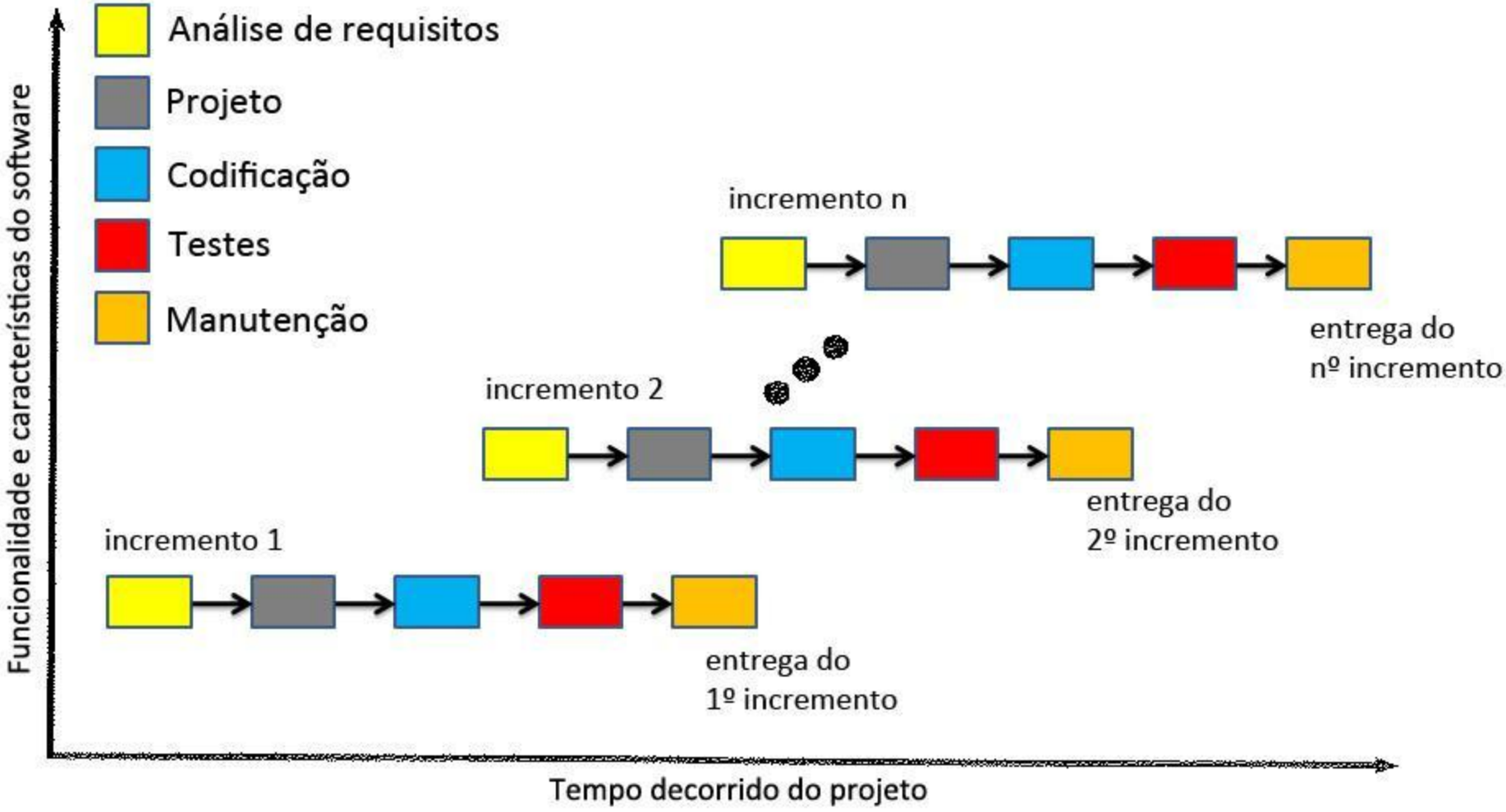
O modelo Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:

- imposição de disciplina, planejamento e gerenciamento.
- a implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos.

Modelo Incremental

- Combina elementos do modelo cascata com a filosofia iterativa da prototipação
- O objetivo é trabalhar junto do usuário para descobrir seus requisitos, de maneira incremental, até que o produto final seja obtido
- A versão inicial é freqüentemente o núcleo do produto (a parte mais importante)
- A evolução acontece quando novas funcionalidades são adicionadas

Modelo Incremental



Modelo Incremental

- Este modelo é interessante quando há dificuldade em fazer o levantamento de todos os requisitos
- O modelo incremental é mais apropriado para sistemas de pequeno porte
- Tem por objetivo apresentar um produto operacional a cada incremento.
- Os primeiros incrementos são versões simplificadas do produto final, mas são funcionais permitindo a avaliação pelo usuário

Modelo Evolucionário

- Software evolui com o passar do tempo
- Requisitos mudam frequentemente a medida que o desenvolvimento prossegue.
- Prazos reduzidos tornam impossível completar um produto com escopo grande.

Modelos evolucionários são iterativos.

Eles são caracterizados de forma a permitir aos engenheiros desenvolver versões cada vez mais completas do software. Exemplos:

- Modelo de Prototipação
- Modelo Espiral

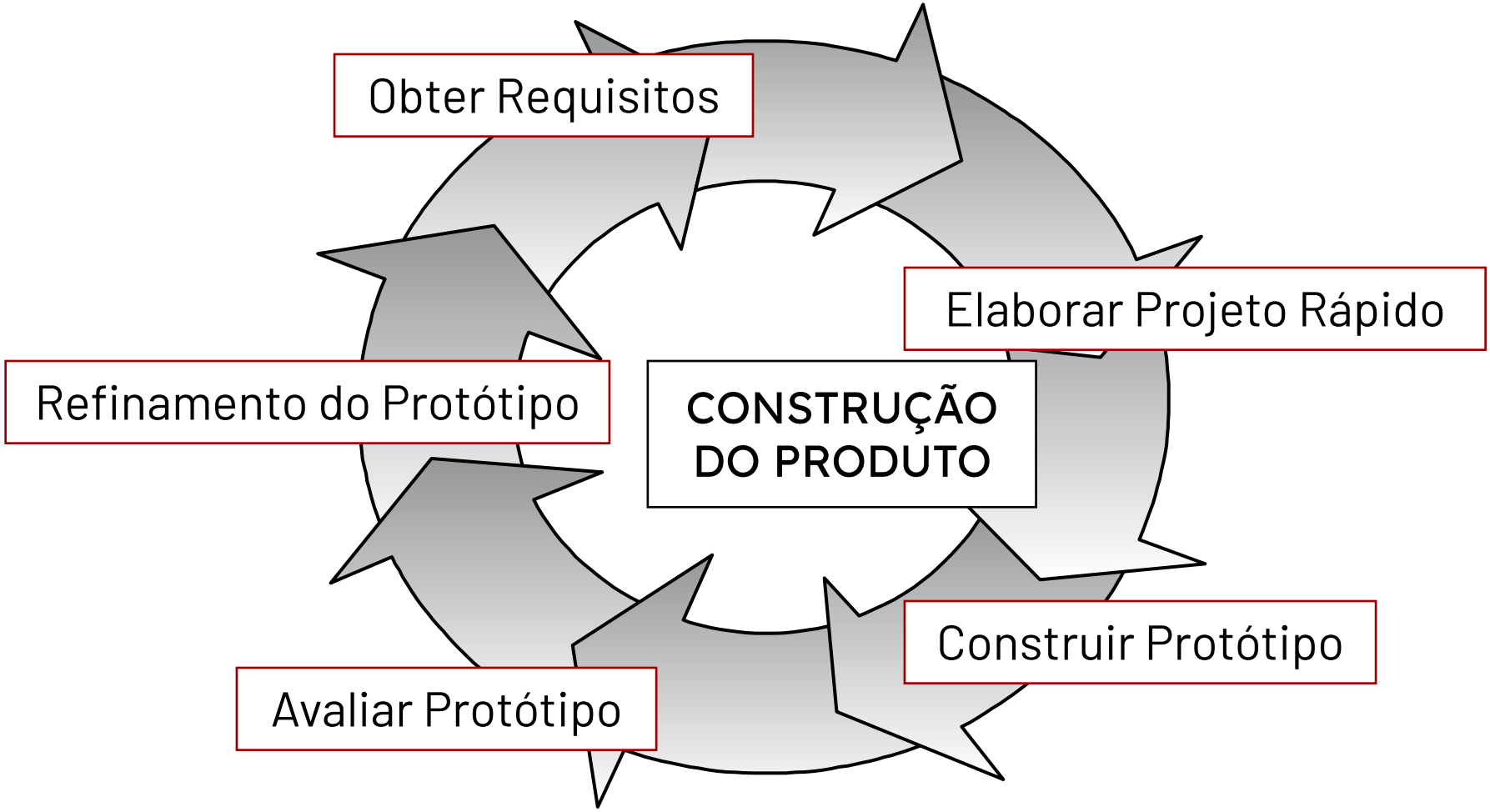
Modelo de Prototipação

- O objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos
- Possibilita que o desenvolvedor crie um modelo (protótipo) do software que deve ser construído
- Apropriado quando o cliente não definiu ou não sabe exatamente os requisitos

Modelo de Prototipação

- Uma iteração de prototipagem é planejada rapidamente e também uma modelagem
- A prototipação extrai os aspectos do software que estarão visíveis para o cliente/usuário, ex.: layout
- O projeto rápido leva a construção de um protótipo que é implantado.
- O feedback é usado para refinar os requisitos do software.
- O protótipo serve como um mecanismo para identificação dos requisitos do software.

Modelo de Prototipação



Modelo de Prototipação

- Clientes e desenvolvedores gostam do paradigma de prototipagem.
- Usuário tem a experiência de um sistema “real” e os desenvolvedores conseguem construir “algo” imediatamente.

Problemas na prototipagem

- Cliente vê o que parece ser uma versão executável do software e ignora que aquela é apenas uma demonstração...
- O desenvolvedor na pressa de entregar um protótipo, acaba usando qualquer framework/biblioteca mais conveniente para apresentar "algo".
- Importante ficar claro entre cliente e desenvolvedor de que o protótipo é construído para servir como um mecanismo de definição dos requisitos, e portanto, depois será descartado.

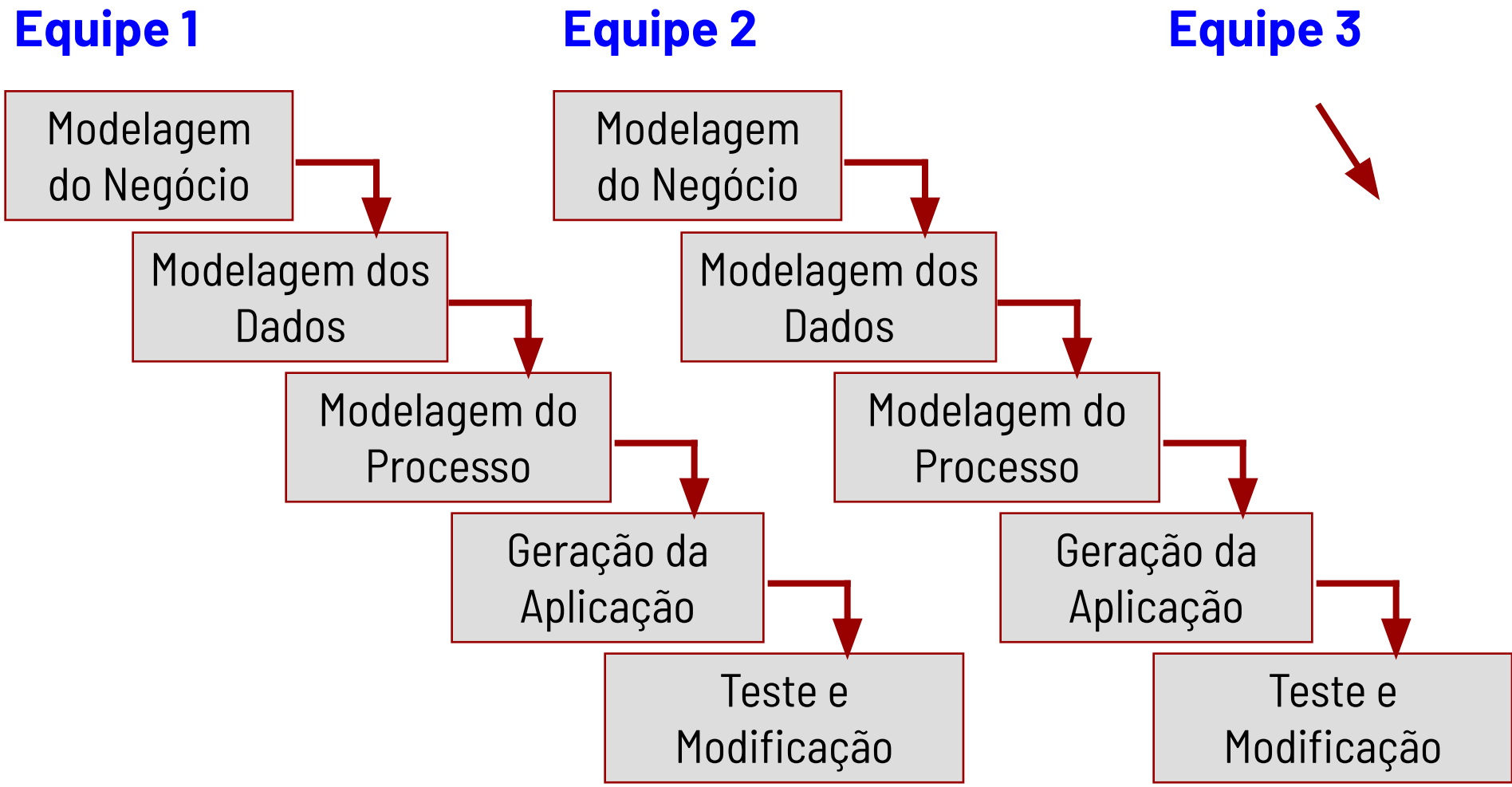
Modelo RAD

- Rapid Application Development – Desenvolvimento rápido de aplicações – é um modelo sequencial linear que enfatiza um ciclo de desenvolvimento extremamente **curto**.
- Rapidez é alcançada usando uma abordagem de construção baseada em componentes.
- Os requisitos devem ser bem entendidos
- Cada função principal pode ser direcionada para uma equipe RAD separada e então posteriormente integrada

Modelo RAD

- Enfatiza um ciclo de desenvolvimento curto.
- É uma adaptação de alta velocidade do modelo em cascata.
- Permite ter um sistema funcional dentro de um período de tempo pequeno (60 a 90 dias).

Modelo RAD


Equipe 1
Equipe 2
Equipe 3
Modelagem do Negócio
Modelagem dos Dados
Modelagem do Processo
Geração da Aplicação
Teste e Modificação
Modelagem do Negócio
Modelagem dos Dados
Modelagem do Processo
Geração da Aplicação
Teste e Modificação

Modelo RAD

Desvantagens:

- Nem todos os tipos de aplicação são apropriadas para o RAD
- Deve ser possível a modularização efetiva da aplicação
- Exige recursos humanos suficientes para todas as equipes

Modelo Espiral

- O modelo espiral acopla a natureza iterativa da prototipação com os aspectos controlados e sistemáticos do modelo cascata
- O modelo espiral é dividido em uma série de atividades de trabalho ou regiões de tarefa

Modelo Espiral

- O modelo em espiral assume que o processo de desenvolvimento ocorre em ciclos, cada um contendo fases de avaliação e planejamento.
- Cada ciclo entrega uma versão de um software executável.
- O modelo acrescenta, um novo elemento – a análise de riscos.
- Considerado uma abordagem mais realística para o desenvolvimento de sistemas de grande escala.

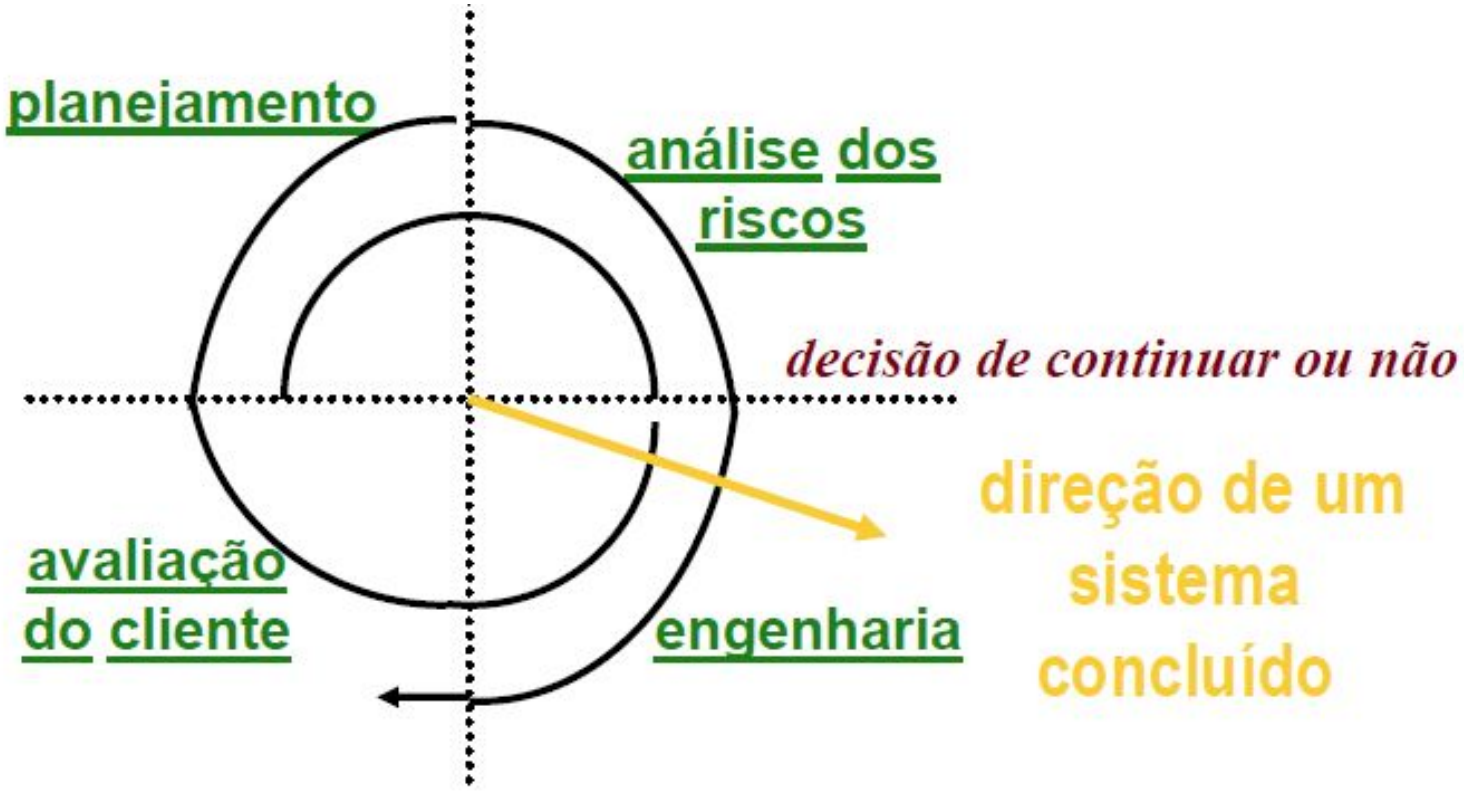
Modelo Espiral

- Sua principal inovação é guiar o processo de desenvolvimento com base em análise de riscos e planejamento que é realizado durante toda a evolução do desenvolvimento.
- Riscos são circunstâncias adversas que podem surgir durante o desenvolvimento de software impedindo o processo ou diminuindo a qualidade do produto.
- São exemplos de riscos:
  - pessoas que abandonam a equipe de desenvolvimento,
  - ferramentas que não podem ser utilizadas,
  - falha em equipamentos usados no desenvolvimento ou que serão utilizados no produto final, etc.

Etapas do Modelo Espiral

- Planejamento: definição dos objetivos, alternativas e restrições.
- Análise dos Riscos: análise de alternativas e identificação dos riscos sob o ponto de vista técnico e de gerência.
- Engenharia: desenvolvimento do produto.
- Avaliação do Cliente.

Modelo Espiral



Vantagens do modelo Espiral

- Permite que o projetista e cliente possam entender e reagir aos riscos em cada etapa evolutiva.
- modelo em espiral permite que ao longo de cada iteração se obtenham versões do sistema cada vez mais completas, recorrendo à prototipagem para reduzir os riscos.

Desvantagens

- Avaliação dos riscos exige experiência.
- O modelo em espiral pode levar ao desenvolvimento em paralelo de múltiplas partes do projeto, cada uma sendo abordada de modo diferenciado, por isso é necessário o uso de técnicas específicas para estimar e sincronizar cronogramas, bem como para determinar os indicadores de custo e progresso mais adequados.

Metodologia Ágil

No desenvolvimento ágil releases são frequentes mesmo com pequenas mudanças, onde o feedback do usuário é essencial na detecção de bugs e falhas, destacando-se:

- Indivíduos e interações em vez de processos e ferramentas.
- Softwares funcionando em vez de documentação abrangente.
- Colaboração do cliente em vez de negociação de contratos.
- Resposta a modificações em vez de seguir um plano.

Suposições

- É difícil prever antecipadamente quais requisitos vão persistir e quais serão modificados.
- Para muitos tipos de software o projeto e a construção são intercalados, ou seja, devem ser realizados juntos.
- É difícil prever o quanto de projeto é necessário antes que a construção seja usada para comprovar o projeto.
- Análise, projeto, construção e testes não são tão previsíveis como gostaríamos.

Adaptação

Como criar um processo que possa gerenciar a imprevisibilidade?

- Para viabilizar a adaptação incremental, uma equipe ágil requer o feedback do cliente.
- Um catalisador efetivo para o feedback do cliente é um protótipo operacional, ou uma porção do sistema operacional.
- Incrementos de software devem ser entregues em curtos períodos de tempo.
- Isso permite a avaliação do cliente e assim obter o feedback para lidar melhor com as imprevisibilidades.

Metodologia Ágil

- Uma equipe ágil é aquela capaz de responder prontamente a modificações. Modificações do software, da equipe, de novas tecnologias, etc...
- Rápida entrega de software operacional
- Adota os clientes como parte da equipe de desenvolvimento, trabalhando para eliminar a atitude nós e eles.

12 princípios da agilidade

1. A maior prioridade é a satisfação do cliente, desde o início, através da entrega contínua de software valioso.
2. Modificações de requisito são bem vindas.
3. Entrega de software funcionando, de modo frequente, a cada 2 semanas ou até 2 meses.
4. Equipe de negócio e os desenvolvedores devem trabalhar juntos, diariamente, durante todo o projeto.
5. Construção de projetos em torno de indivíduos motivados.
6. O método mais eficiente e efetivo de levar informação para a equipe é a conversa face a face.

12 princípios da agilidade

7. Software funcionando é a principal medida de progresso.
8. Promover o desenvolvimento sustentável, através de um ritmo constante.
9. Atenção contínua a excelência técnica
10. Simplicidade
11. As melhores arquiteturas, requisitos e projetos surgem de equipes auto-organizadas.
12. Em intervalos regulares, a equipe reflete sobres como se tornar mais efetiva.