SISTEMAS DE INFORMAÇÃO

Engenharia de Software

Prof. Ma.Claudete Werner

Processo de software

- *Sommerville (2014)* afirma que um "processo de software é um **conjunto de atividades** relacionadas que levam à **produção de um produto de software".**

- *Pressman (2011)* complementa dizendo que "processo **é uma metodologia** para as atividades, ações e tarefas necessárias para desenvolver um **software de alta qualidade**".

Processo de software

- É uma **série de passos** (um **ROTEIRO**).

- Para criar **EM TEMPO** um **SOFTWARE** de **ALTA QUALIDADE**, sem estourar o **ORÇAMENTO**.

- Como “escolher“ um processo?
  - ✓ As **CARACTERÍSTICAS DA APLICAÇÃO** (domínio do problema, tamanho, complexidade etc);
  - ✓ A **TECNOLOGIA** a ser adotada na sua construção (paradigma de desenvolvimento, linguagem de programação, mecanismo de persistência etc), a organização;
  - ✓ **ONDE** o produto será desenvolvido;
  - ✓ O **PERFIL DA EQUIPE** de desenvolvimento.

Engenharia de Software - Ciclo de vida

- Independente da metodologia que será utilizada para desenvolvimento do software, sempre existirá um *ciclo de vida* existente em todas as metodologias.

- *Ciclo de vida* pode ser definido como uma **"estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção** de um produto de software, abrangendo a vida do sistema desde a definição de seus requisitos até o término de seu uso".

Engenharia de Software - Ciclo de vida

- Em geral, um **ciclo de vida** envolve as etapas:
  - ✓ Planejamento
  - ✓ Análise e Especificação de Requisitos
  - ✓ Projeto
  - ✓ Implementação
  - ✓ Testes
  - ✓ Entrega e Implantação
  - ✓ Operação
  - ✓ Manutenção

Ciclo de vida - Planejamento

- Fornece uma estrutura que possibilita ao gerente fazer estimativas iniciais de recursos, custos e prazos;
- O escopo do software é estabelecido;
- Um plano de projeto deve ser elaborado configurando o processo a ser utilizado;
- Esta atividade faz parte da ***gerência de projeto.***

Ciclo de vida - Análise e Especificação de Requisitos

- O escopo do software é refinado;
- Nessa fase, a interação entre quem desenvolverá e o cliente é muito grande, contudo não se discute como será feito o software, mas sim o que ele deve fazer.
- Devem ser analisados o domínio do problema e o domínio da solução.
- São usadas as histórias de usuário (*User Stories*), que são pequenas histórias escritas para auxiliar na definição do requisito entre quem desenvolve e o cliente.
- Todo esse material é compilado e gera um relatório que servirá para os tomadores de decisão definirem, ou não, a continuidade e o desenvolvimento do software.

Ciclo de vida – Projeto

- Utiliza a fase anterior como insumo.
- Essa fase é voltada ao programador do software.
- Definição de como seria a interface que o usuário opera, quais cores seriam utilizadas na interface, como seria o fluxo de funcionamento de cada botão existente na interface, em qual banco de dados ficariam os dados gerados.
- O projeto, diferente da fase anterior, define como as coisas acontecerão.

Ciclo de vida – Implementação

- Essa fase deve traduzir o projeto em um software, utilizando ferramentas e linguagens adequadas.
- Dadas as inúmeras metodologias de desenvolvimento, cada programador pode desenvolver o código do sistema de uma forma diferente

Ciclo de vida – Testes

- **Teste de unidade:** cada componente é testado individualmente, sem conexão com outros componentes.
  - ✓ Exemplo: testar apenas o método que faz a soma dos valores das vendas, dados dois números, ele deve retornar a soma dos dois.
- **Teste de módulo:** um módulo é um agrupamento de pequenos componentes, que tem uma função específica.
  - ✓ Exemplo: testar a geração do relatório de vendas de um produto.
- **Teste de subsistemas:** são testados módulos integrados que controlam as interfaces do sistema.
  - ✓ Exemplo: testar as interfaces do sistema de compra e venda.

Ciclo de vida – Testes

- **Teste de sistema:** testa a integração dos subsistemas que formam o sistema principal.
  - ✓Exemplo: realizar login, operações e geração de relatórios em um sistema.
- **Teste de aceitação:** testes realizados com dados reais fornecidos pelos clientes. Último teste antes de colocar o sistema em operação.
  - ✓Exemplo: quando há muitos usuários na base e o login não acontece de forma instantânea.

Ciclo de vida
Entrega e Implantação

- O software deve ser instalado em ambiente produção.
- Envolve:
  - ✓ Treinamento de usuários;
  - ✓ Configuração do ambiente de produção;
  - ✓ Conversão bases de dados (se necessário).

Ciclo de vida – Operação

Após os testes, entrega e implantação, o software passa a ser utilizado de fato em um ambiente de produção.

http://resgatecontabil.com.br/noticias_detalhe
hp?id=381

Ciclo de vida – Manutenção

- *Pressman* diz que “a manutenção começa quase sempre imediatamente. O software é liberado para os usuários finais e em alguns dias, os relatos de bugs começam a chegar à organização de Engenharia de Software”.
  - ✓Manutenção corretiva.
  - ✓Manutenção adaptativa.
  - ✓Manutenção evolutiva.
  - ✓Manutenção preventiva.

Ciclo de vida – Manutenção

- **Manutenção corretiva:**
  - ✓ Correção dos erros encontrados pelo cliente e que não foram detectados nas fases de testes anteriores.
- **Manutenção adaptativa:**
  - ✓ Adaptação do software relacionado as mudanças do ambiente externo.
- **Manutenção evolutiva:**
  - ✓ Mudanças não previstas nos requisitos originais, visando a melhorias de desempenho e novas funcionalidades.
- **Manutenção preventiva:**
  - ✓ A iniciativa parte da equipe de desenvolvedores, visando evitar futuros problemas e melhorias em futuras manutenções.

Modelos de Processos

- Modelos de Processos podem ser divididos em dois grandes modelos:

  - Modelos de Processos Tradicionais e
  - Modelos de Processos Ágeis,

Cada Modelos de Processo possui características específicas e inerentes.

Modelos de Processos

- **Modelos de Processos Tradicionais**

- É possível observar que o produto computacional de software, possui um processo mais extenso de desenvolvimento, todo o desenvolvimento deste produto de software possui etapas que devem ser seguidas, e só após todas etapas finalizadas é que se é possível, realizar a implementação desde produto de software no cliente para sua utilização. (SOMMERVILLE, 2011)

Modelos de Processos

- **Modelos de Processos Tradicionais**

No Modelo de Processo Tradicional, entende-se que o produto só faz sentido quando é entregue em sua totalidade ao cliente, ou seja, apenas com 100% do projeto cumprido é que o *software será implantado para uso do cliente.* (*REIS, 2014)*

Modelos de Processos

Modelos de Processos Tradicionais

Os **Modelos de Processos Tradicionais** são os paradigmas **mais antigos** da Engenharia de *Software. Entretanto muitos autores afirmam que o Modelo de* Processo Tradicional **não é o mais adequado** para o desenvolvimento dos **produtos de *softwares mais modernos***, *pois propõe um fluxo de processos linear,* característica esta, que não satisfaz a maioria do desenvolvimento de produtos de *softwares modernos, onde são observadas contínuas alterações, softwares que* evoluem rapidamente e em prazos apertados, entretanto no **desenvolvimento de produtos de *software*** *em que possui* ***requisitos bem definidos e estáveis***, *este* Modelo de Processo possui **aplicabilidade**. (PRESSMAN, 2011)

Modelos de Processos

- **Modelos de Processos Ágil**

Modelo de desenvolvimento, com maior **rapidez de entrega** ao cliente. Este modelo de processo caracteriza-se, pela entrega de **"módulos"** do *software que está em* desenvolvimento ao cliente, dessa forma o **cliente, pode avaliar** o que já foi desenvolvido e implementado em determinado **módulo**, proporcionando a utilização imediata desses "módulos", até que o *software seja entregue em sua totalidade com* todas as funcionalidades e módulos. (SOMMERVILLE, 2011)

Modelos de Processos

Modelos de Processos Ágil

Metodologias Ágeis se desenvolveram em um esforço para sanar fraquezas reais e perceptíveis da Engenharia de *Software Convencional.* (PRESSMAN, 2011)

Modelos de Processos

Modelos de Processos Ágil

- Jacobson (1993 apud Pressman, 2011), constata que a **agilidade** se tornou a **palavra da moda** quando se descreve um moderno processo de *software.*
- *Para se* obter agilidade **todo mundo** que faz parte do projeto tem de **ser ágil**. Uma **equipe ágil** é aquela **rápida e capaz de responder apropriadamente a mudanças**, mudanças estas que compreendem o desenvolvimento do *software, as funcionalidades que* compõe o novo produto de *software, mudanças nos membros da equipe, mudanças* inerentes a novas tecnologias e mudanças que poderão ter um impacto no projeto.

Desenvolvimento Ágil de Software

- As empresas operam em um ambiente global com mudanças rápidas.
- Softwares fazem parte de quase todas as operações de negócios.
- Desta forma novos softwares são desenvolvidos rapidamente para as empresas obterem proveito de novas oportunidades e responder as pressões competitivas.
- Com a mudança rápida do mercado é impossível obter um conjunto completo de requisitos de software estável.
- Processos de desenvolvimento rápido de software são concebidos para produzir, de forma rápida, software úteis.

Desenvolvimento Ágil de Software

- *Desenvolvimento ágil de software* ou *Método ágil* é um conjunto de metodologias de desenvolvimento de software. O desenvolvimento ágil, tal como qualquer metodologia de software, tem como objetivo providenciar uma estrutura conceitual para reger projetos de engenharia de software.
- As definições modernas de desenvolvimento de *software ágil* evoluíram a partir da metade de 1990 como parte de uma reação contra métodos "pesados", caracterizados por uma pesada regulamentação, regimentação e micro gerenciamento usado o modelo em cascata para desenvolvimento.
- A expressão "*Metodologias Ágeis*" tornou-se conhecida em 2001, quando especialistas em processos de desenvolvimento de software se reuniram para estabelecer princípios e características comuns destes métodos. Assim foi criada a "***Aliança Ágil***" e efetuou-se o estabelecimento do "***Manifesto Ágil***".

Manifesto Ágil

Métodos ágeis focam em **simplicidade**, **software funcional** no início das iterações, flexibilidade e intensa comunicação tanto internamente quanto com clientes.

Indivíduos e interações são mais importantes que *processos e ferramentas.*

Software funcionando é mais importante do que *documentação completa e detalhada*.

Colaboração com o cliente é mais importante do que *negociação de contratos*.

Adaptação a mudanças é mais importante do que *seguir o plano inicial*.

Princípios dos métodos ágeis

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos dos sistema e avaliar suas interações.
Entrega incremental	O software desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe tem a liberdade de desenvolver suas próprias maneiras de trabalhar.
Aceitar as mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso o sistema projetado de acomodar tais mudanças.
Manter a simplicidade	Foco na simplicidade, tanto do software a ser desenvolvido como do processo de desenvolvimento. Trabalhar de forma ativa para eliminar a complexidade do sistema.

Metodologias ágeis

Métodos de desenvolvimento de software ágil aplicam desenvolvimento **iterativo e incremental**, **planejamento adaptativo**, flexibilidade e rápida resposta para mudanças.

Planejamento Adaptativo

- ✓ Planejamento adaptativo é adequado para desenvolvimento de novos produtos.
- ✓ Requisitos são escritos para um contexto e ambiente específico que estão em constante mudança.
- ✓ Desenvolvimento de novos produtos solicita um certo grau de pesquisa, inovação e criatividade.
- ✓ A tecnologia envolvida exige ferramentas que podem estar ou em desenvolvimento ou estabilização.

Desenvolvimento Iterativo e Incremental

Desenvolvimento Incremental é uma estratégia de planejamento estagiado em que várias partes do sistema são desenvolvidas em paralelo, e integradas quando completas.

Desenvolvimento Iterativo e Incremental

Desenvolvimento iterativo é uma estrategia de planejamento de retrabalho em que o tempo de revisão e melhorias de partes do sistema é pré-definido.

Metodologias Tradicionais X ágeis

E agora qual usar?

https://www.youtube.com/watch?v=5oFzY1_-pXI

Metodologias de Desenvolvimento de Software

- Conforme a necessidade de desenvolvimento de sistemas foi crescendo, as atividades realizadas pelos responsáveis no desenvolvimento se repetiam a cada novo software.

- A partir do estudo dessas atividades, foram surgindo metodologias e processos, visando padronizar e agilizar o tempo do projeto.

- **Modelos de processo (Ciclo de vida do Software)**

- Algumas das propostas de Ciclo de Vida de Software atualmente utilizadas pela comunidade de Engenheiros de Software são:
  - » 1) **Modelo Sequencial Linear(Clássico ou Cascata)**
  - » **2) Modelo de prototipação**
  - » **3) Desenvolvimento Rápido de Aplicação - RAD**
  - » **4) Modelo Espiral**
  - » **5) Modelo incremental**
  - » **6) Desenvolvimento Formal de Sistemas**
  - » **7) Desenvolvimento Orientado a Reuso**

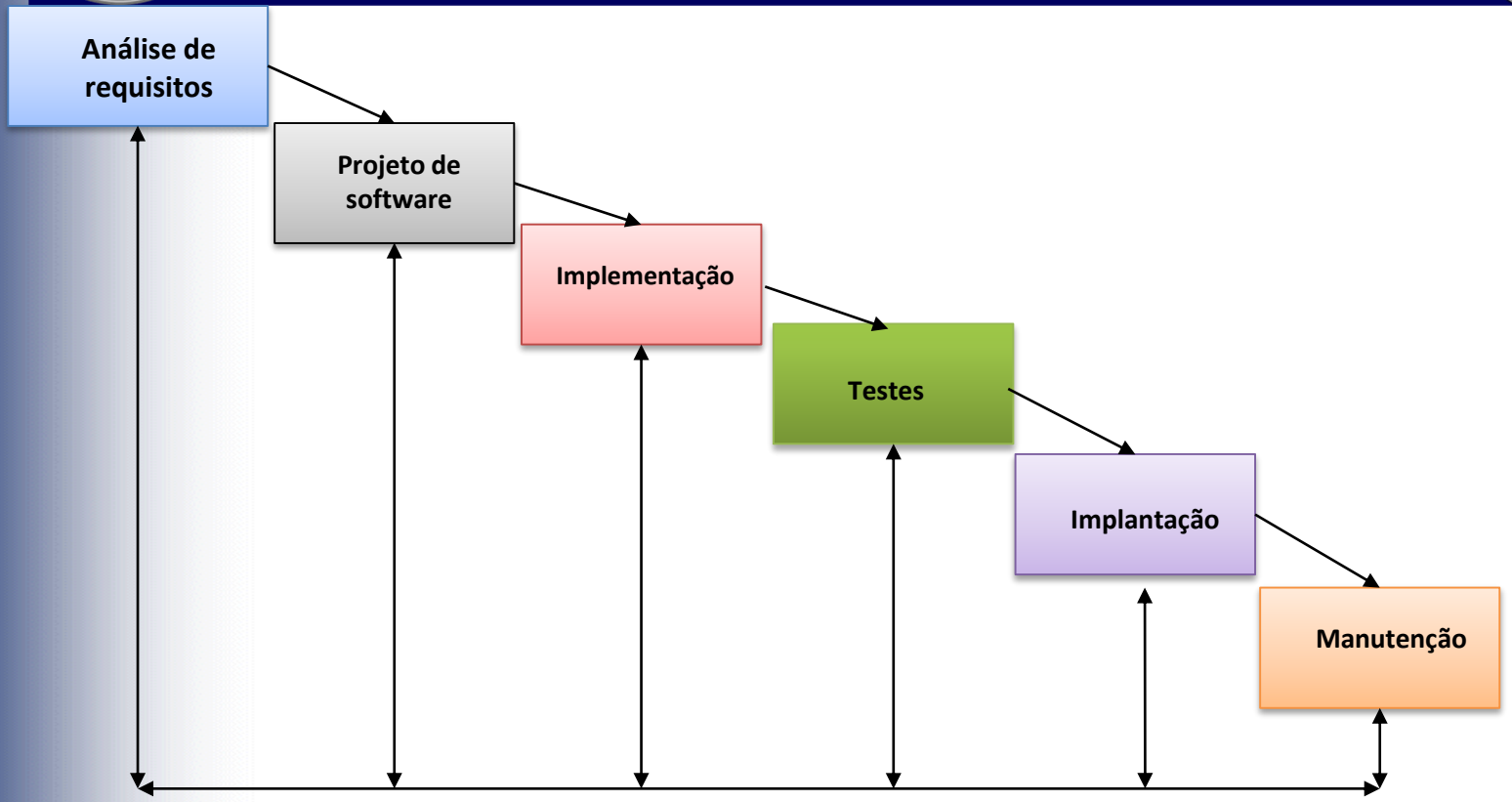
1. Modelo em Cascata ou Sequencial Linear

- Baseado em projetos de engenharia clássicos ou ciclo da engenharia convencional, foi consolidado em 1970 por Royce;
- Organiza o processo em uma **sequência linear de fases.**
- É utilizado quando os **requisitos são muito bem definidos**.
- O modelo de ciclo de vida em cascata ***foi o primeiro modelo*** a ser conhecido em engenharia de software e está na base de muitos ciclos de vida utilizados hoje em dia. Este consiste basicamente num modelo linear **em que cada passo deve ser completado antes que o próximo passo possa ser iniciado.**
- O modelo em cascata é um exemplo de um processo dirigido a planos, isto é, as atividades devem ser programadas e planejadas antes de serem iniciadas.

1. Modelo em Cascata ou Sequencial Linear

- Características:
  - ✓ Ajuda os desenvolvedores a descrever o que precisam fazer (útil);
  - ✓ Ajuda a explicar o processo de desenvolvimento aos clientes (simples e fácil);
  - ✓ Os produtos intermediários são finalizados para começar próximo estágio e servem de insumo para o seu desenvolvimento;
  - ✓ Seu enfoque está nos documentos e artefatos (requisitos, projetos, códigos).

1. Modelo em Cascata ou Sequencial Linear



Representação clássica do modelo em Cascata - Pressman

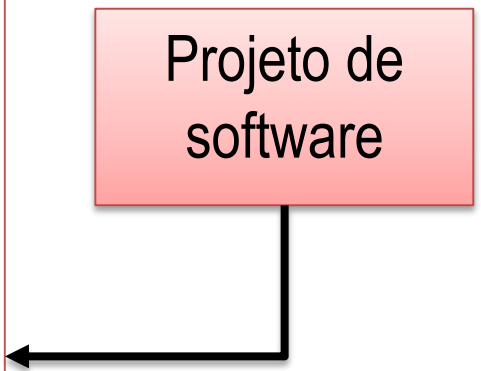
1. Modelo em Cascata ou Sequencial Linear

- As funções, as restrições e os objetivos do sistema são estabelecidos por meio da consulta aos usuários do sistema.
- Deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos.
- Os requisitos (para o sistema e para o software)são documentados e revistos com o Cliente.

Analise de requisitos

1. Modelo em Cascata ou Sequencial Linear

- Realiza todo o planejamento.
- São realizados os testes e as estimativas do desenvolvimento e criados os cronogramas para posterior acompanhamento do desenvolvimento do software.
- Define a arquitetura do sistema geral. É a tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie.



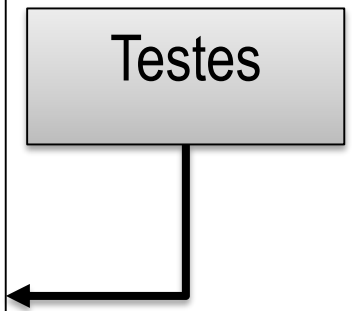
1. Modelo em Cascata ou Sequencial Linear

- Escreve os códigos, as interfaces com os clientes, a arquitetura do software e a estrutura de dados, para atender a todas as regras de negócio levantadas na primeira fase.
- Recomenda-se realizar testes unitários nos módulos durante essa fase.

Implementação

1. Modelo em Cascata ou Sequencial Linear

- O teste se concentra nos aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas;
- Se concentra também nos aspectos funcionais externos, para descobrir erros e garantir que a entrada definida produza resultados que concordem com os esperados.



1. Modelo em Cascata ou Sequencial Linear

- O sistema é instalado e colocado em operação.

Implantação

- Normalmente, é a fase mais longa do processo, pois tratará do suporte ao cliente, resolvendo os problemas quando o cliente der o feedback e implementando novos requisitos conforme necessidade.

Manutenção

1. Modelo em Cascata ou Sequencial Linear

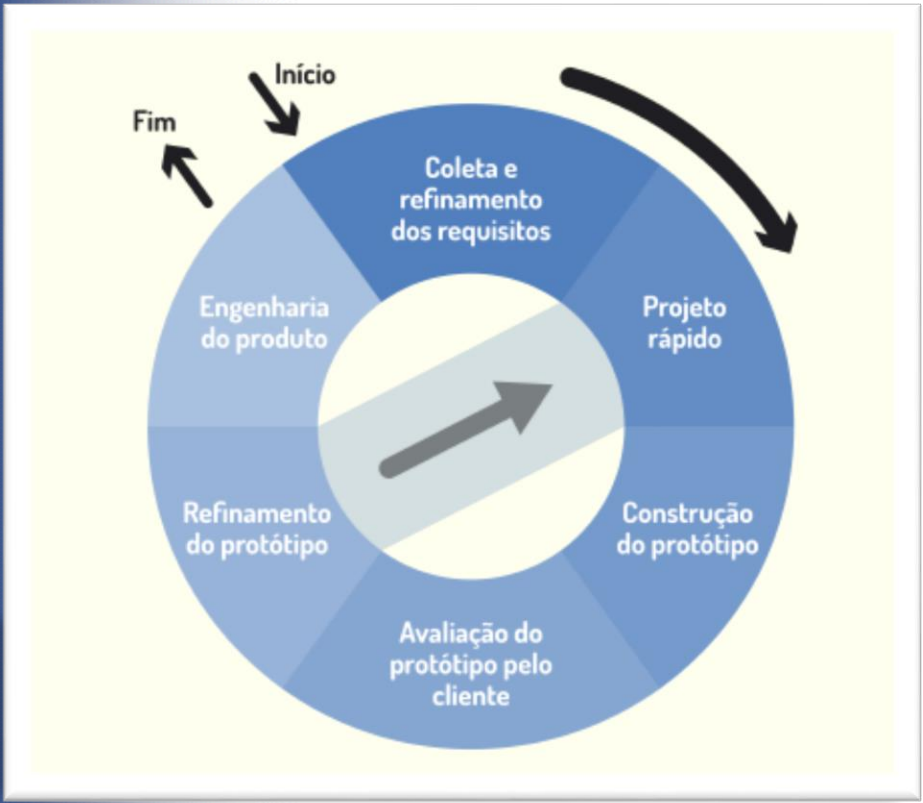
- Apesar do modelo ser encadeado, passando de fase apenas com o término da fase anterior, na prática, acaba acontecendo uma sobreposição de algumas fases, ou seja, uma fase serve de insumo para a próxima fase e pode descobrir um problema na fase anterior, levando a fase anterior a ser modificada.

- Essa abordagem apresenta a ***vantagem de produzir uma documentação muito consistente em cada fase, porém é recomendada apenas quando os requisitos são pré-estabeleci***dos, bem conhecidos e com pouca probabilidade de mudança com o passar do tempo.

2. Modelo de prototipação

- *Laudon (2014)* afirma que prototipagem “consiste em montar um sistema experimental rapidamente e sem muitos gastos para submetê-lo à avaliação de usuários finais. O protótipo é uma versão funcional de um sistema de informação, ou de parte dele, mas deve ser considerado apenas um modelo preliminar.”
- De acordo com *Pressman* (2011), o protótipo serve como um mecanismo para identificação dos requisitos do software, pois, se ele é elaborado, o desenvolvedor tenta usar partes de programas existentes ou aplicar ferramentas que possibilitem programas executáveis serem gerados rapidamente.

2. Modelo de prototipação

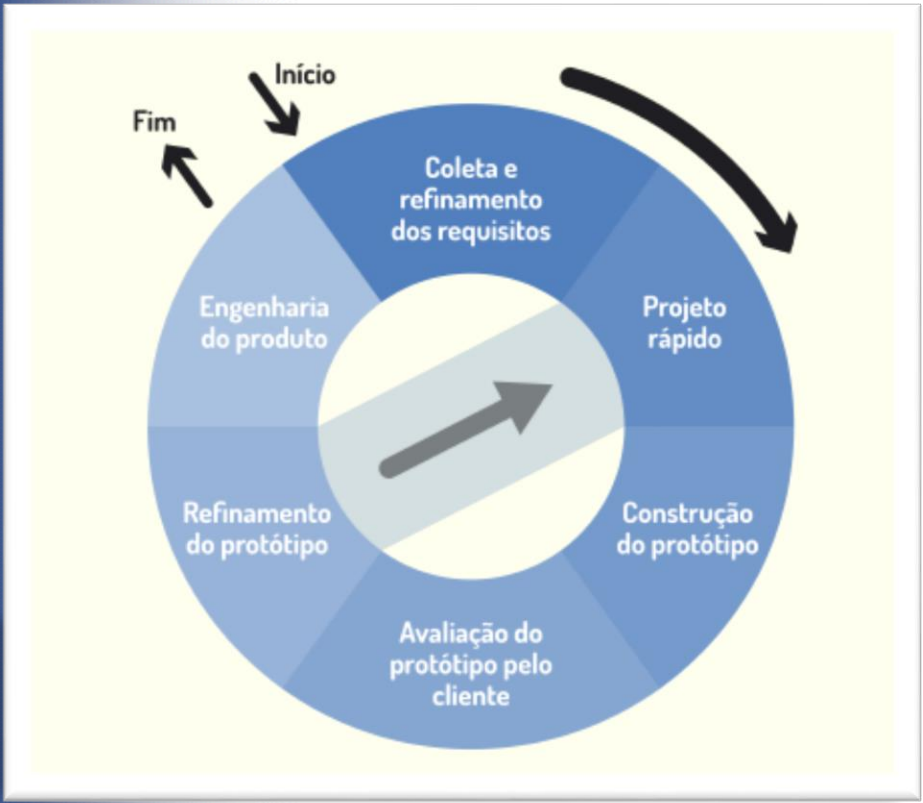


Pressman (2011)

Coleta e refinamento dos requisitos: ocorre a identificação dos requisitos junto ao cliente. Nessa fase, o engenheiro mantém contato direto com o cliente apenas para captar as necessidades básicas de informação.

Projeto rápido: fase que define como será a iteração da prototipação e na qual acontece a criação de um projeto rápido, que contempla a parte do software que estará visível ao cliente. Abordagens de entrada e formatos de saída.

2. Modelo de prototipação



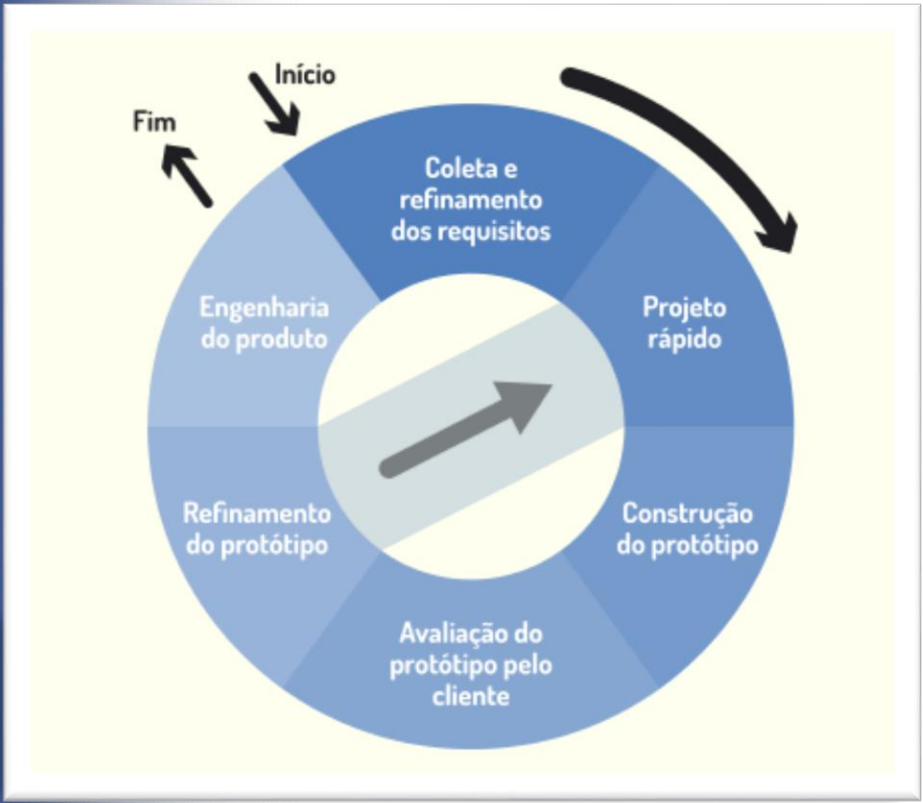
Pressman (2011)

Construção Protótipo: implementação rápida do projeto.

Avaliação do protótipo pelo cliente: fase de utilização do protótipo desenvolvido na fase anterior pelo usuário final, a fim de validar o software, sugerir mudanças e aperfeiçoamentos.

Refinamento do Protótipo: cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido. Ocorre neste ponto um processo de interação que pode conduzir à primeira atividade até que as necessidades do cliente sejam satisfeitas e o desenvolvedor compreenda o que precisa ser feito.

2. Modelo de prototipação



Pressman (2011)

Engenharia do Produto: identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.

2. Modelo de prototipação

- **Vantagens:**
  - ✓ Todo o requisitos de sistema não tem que ser completamente determinado antecipadamente
  - ✓ A satisfação e interação do cliente no desenvolvimento do protótipo;
  - ✓ Testes através do protótipo para atingir o objetivo final proposto.
- **Desvantagens:**
  - ✓ O processo de prototipação pode dar ao usuário final a impressão que praticamente qualquer sugestão pode ser implementada, não importa qual estágio do processo de desenvolvimento se está.
  - ✓ Além disso, para o usuário final não está claro o porquê da demora para entregar a aplicação final depois que uma versão demo do sistema foi exibida.
  - ✓ O desenvolvedor frequentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo. Depois de um tempo ele se familiariza com essas escolhas, e esquece que elas não são apropriadas para o produto final.

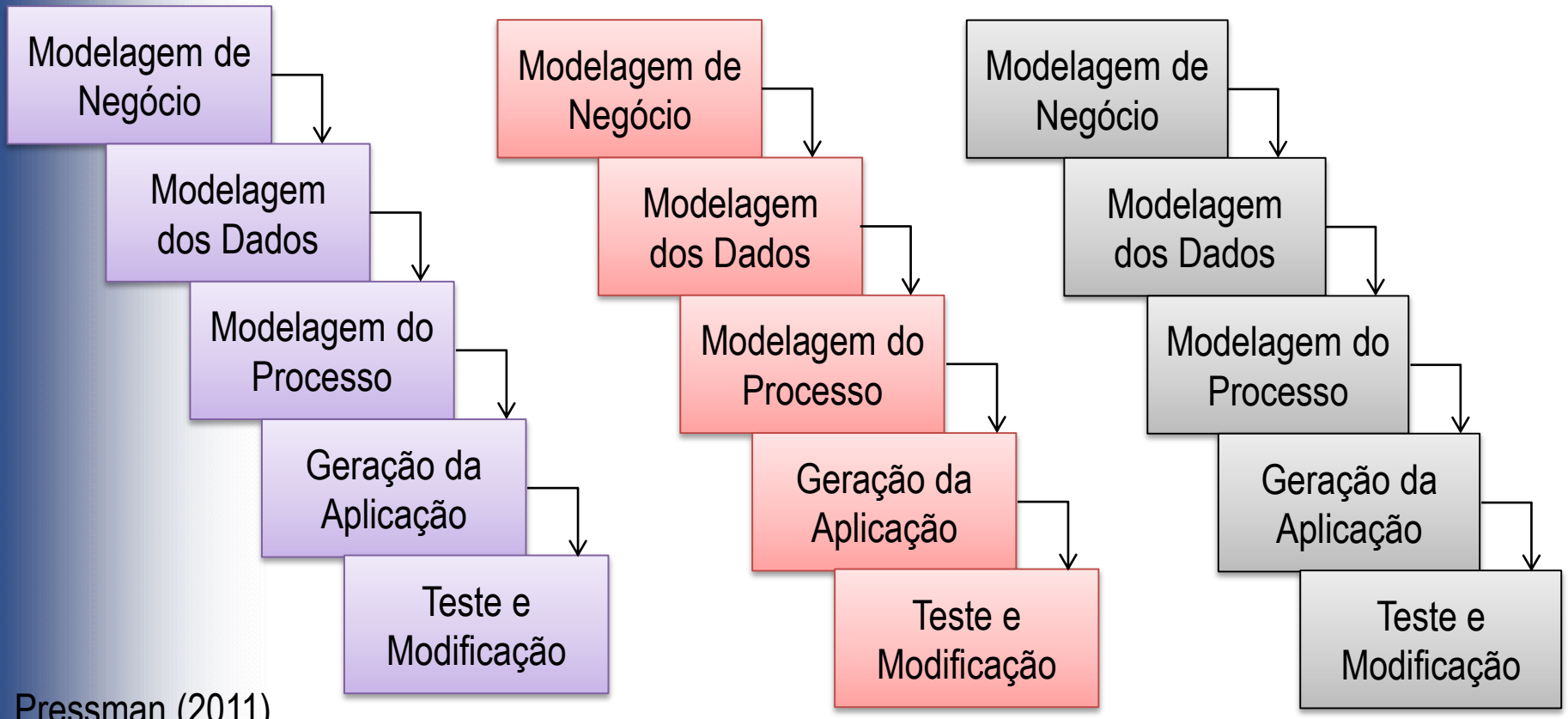
3. Desenvolvimento Rápido de Aplicação - RAD

- É o modelo sequencial linear mas que enfatiza um desenvolvimento extremamente rápido.
- A “alta velocidade” é conseguida através de uma abordagem de construção baseada em componentes.
- O desenvolvimento rápido de aplicação é adequado para construção de aplicações em curto espaço de tempo, utilizando o método incremental da prototipação, com o ciclo extremamente curto de desenvolvimento, que só é possível quando temos grande entendimento dos requisitos e o projeto é curto e simples.
- É um modelo de processo de desenvolvimento de software iterativo e incremental que enfatiza um ciclo de desenvolvimento extremamente curto (entre 60 e 90 dias).

3. Desenvolvimento Rápido de Aplicação - RAD

- A aplicação deve ser modularizada de forma que cada função deva ser completada em pelo menos 3 meses.
- Cada modulo pode ser alocado a uma equipe distinta e depois serem integradas para formar um todo.
- Criação e reutilização de componentes.
- No **RAD**, há o sequenciamento existente no modelo cascata e a ideia de apresentações parciais ao cliente para validação e continuação do desenvolvimento, vinda do modelo de prototipação, porém, nesse modelo, o que é apresentado ao cliente é, de fato, parte do software, e não apenas um protótipo.

3. Desenvolvimento Rápido de Aplicação - RAD



Pressman (2011)

4) MODELO ESPIRAL

- Engloba as melhores características Modelo Sequencial Linear com o da Prototipação, adicionando um novo elemento: a *ANÁLISE DOS RISCOS*
-
- Usa a *Prototipação,* em qualquer etapa da evolução do produto, como mecanismo de redução de riscos

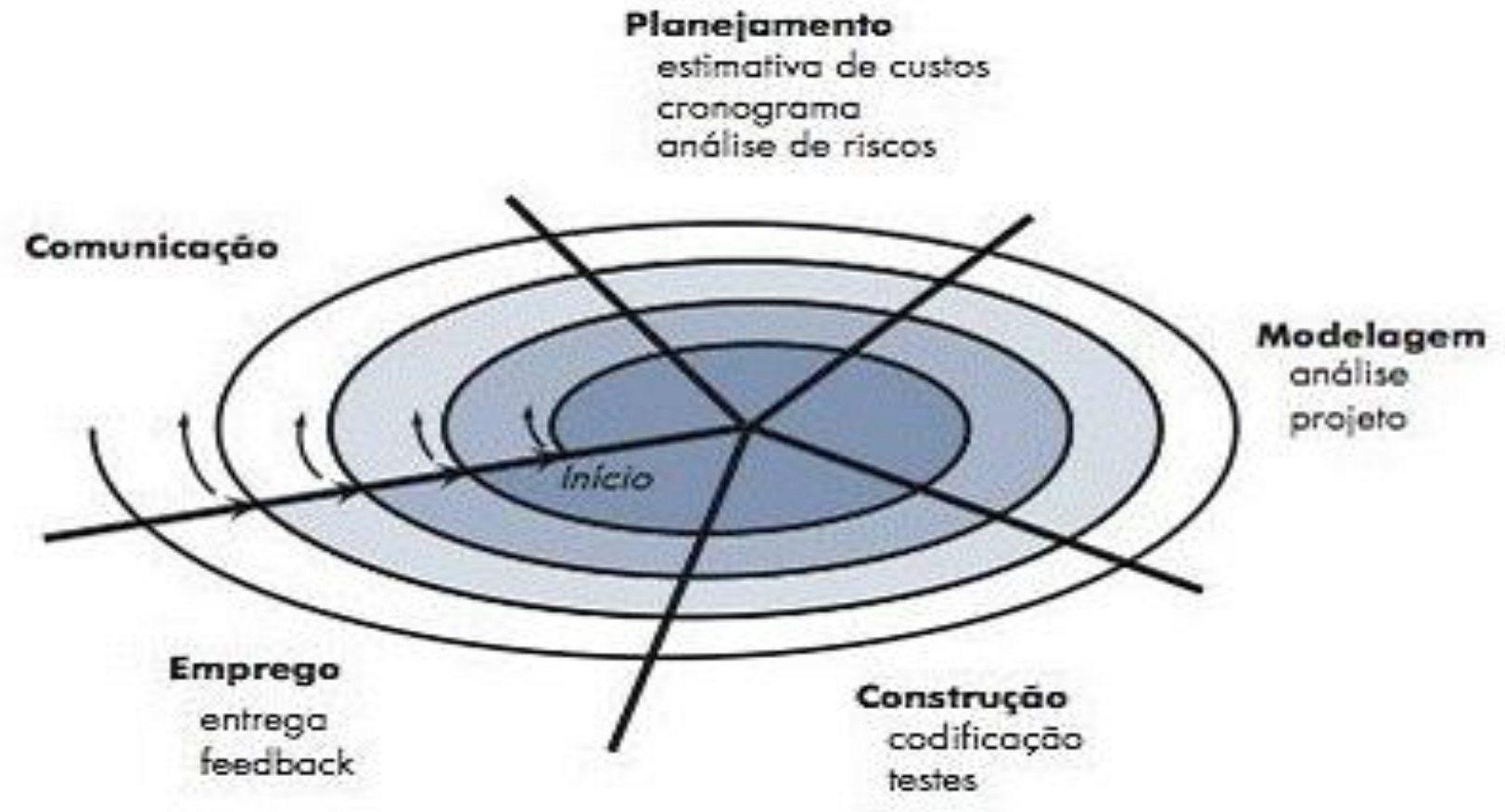
4) MODELO ESPIRAL

É um **modelo evolucionário** de processo de software que combina a **natureza iterativa da prototipagem** com os **aspectos controlados e sistemáticos do modelo Modelo Sequencial Linear** . Ele fornece o potencial para o desenvolvimento rápido de versões de software cada vez mais completas (PRESSMAN, 2011).

4) MODELO ESPIRAL

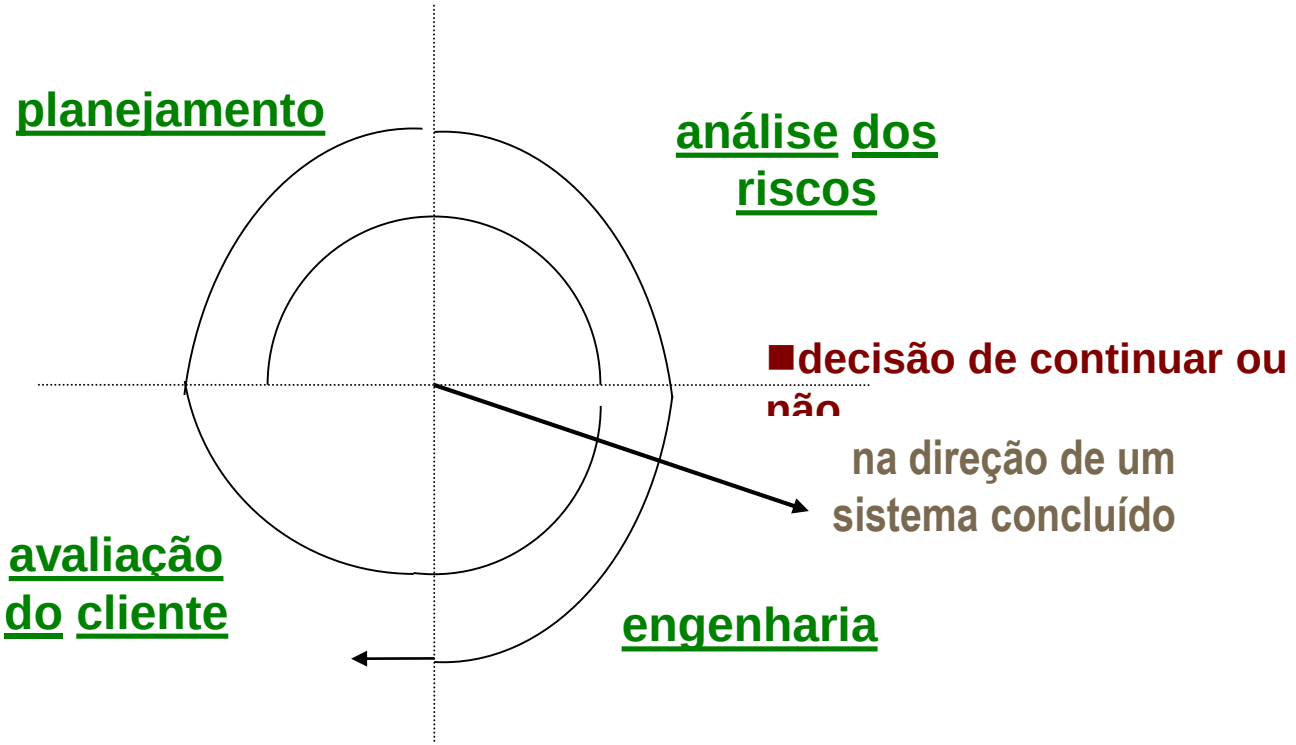
- Ainda para Pressman (2011), o modelo espiral, pode ser adaptado para aplicação ao longo da vida do software de computador. Assim, o primeiro circuito em volta da espiral poderia representar um projeto de desenvolvimento de conceitos que começa no centro da espiral e continua por várias iterações até que o desenvolvimento do conceito seja completado. Em resumo, a **espiral permanece operacional até que o software seja retirado de serviço**

4) MODELO ESPIRAL



Modelo Espiral. Fonte: Pressman (2011).

3) MODELO ESPIRAL



4) MODELO ESPIRAL

1- ***PLANEJAMENTO:*** **determinação dos objetivos, alternativas e restrições**

2- ***ANÁLISE DE RISCO:*** **análise das alternativas e identificação / resolução dos riscos**

3- ***CONSTRUÇÃO:*** **desenvolvimento do produto no nível seguinte**

4- ***AVALIAÇÃO DO CLIENTE:*** **avaliação do produto e planejamento das novas fases**

4) MODELO ESPIRAL

- É uma abordagem realística para o desenvolvimento de software em grande escala.

- Usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva.

4) MODELO ESPIRAL

- A cada iteração ao redor da espiral, versões progressivamente mais completas do software são construídas

5) Modelo incremental

- “O modelo incremental entrega uma série de versões chamadas de incrementos, que fornecem progressivamente mais funcionalidade para os clientes à medida que cada incremento é entregue” (PRESSMAN, 2011).

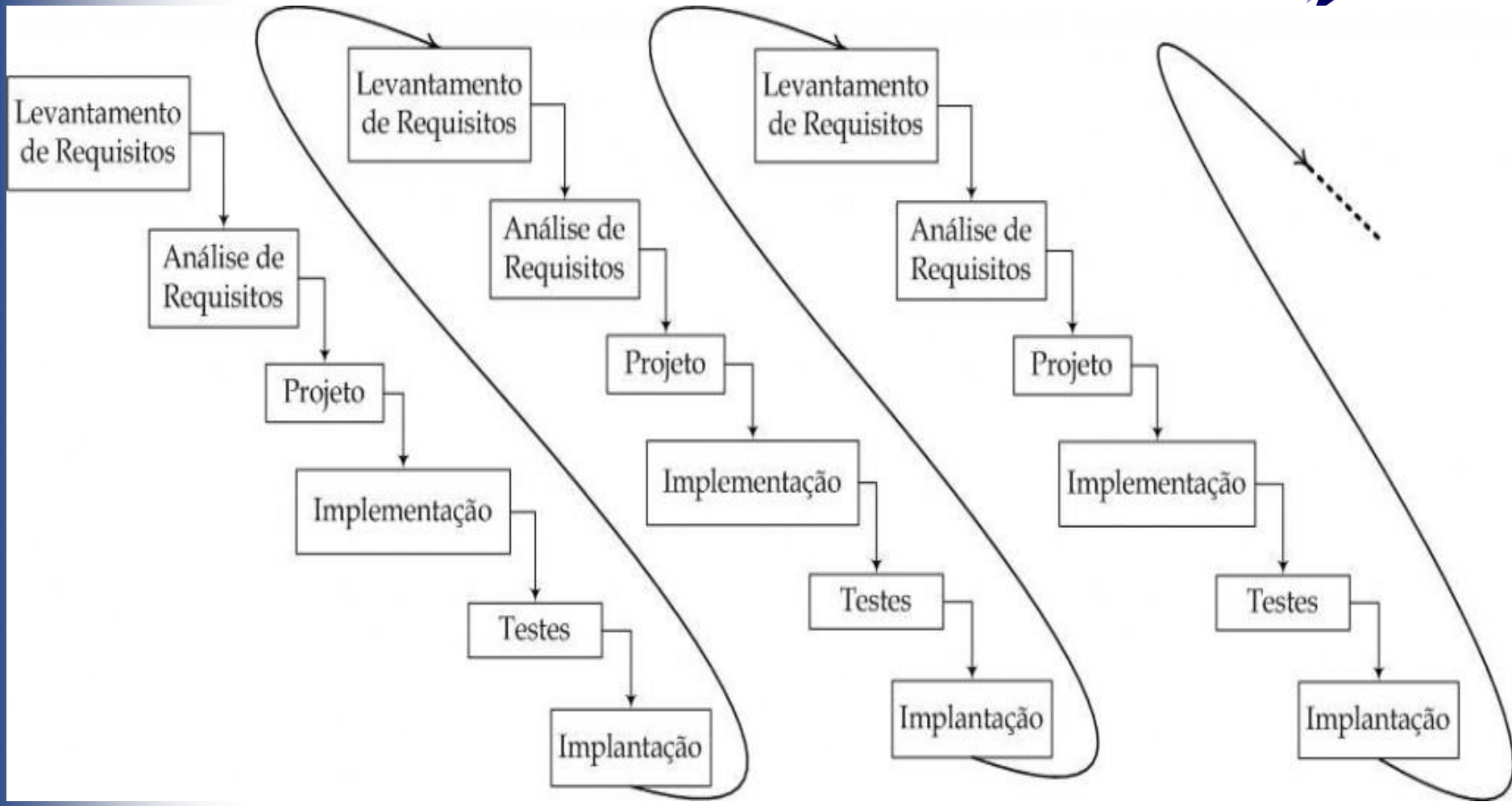
Para o autor, ele é particularmente útil quando não há mão-de-obra disponível para o projeto e os incrementos podem ser planejados para gerir os riscos técnicos.

5) Modelo incremental

- Segundo Pressman (2011), quando essa metodologia é aplicada, o **primeiro incremento** é frequentemente chamado de **núcleo do produto**, ou seja, os requisitos básicos são satisfeitos, mas características complementares deixam de ser elaboradas.

- Esse núcleo é usado pelo cliente e um plano é desenvolvido para o próximo incremento como resultado do uso e/ou avaliação. Este visa a modificação do núcleo para melhor satisfazer às necessidades do cliente e à elaboração de características e funcionalidades adicionais. Esse processo,, é repetido após a realização de cada incremento, até que o produto completo seja produzido

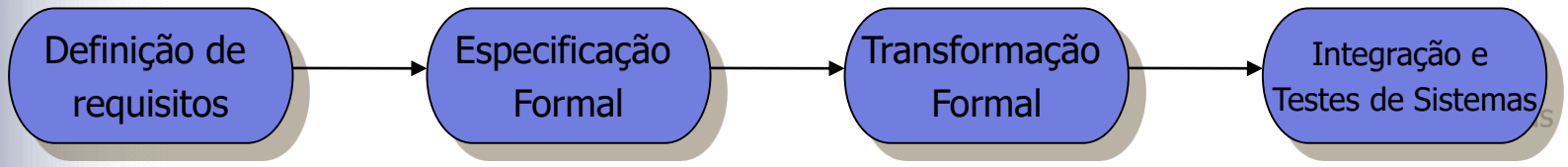
5) Modelo incremental



6- Desenvolvimento Formal de Sistemas

O desenvolvimento formal de sistemas é uma abordagem do desenvolvimento de software que tem algo em comum com o modelo Sequencial Linear, mas cujo processo de desenvolvimento tem como base a ***transformação matemática formal de uma especificação de sistemas em um programa executável.***

6- Desenvolvimento Formal de Sistemas



6- Desenvolvimento Formal de Sistemas

- Exemplos de sistemas nos quais métodos formais foram aplicados com sucesso:
  - Sistemas de Informação de controle de tráfego aéreo
  - Sistemas de sinalização de estrada de ferro
  - Os sistemas de naves espaciais

6- Desenvolvimento Formal de Sistemas

- Existem três níveis de especificação de sistemas:
  - Requisitos de usuários (são a especificação mais abstrata)
  - Requisitos de sistemas
  - Especificação de projeto de software (mais detalhada)
- Em geral, as especificações matemáticas formais encontram-se em algum ponto entre os requisitos de sistema e as especificações de projeto de software.

7- Desenvolvimento Orientado a Reuso

- Na maioria dos projetos de software, ocorre algum reuso de software. Isso, em geral, acontece informalmente, quando as pessoas que trabalham no projeto conhecem projetos ou códigos similares àquele exigido.

- Essa abordagem orientada a reuso conta com uma ampla base de componentes de softwares reutilizáveis.

Prof. Claudete Werner

claudete@unipar.br

- Mestre em Ciência da Computação
- Coordenadora do Curso de Sistemas de Informação da UNIPAR Paranavaí.
- Coordenadora do Núcleo de Software e Tecnologia do Semipresencial UNIPAR
- Atua na UNIPAR à 24 anos
- É consultora do MEC/INEP como Avaliador de Cursos de Ensino Superior da área da computação e de Instituições de Ensino Superior
- Representante Institucional da Sociedade Brasileira de Computação

- Bibliografia Básica:
-
- PRESSMAN, Roger S. **Engenharia de software.** Makron Books; São Paulo, 2011.
-
- [SOMMERVILLE, Ian. Engenharia de software, trad. Mauricio de Andrade. Sao Paulo : Addison Wesley , 2003