Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP
Câmpus Jacareí

Tecnologia em Análise e Desenvolvimento de Sistemas - ADS

1º Semestre de 2023

Engenharia de Software 1 – JCRESW1

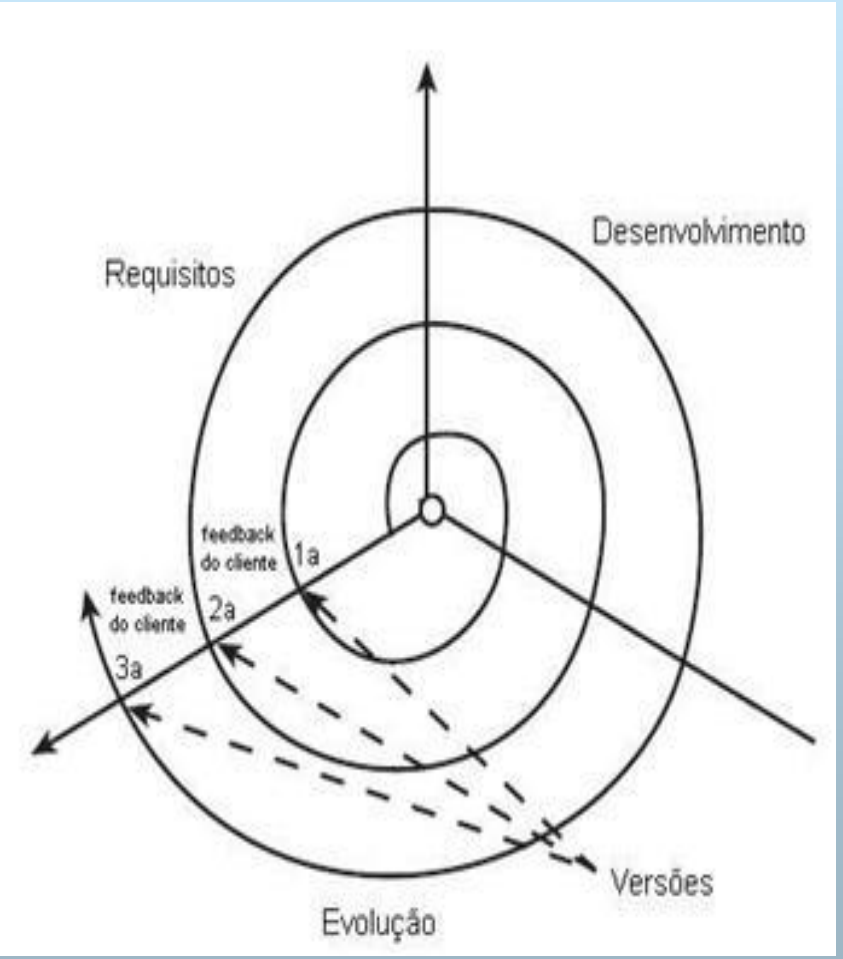
Prof. Lineu Mialaret

Aula 7: Processos de Desenvolvimento de Software (2)

Modelo Evolutivo

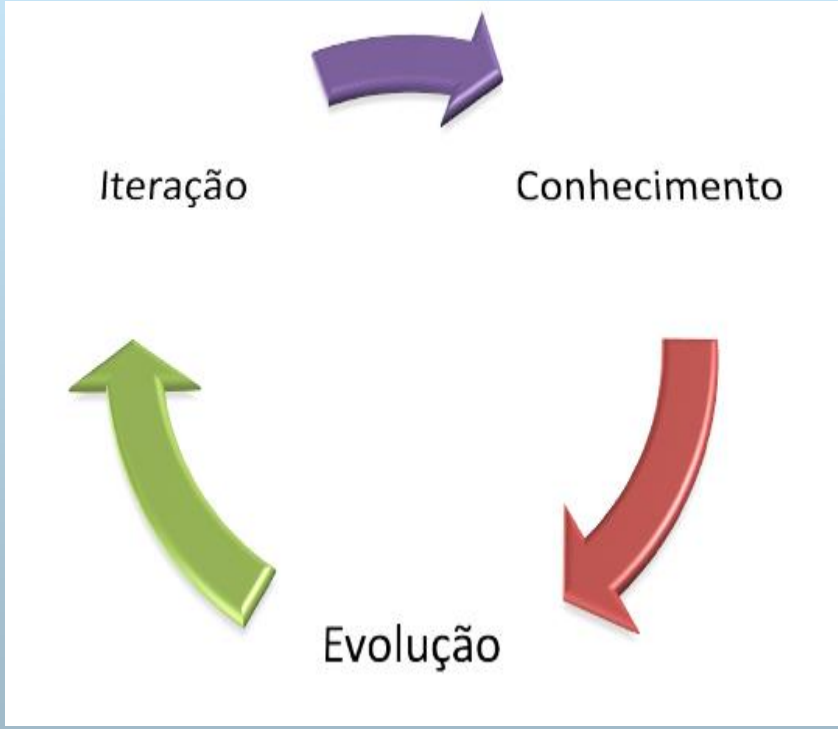
- No Modelo Evolutivo, requisitos são adquiridos em paralelo à evolução do sistema.
  - O Modelo Evolutivo parte do princípio que o cliente não expõe todos os requisitos, ou os requisitos não são tão bem conhecidos, ou os requisitos ainda estão sofrendo mudanças.
  - Desta forma, a análise é feita em cima dos requisitos obtidos até então, e a primeira versão é entregue ao cliente.
  - O cliente usa o software no seu ambiente operacional, e como *feedback*, esclarece o que não foi bem entendido e dá mais informações sobre o que precisa e sobre o que deseja (ou seja, mais requisitos).

Modelo Evolutivo (cont.)



- A partir deste *feedback*, novas etapas de análise, projeto e desenvolvimento são realizados, e uma segunda versão do software é entregue ao cliente que, novamente, retorna com mais *feedbacks*.
  - O software vai evoluindo, se tornando mais completo, até atender todas as necessidades do cliente dentro do escopo estabelecido.
  - Tem-se assim a versão final, pelo menos até novos requisitos aparecerem.

Modelo Evolutivo (cont.)



- As especificações evoluem a cada iteração.
  - A cada iteração, uma parte do software fica pronta.
  - O conhecimento sobre o software aumenta.
  - As especificações são evoluídas para retratar esse aumento de conhecimento sobre o que é o software.

Modelo Evolutivo (cont.)

- A participação constante do cliente é uma grande vantagem desse modelo, o que diminui o risco de má interpretação de requisitos dos modelos que só oferecem a primeira versão do software no final do processo.
  - Da mesma forma, o software já atende algumas necessidades do cliente muito mais cedo no processo.

Modelo Evolutivo (cont.)

- Não é dada muita ênfase à documentação, pois a geração de versões torna este trabalho muito árduo.
  - Além disso, como a análise de requisitos e desenvolvimento estão sempre acontecendo, a preocupação em documentar todo o processo pode fazer com que haja atrasos na entrega.

Modelo Evolutivo (cont.)

- Há uma alta necessidade de gerenciamento nesse tipo de modelo, pois a falta de documentação adequada, o escopo de requisitos não determinado, o software evoluindo e estando ao mesmo tempo em produção podem ter consequências negativas, com cenários ocasionando ônus em termos financeiros e de prazo.
- Exemplos desses cenários:
  - O sistema nunca termina, pois o cliente sempre pede uma alteração.
  - O sistema pode não ter uma estrutura robusta a falhas nem propícia a uma fácil manutenção, pelas constantes alterações.

Modelo Evolutivo (cont.)

- O cliente pode mudar de ideia radicalmente entre uma versão e outra ou revelar um requisito que exija uma versão bem diferente da anterior, fazendo com que toda a base (de dados ou de programação) precise ser revista.

Modelo Evolutivo (cont.)

- É muito importante que o cliente esteja ciente do que se trata este ciclo de vida e que sejam esclarecidos os limites de escopo, de tempo e de custos, para que não haja frustrações de expectativas.

Modelo de Reutilização de Software

- Na maioria dos projetos de desenvolvimento de sistemas há alguma reutilização de software.
- Com frequência, isso acontece informalmente quando desenvolvedores que trabalham em um projeto de desenvolvimento de software conhecem ou procuram algum código similar ao que é necessário.
  - Elas procuram por esse código, modificam-no conforme a necessidade e integram-no ao novo código que desenvolveram.

Modelo de Reutilização de Software (cont.)

- Desde os anos 2000, processos de desenvolvimento de software que se concentram no reuso de código existente passaram a ser amplamente utilizados (*standard approach*), gerando a assim chamada Engenharia de Software Baseada em Reutilização ou Engenharia de Software Orientada para Reutilização.
  - As abordagens orientadas à reutilização contam com bases de componentes de software reutilizáveis e *frameworks* de integração para a composição desses componentes.
  - Estes componentes são denominados de COTS (*Commercial-off-the-shef components*).
  - Estes componentes podem ser customizados para adaptar seu comportamento e funcionalidade para satisfazer requisitos de usuários.

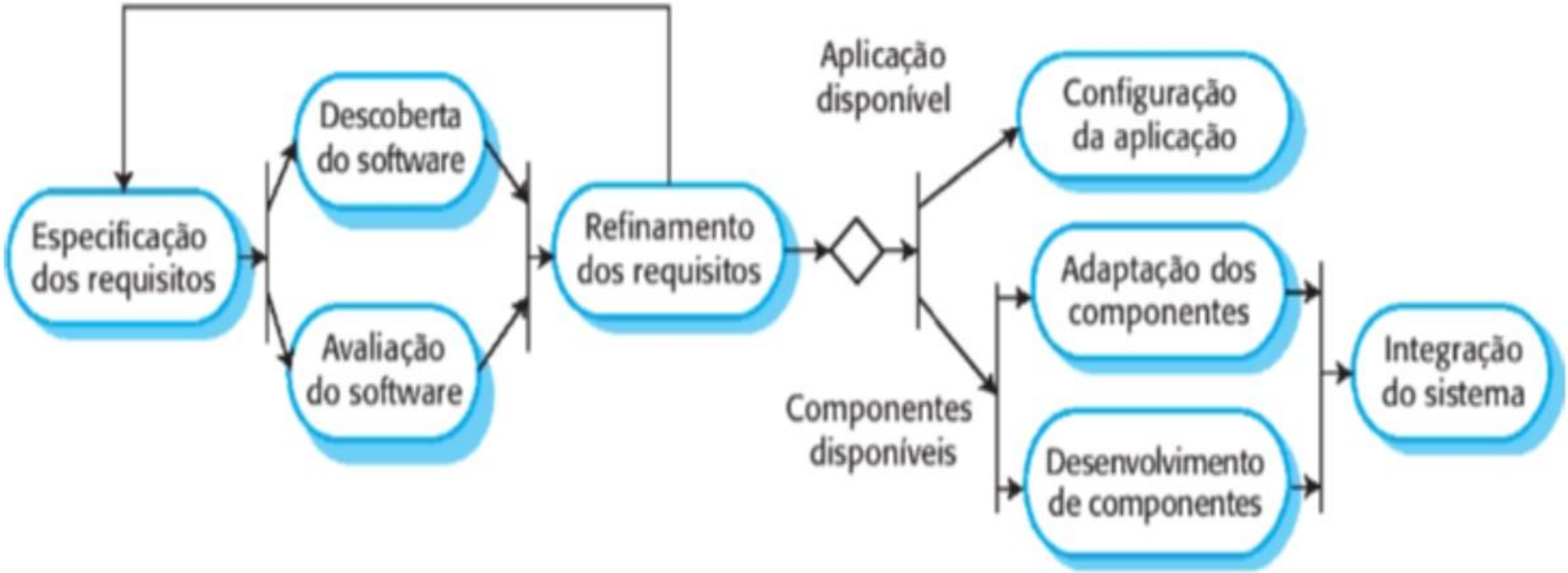
Modelo de Reutilização de Software (cont.)

- Três tipos de componentes de software são reutilizados frequentemente:
  - Sistemas de aplicação *stand alone* configurados para utilização em um ambiente particular. Esses sistemas são de uso geral e possuem muitas características, mas precisam ser adaptados para uso em uma aplicação específica.
  - Coleções de objetos desenvolvidos como um componente ou como um pacote a ser integrado a *frameworks* de componentes, como o *Java Spring Framework*.
  - *Web Services* desenvolvidos de acordo com os padrões de serviço e que estão disponíveis para uso remoto na Internet.

Modelo de Reutilização de Software (cont.)

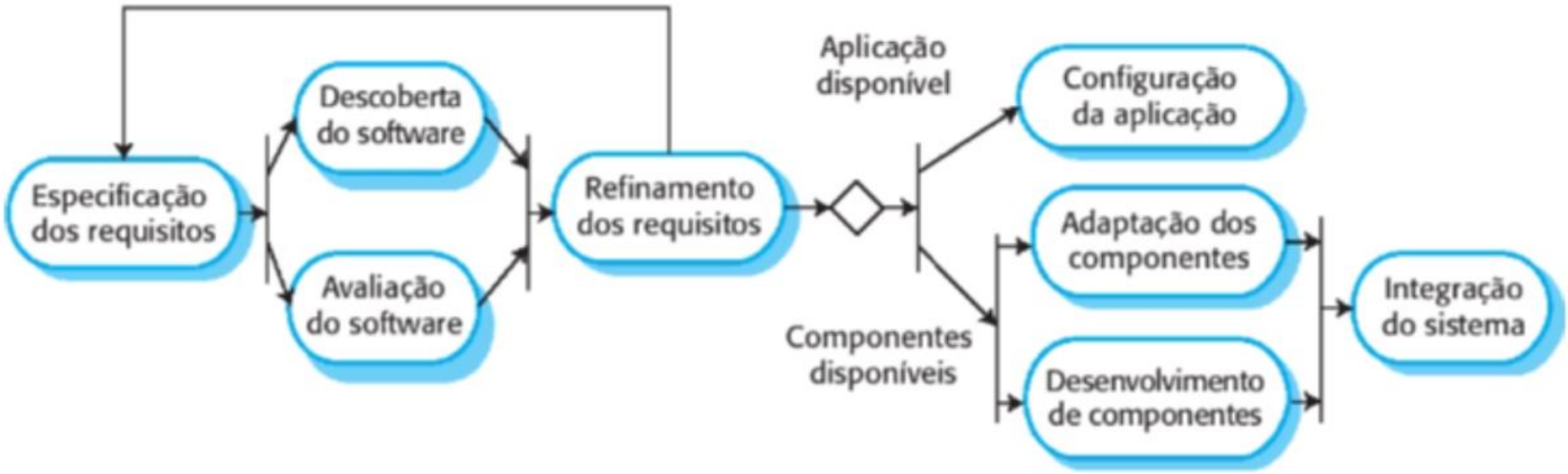
- Há um modelo de processo para a reutilização de software.

Processo da Engenharia de Software Orientada para Reutilização



Modelo de Reutilização de Software (cont.)

Processo da Engenharia de Software Orientada para Reutilização

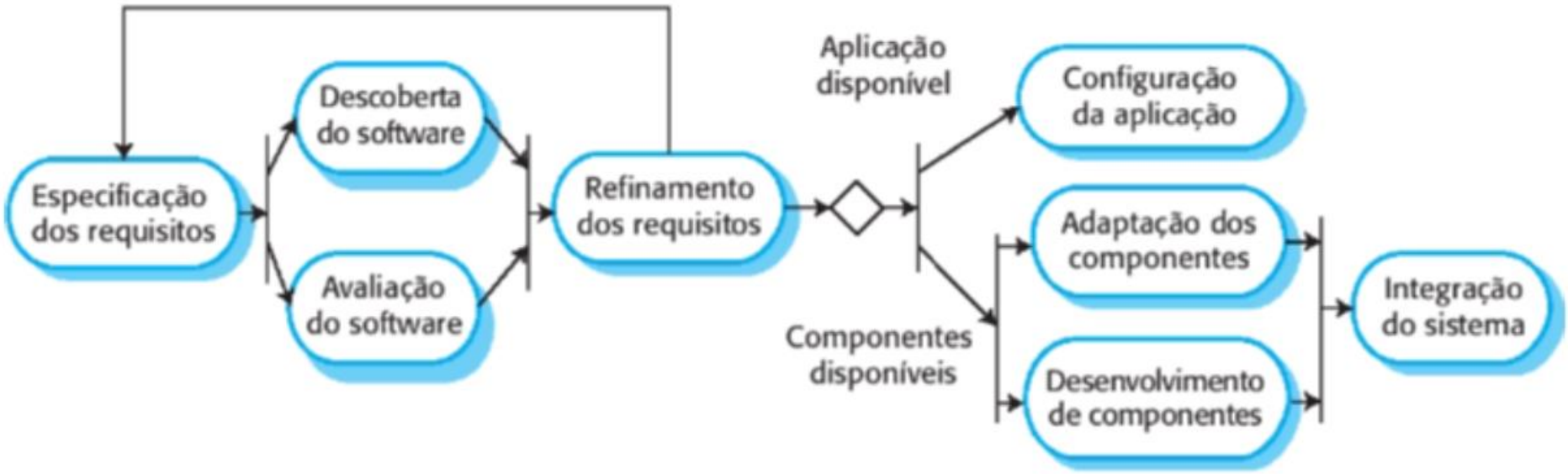


Especificação dos Requisitos

- Os requisitos iniciais do sistema são propostos.
- Eles não precisam ser elaborados em detalhes, mas devem incluir descrições breves dos requisitos essenciais e das características de sistema desejáveis.

Modelo de Reutilização de Software (cont.)

Processo da Engenharia de Software Orientada para Reutilização

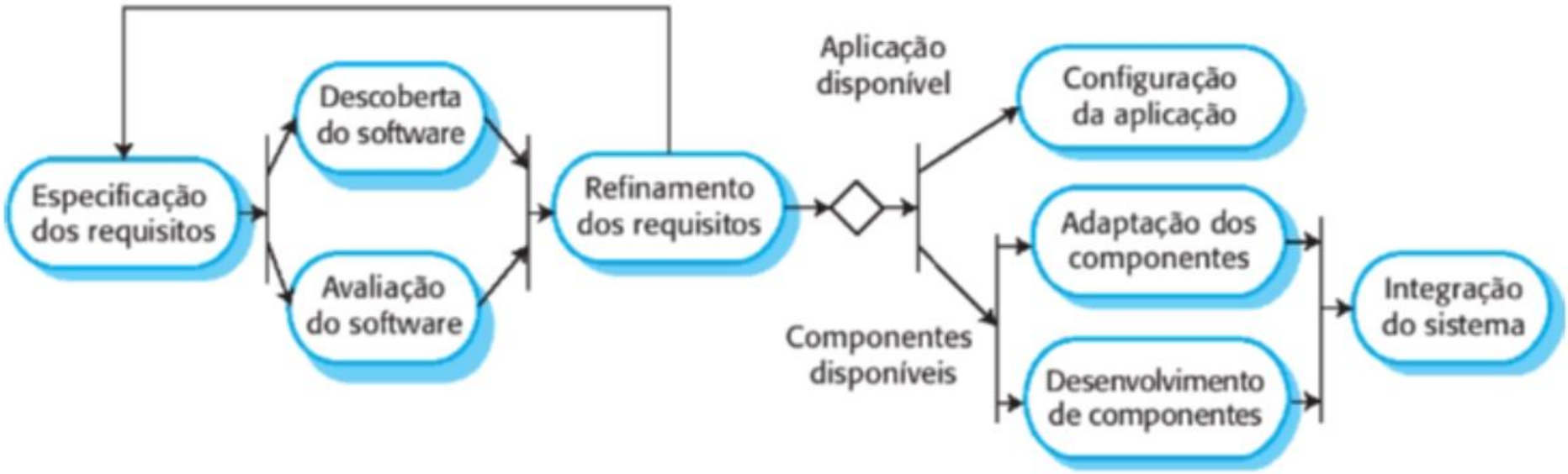


Descoberta e Avaliação do Software

- Com base em uma descrição dos requisitos de software, é feita uma busca pelos componentes e sistemas que fornecem a funcionalidade necessária.
- Os componentes candidatos são avaliados para ver se satisfazem os requisitos essenciais e se são genericamente adequados ao uso no sistema.

Modelo de Reutilização de Software (cont.)

Processo da Engenharia de Software Orientada para Reutilização

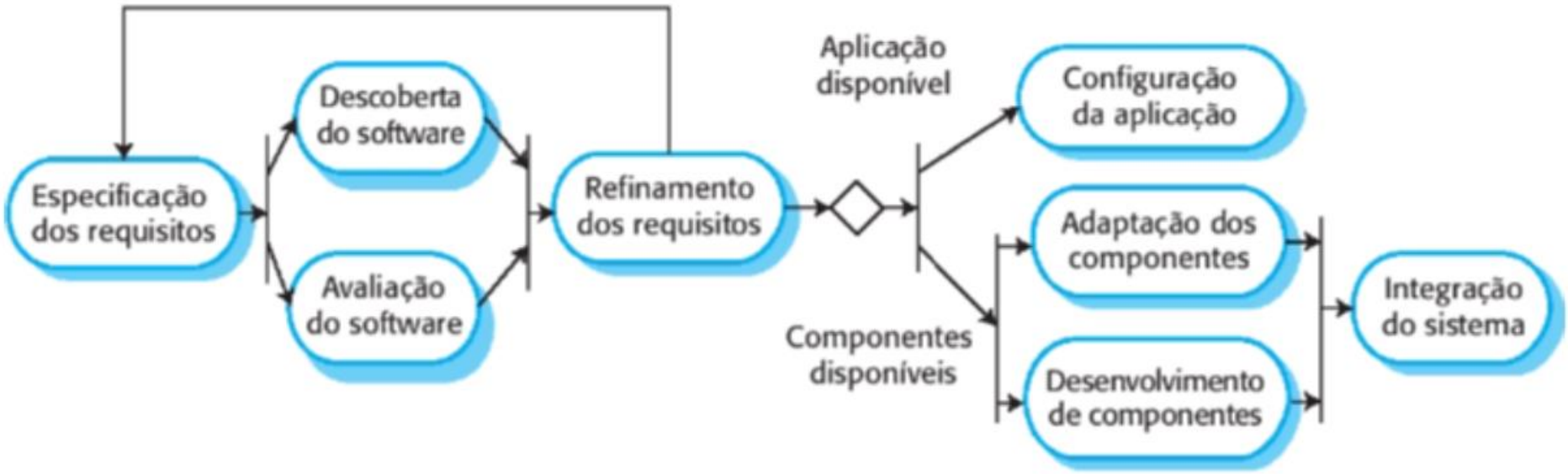


Refinamento dos Requisitos

- Nesse estágio, os requisitos são definidos com base nas informações dos componentes reusáveis e das aplicações que foram descobertas.
- Os requisitos são modificados para refletir os componentes disponíveis, e a especificação do sistema é redefinida.
- Quando modificações forem impossíveis, a atividade da análise de componentes pode ser reintroduzida para procurar soluções alternativas.

Modelo de Reutilização de Software (cont.)

Processo da Engenharia de Software Orientada para Reutilização

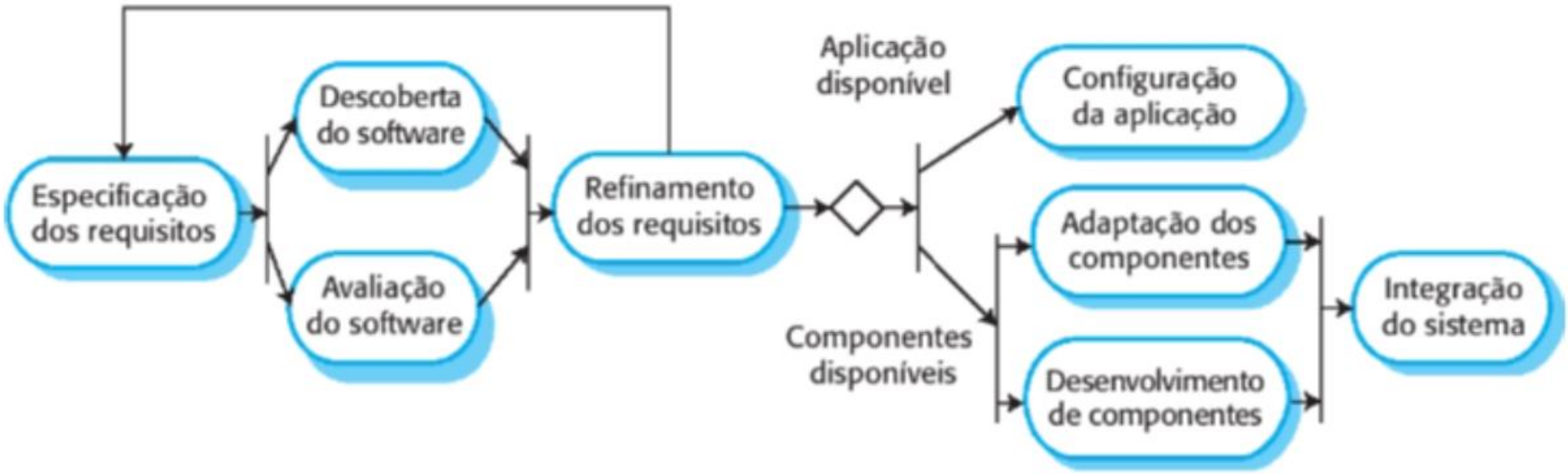


Configuração da Aplicação

- Se estiver disponível uma aplicação de prateleira que satisfaça os requisitos, ela pode ser configurada para utilização a fim de criar o novo sistema.

Modelo de Reutilização de Software (cont.)

Processo da Engenharia de Software Orientada para Reutilização

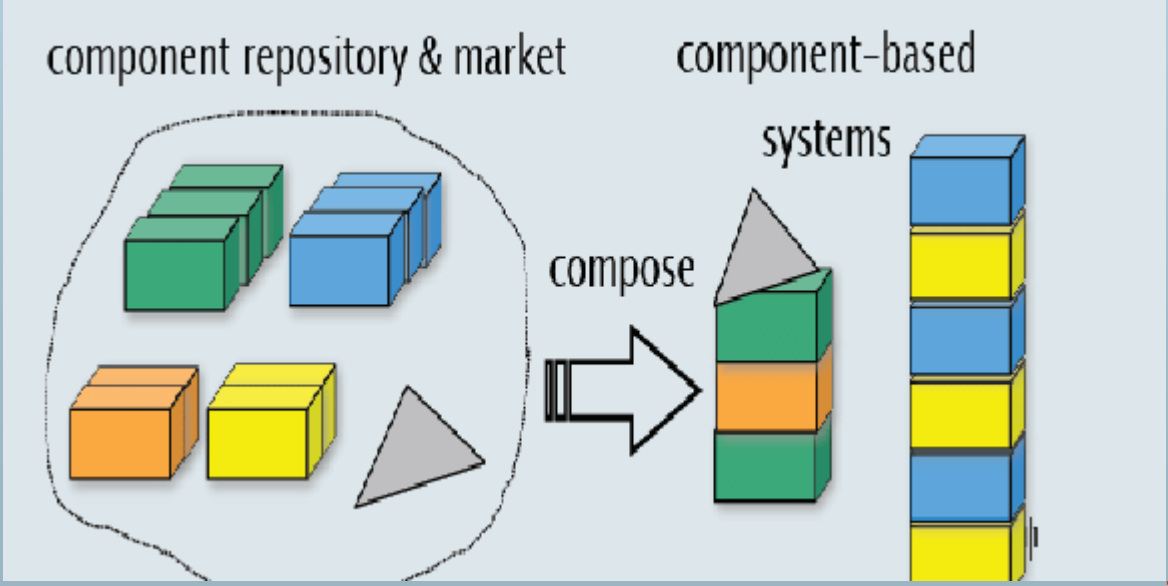


Adaptação, Desenvolvimento e Integração dos Componentes

- Se não houver uma aplicação de prateleira, componentes reusáveis podem ser modificados ou novos componentes podem ser desenvolvidos, visando a integração posterior ao sistema.

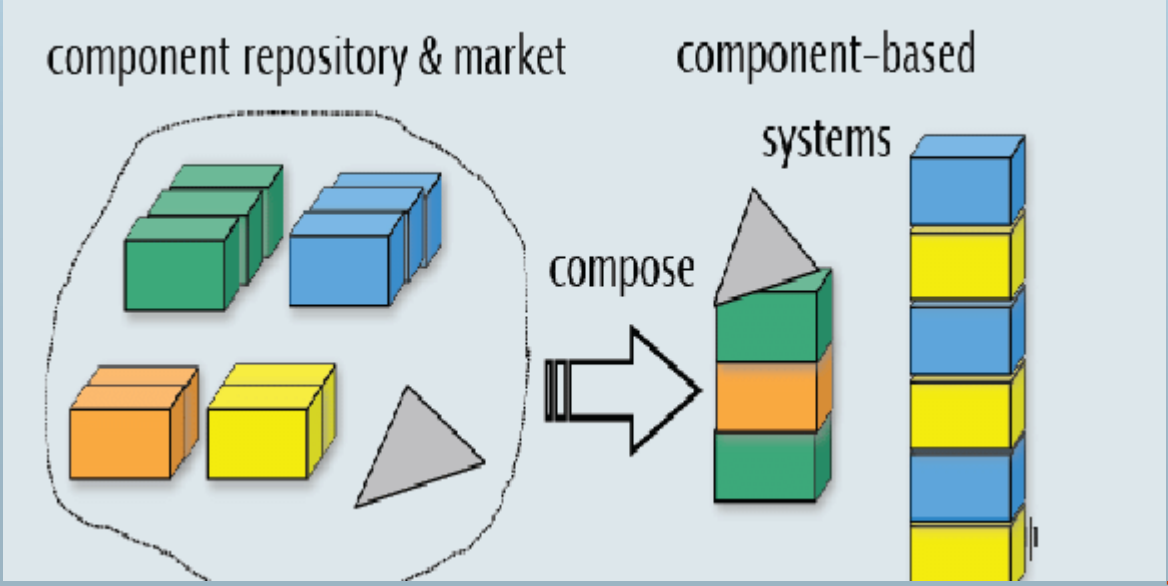
Modelo de Reutilização de Software (cont.)

- Vantagens.
  - A Engenharia de Software Orientada para a Reutilização, articulada em torno da configuração e da integração, tem a vantagem óbvia de reduzir a quantidade de software a ser desenvolvido, diminuindo custos e riscos.
  - Normalmente, isso também leva a uma entrega mais rápida do software.



Modelo de Reutilização de Software (cont.)

- Desvantagens.
  - Concessões quanto aos requisitos são inevitáveis, o que pode resultar em um sistema que não satisfaz as necessidades reais dos usuários.
  - Parte do controle sobre a evolução do sistema se perde, já que novas versões dos componentes reusáveis não estão sob o controle da organização que os utiliza.



Modelo de Reutilização de Software (cont.)

- Caso real.
  - Em 2016, uma disputa de direitos autorais motivou um desenvolvedor a remover uma de suas bibliotecas do diretório npm, muito usado para armazenar e distribuir bibliotecas node.js/JavaScript.
  - A biblioteca removida, chamada de leftPad, tinha uma única função JavaScript, de nome leftPad, com apenas 11 linhas de código.
  - Ela preenchia uma string com brancos à esquerda.
  - Por exemplo, leftPad('foo', 2) iria retornar ' foo', ou seja, 'foo' com dois brancos à esquerda.
  - Milhares de sistemas Web dependiam dessa função trivial, porém a dependência ocorria de modo indireto.

Modelo de Reutilização de Software (cont.)

- Os sistemas usavam o npm para baixar dinamicamente o código JavaScript de uma biblioteca B1, que por sua vez dependia de uma biblioteca B2 cujo código também estava no npm e, assim por diante, até alcançar uma e-nésima biblioteca Bn que dependia do left-pad.
- Como resultado, todos os sistemas que dependiam do left-pad, de forma direta ou indireta, ficaram fora do ar por algumas horas, até que a biblioteca fosse inserida de novo no npm.

- Em resumo, os sistemas foram afetados por um problema em uma biblioteca trivial.
  - E eles não tinham a menor ideia de que estavam acoplados a ela.

Modelo de Prototipação

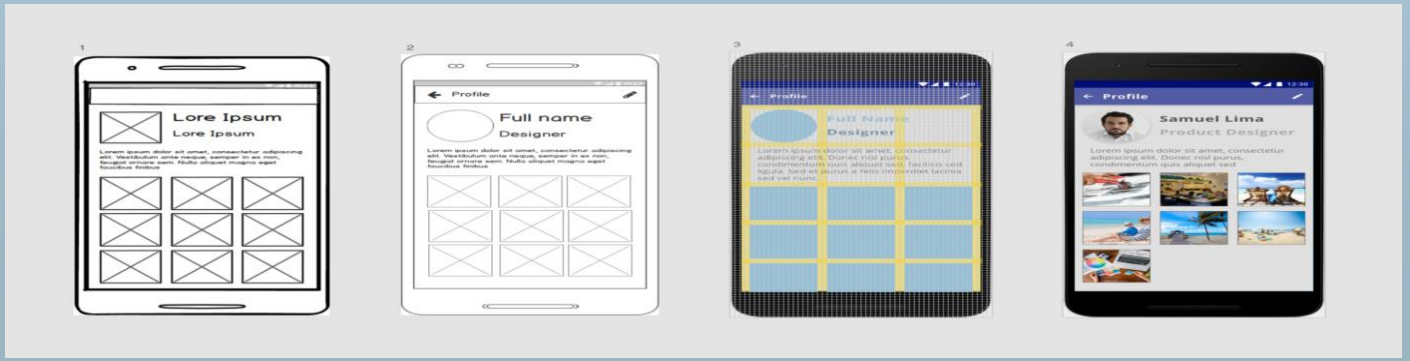
- O Modelo de Prototipação ou Prototipagem é a construção de um exemplar do que foi entendido dos requisitos capturados do cliente.
  - Pode ser considerado um ciclo de vida ou pode ser usado como ferramenta em outros ciclos de vida.

Modelo de Prototipação

- Um protótipo em Engenharia de Software pode ser o desenho de uma tela ou um software contendo algumas funcionalidades do sistema.
  - São considerados operacionais (quando já podem ser utilizados pelo cliente no ambiente real, ou seja, em produção), ou não operacionais (não estão aptos para serem utilizados em produção).
  - Os protótipos podem ser descartados, ou reaproveitados para evoluírem até a versão final.

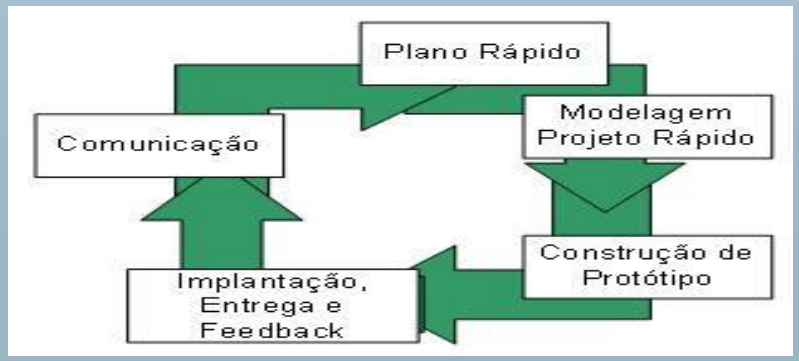
Modelo de Prototipação (cont.)

- Na Prototipagem, não é exigido um conhecimento aprofundado dos requisitos num primeiro momento.
  - Isso é bastante útil quando os requisitos não são totalmente conhecidos, são muitos complexos ou confusos.
  - Desta forma, se o cliente não sabe expressar o que deseja (o que ocorre bastante quando não é um sistema legado), a melhor maneira de evitar que se perca tempo e recursos com uma má interpretação é a construção de modelos, ou seja, de protótipos do que o software faria.



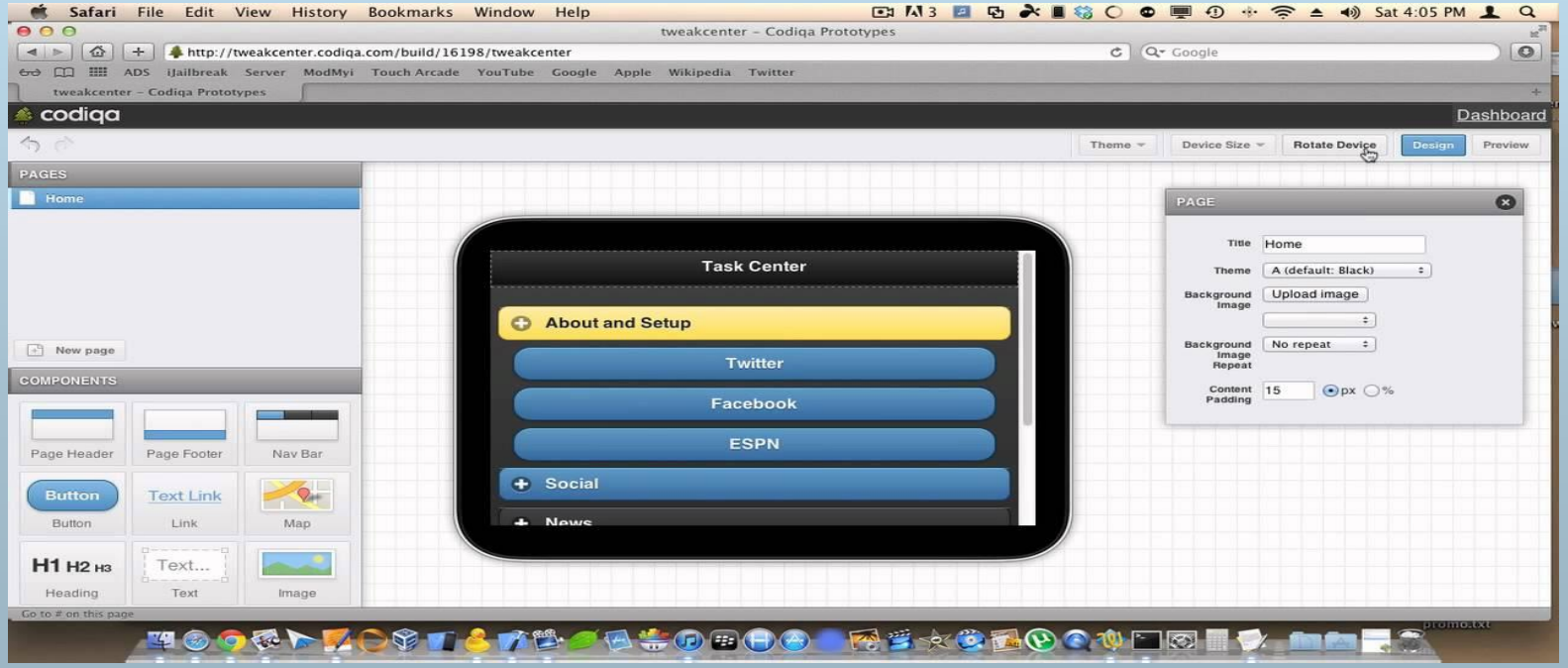
Modelo de Prototipação (cont.)

- Assim, com o protótipo o cliente experimentará, na prática, como o sistema ou parte dele funcionará.
  - A partir desse primeiro contato, o cliente esclarece o que não foi bem interpretado, aprofunda alguns conceitos e até descobre um pouco mais sobre o que realmente precisa.
  - A partir deste *feedback*, novos requisitos são colhidos e o projeto ganha maior profundidade.
  - Outro protótipo é gerado e apresentado ao cliente, que retorna com mais *feedbacks*.
  - O cliente participa ativamente do início ao fim do processo.



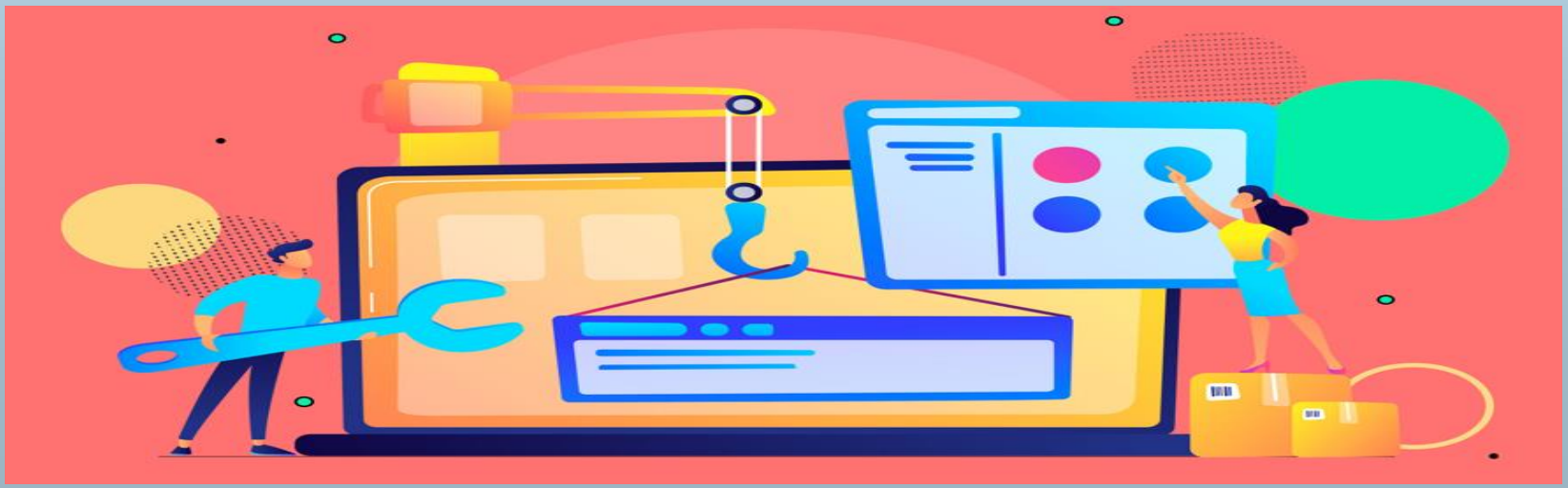
Modelo de Prototipação (cont.)

- A geração de protótipos pode ser facilitada por ferramentas geradoras de telas, de relatórios, poupando esforço de programação e diminuindo o tempo de entrega.



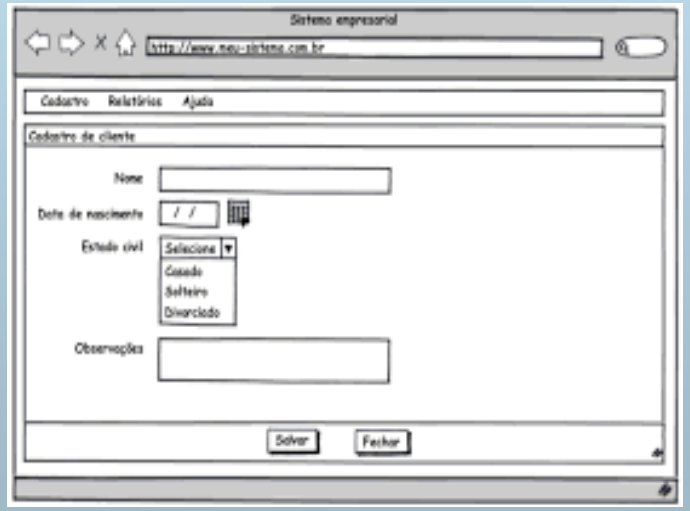
Modelo de Prototipação (cont.)

- Cada protótipo tem uma finalidade diferente.
  - Protótipos podem servir para esclarecer dúvidas sobre um processo ou rotina, demonstrar a aparência das telas, conteúdo de tabelas, formato de relatórios.
  - Protótipos podem ser utilizados para apresentar opções ao cliente para que ele escolha a que mais lhe agrade, como opções de navegação, de fluxo de telas, entre outras.



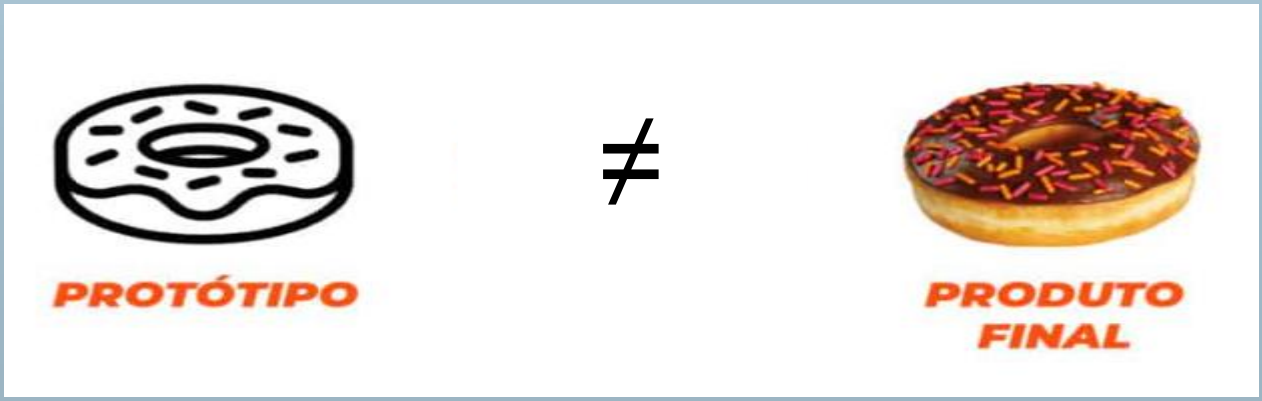
Modelo de Prototipação (cont.)

- É muito importante explicar previamente ao cliente que protótipos são apenas modelos para melhorar a comunicação.
  - Caso contrário, pode causar uma frustração por não funcionar corretamente, ter funções limitadas, ter resposta lenta, ou a aparência ruim.
- Certamente, um protótipo construído para esclarecer uma rotina provavelmente
  - Poderá ter uma “cara feia”, para demonstrar a aparência das telas.
  - Não terá funcionalidade, pois para apresentar o formato dos relatórios, os dados podem não ser coerentes.



Modelo de Prototipação (cont.)

- Deve-se destacar as diferenças entre um protótipo e o sistema final, pois o cliente poderá fazer comparações entre este último (sistema final) e o que foi “prometido” por meio do protótipo e pode ficar insatisfeito.
  - Por exemplo, geralmente o protótipo não acessa a rede ou banco de dados, pois as informações são “desenhadas” com a tela, fazendo com que tudo fique muito rápido.
  - Já no ambiente operacional haverá uma degradação de desempenho e o cliente pode se decepcionar.

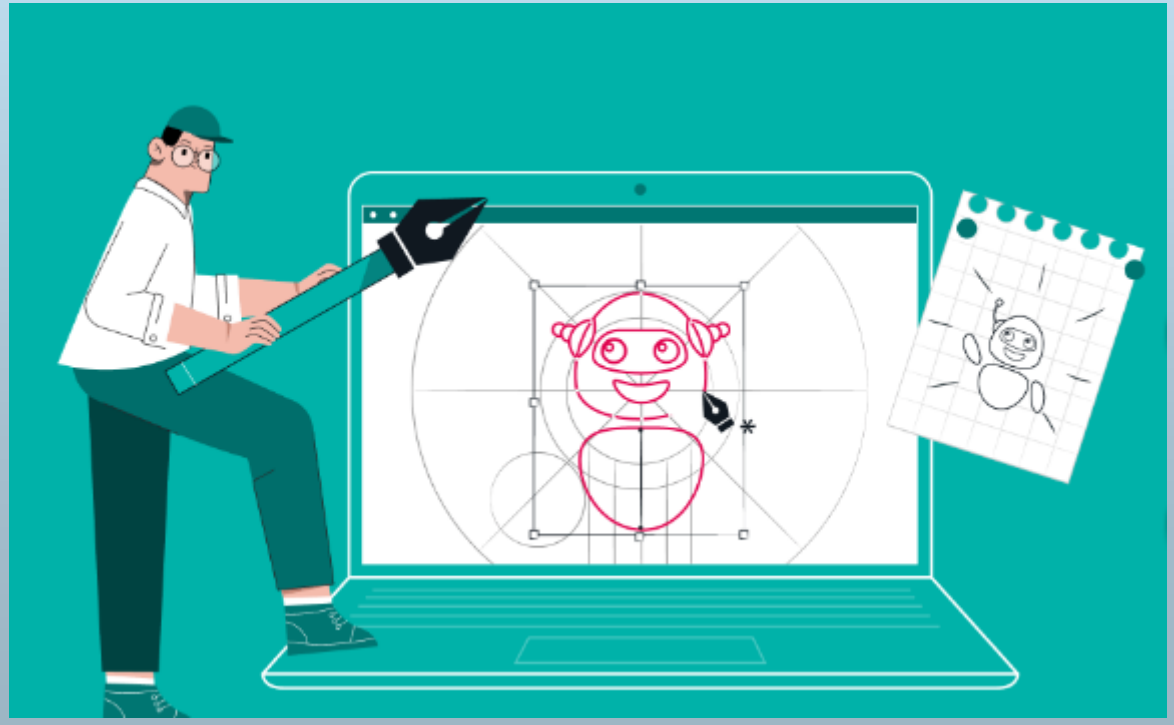


Modelo de Prototipação (cont.)

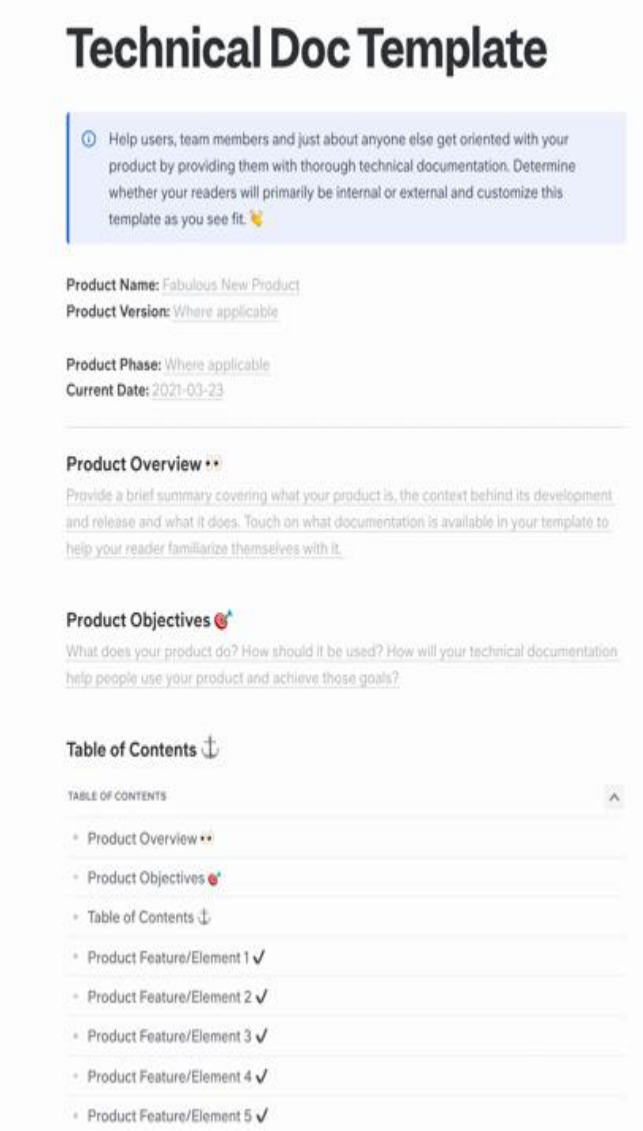
- Faz parte de um bom gerenciamento no Modelo de Prototipação planejar se, quais e que funções dos protótipos não operacionais serão reaproveitadas na versão operacional, para que sua confecção siga as boas práticas de Engenharia de Software.
  - Os protótipos não operacionais são construídos com pouca qualidade em prol da velocidade.
  - Ou seja, não há preocupação na programação
    - em refinar o código;
    - em usar comentários;
    - em aproveitar eficientemente os recursos de hardware e software;
    - na manutenção;
    - no reuso de componentes; e
    - na integração com outras funções ou sistemas.

Modelo de Prototipação (cont.)

- Com certeza será um problema se a equipe sucumbir à pressão do cliente, cada vez mais ansioso para ver a versão final daquele trabalho, e transformar protótipos não operacionais em operacionais.



Modelo de Prototipação (cont.)



Technical Doc Template

Help users, team members and just about anyone else get oriented with your product by providing them with thorough technical documentation. Determine whether your readers will primarily be internal or external and customize this template as you see fit.

Product Name: Fabulous New Product
Product Version: Where applicable

Product Phase: Where applicable
Current Date: 2021-03-23

Product Overview

Provide a brief summary covering what your product is, the context behind its development and release and what it does. Touch on what documentation is available in your template to help your reader familiarize themselves with it.

Product Objectives

What does your product do? How should it be used? How will your technical documentation help people use your product and achieve those goals?

Table of Contents

TABLE OF CONTENTS

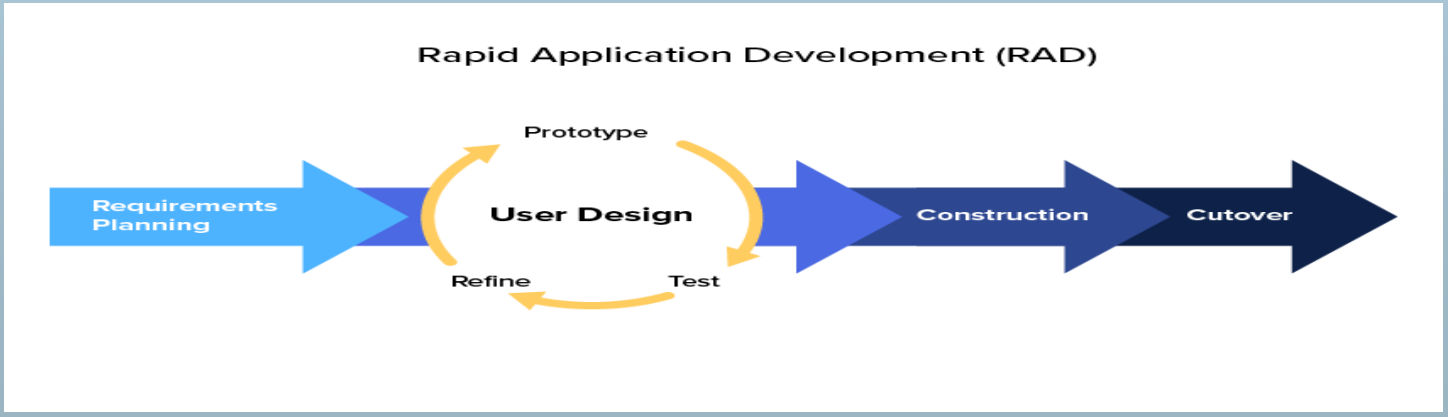
- Product Overview
- Product Objectives
- Table of Contents
- Product Feature/Element 1 ✓
- Product Feature/Element 2 ✓
- Product Feature/Element 3 ✓
- Product Feature/Element 4 ✓
- Product Feature/Element 5 ✓

O gerente também deve se preocupar com o escopo do projeto versus a quantidade de protótipos, para que não se perca muito tempo nesse processo, tampouco se transforme num processo de “tentativa e erro”.

- Não é uma tarefa fácil documentar o Modelo de Prototipação devido aos requisitos não serem totalmente conhecidos no primeiro momento e a consequente quantidade de mudanças ocorridas.

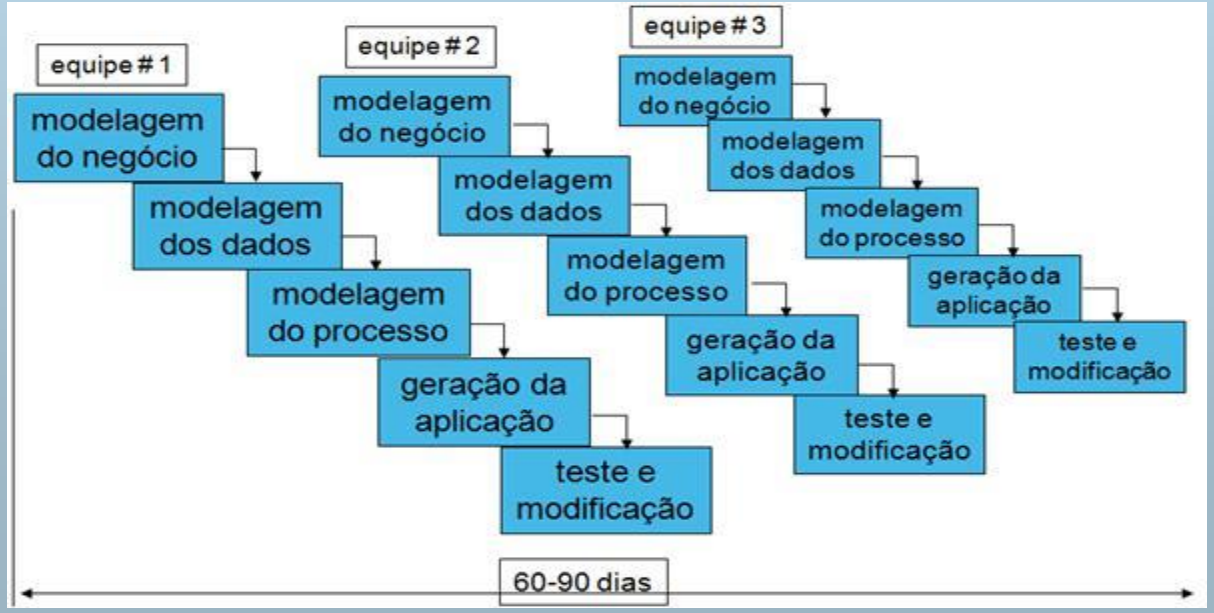
Modelo RAD

- O Modelo *Rapid Application Development* (RAD *Model*), foi formalizado por James Martin em 1991 como uma evolução da "prototipação", destacando-se pelo desenvolvimento rápido da aplicação.
  - O ciclo de vida é extremamente comprimido, de forma a encontrarem-se exemplos, na literatura, de duração de 60 e 90 dias.
  - É ideal para clientes buscando lançar soluções pioneiras no mercado.



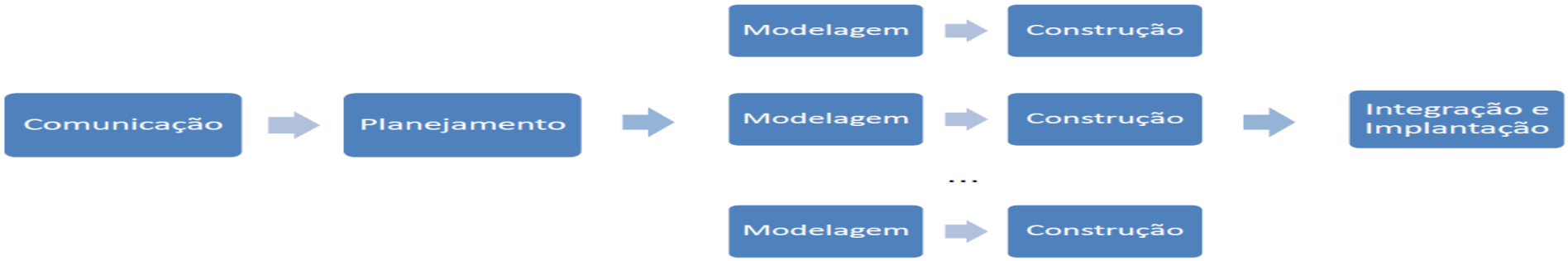
Modelo RAD (cont.)

- É um ciclo de vida incremental e iterativo, onde é preferível que os requisitos tenham escopo restrito.
  - Há um forte paralelismo das atividades, requerendo, assim, módulos bastante independentes.
  - Aqui os incrementos são desenvolvidos ao mesmo tempo, por equipes diferentes.



Modelo RAD (cont.)

- Além do paralelismo, a obtenção do curto tempo de desenvolvimento se dá graças à compressão da fase de requisitos e da fase de implantação.
  - Isso significa que, na obtenção dos requisitos, costumam-se optar por metodologias mais dinâmicas e rápidas, como *workshops* ao invés de entrevistas.
  - Permite-se também um desenvolvimento inicial no nível mais alto de abstração dos requisitos, em razão do envolvimento maior do usuário e da visibilidade mais cedo dos protótipos.



Modelo RAD (cont.)

- Há um forte utilização de:
  - Ferramentas de desenvolvimento.
  - Modelagem.
  - Componentes reutilizáveis (APIs, *templates*,...).
  - Utilização de atributos de manutenibilidade para facilitar a manutenção (ex.: linguagens de programação que suportem Orientação a Objetos, tratamento de exceção, ponteiros).
  - Utilização de ferramentas maduras e consolidadas, pois não há tempo de atualizar versões e tratar erros inesperados.

3.App Press

App Press is a web-based RAD tool with no code that can develop mobile applications for iPhone, iPad and Android devices. The target users of App Press are mainly designers, so the user interface they adopt is similar to Photoshop, which facilitates the assembly of various visual elements, layers and windows. The back end of App Press relies on the Amazon cloud platform, claiming to allow skilled designers to develop an app within a day.

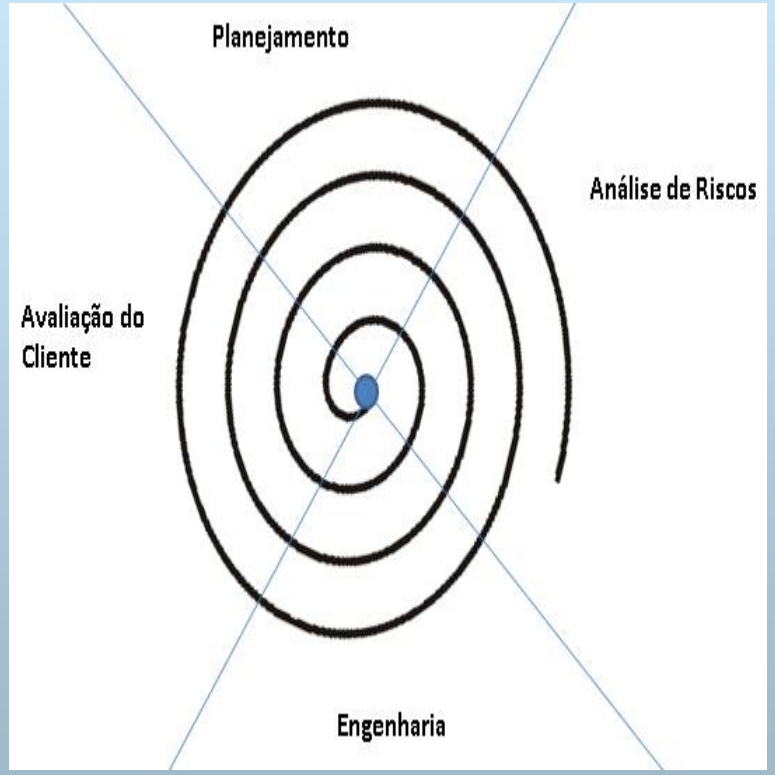
7.QuickBase

QuickBase is an online RAD tool for web data and mobile web database applications and provides limited support for the integration of external databases other than QuickBase. QuickBase has more than 800 customizable application templates, including project management, purchase order and sales management etc. Users can also use QuickBase to design a data structure and develop a brand new database application from scratch.

Modelo RAD (cont.)

- Os sistemas desenvolvidos no Modelo RAD tendem a ter uma padronização de telas muito forte, devido a bibliotecas reutilizáveis e *templates*, porém tendem a perder em desempenho do sistema e na análise de risco (atividades estas que demandam tempo em qualquer projeto).
  - Assim, é preferível seu uso para softwares de distribuição pequena.
- O Modelo RAD é difícil de ser utilizado em novos domínios de conhecimento ou domínios que são altamente instáveis.

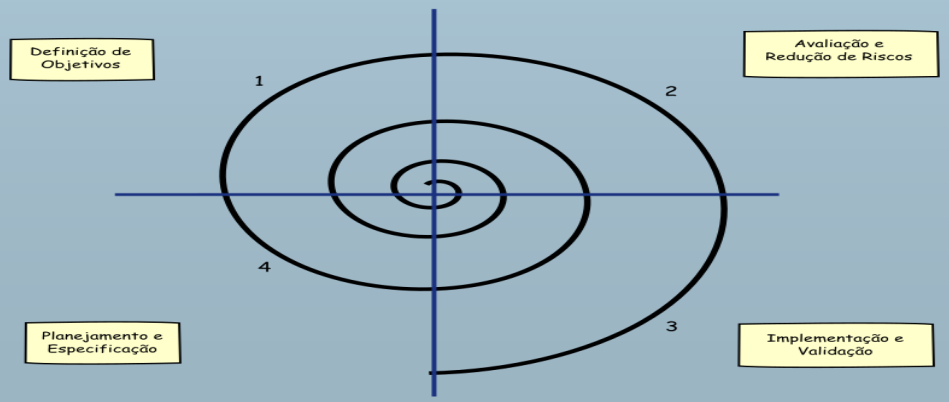
Modelo Espiral



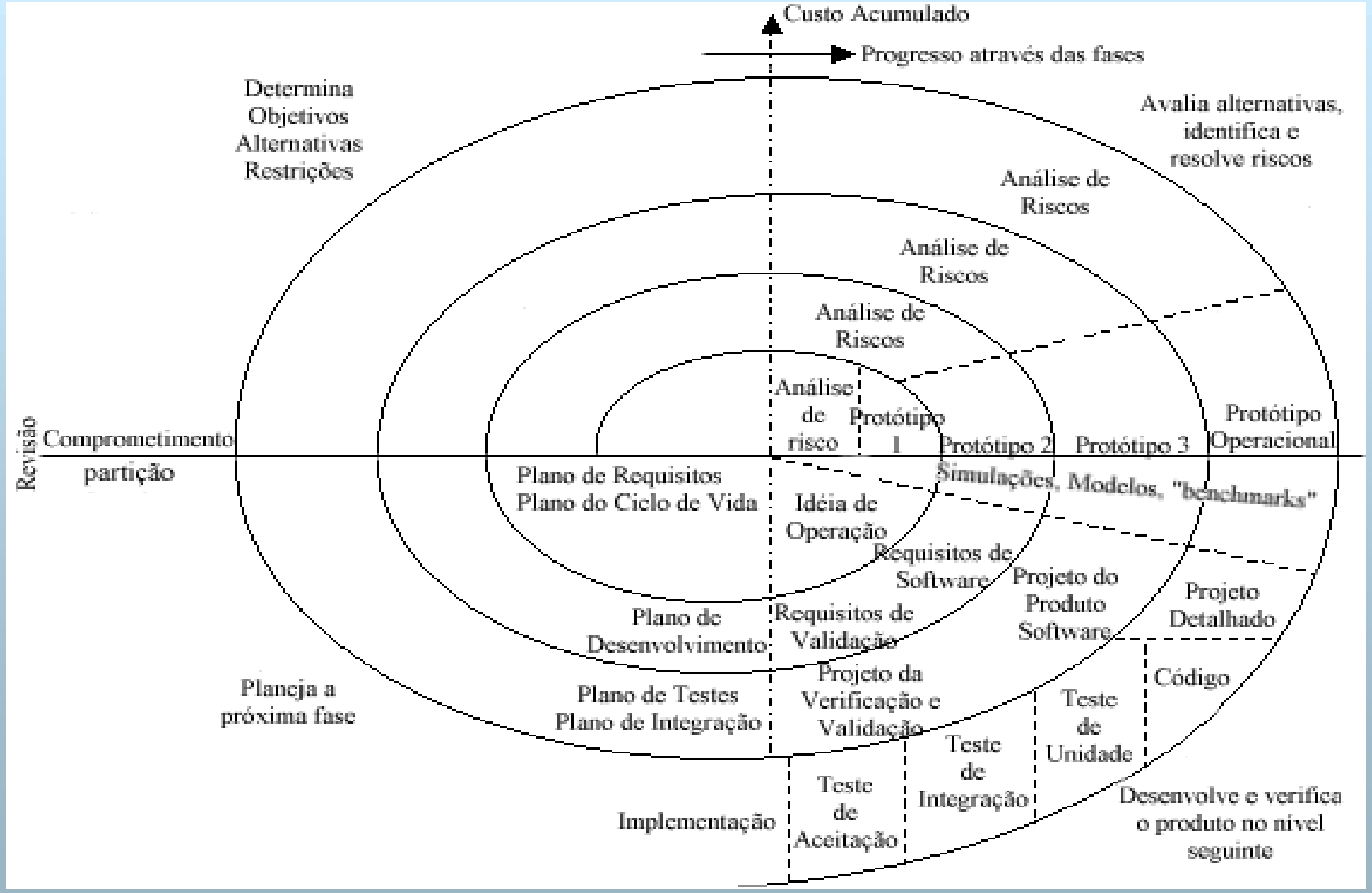
- O Modelo Espiral, proposto por Boehm em 1988, é uma abordagem cíclica das fases do processo, onde a cada “volta” ou iteração tem-se versões evolucionárias do sistema.
- É um modelo guiado por risco, suporta sistemas complexos e/ou de grande porte, onde falhas não são toleráveis.
  - A cada iteração há uma atividade dedicada à análise de riscos e apoiada através de geração de protótipos, não necessariamente operacionais (desenhos de tela, por exemplo) para que haja um envolvimento constante do cliente nas decisões.

Modelo Espiral (cont.)

- Cada iteração ou volta é dedicada a uma fase do processo de vida de um software (viabilidade do projeto, definição de requisitos, desenvolvimento e teste,...).
- Ao mesmo tempo, cada volta é seccionada em 4 setores.
  - Definição de Objetivos.
  - Análise de Riscos.
  - Desenvolvimento e Validação.
  - Planejamento da Próxima Fase.

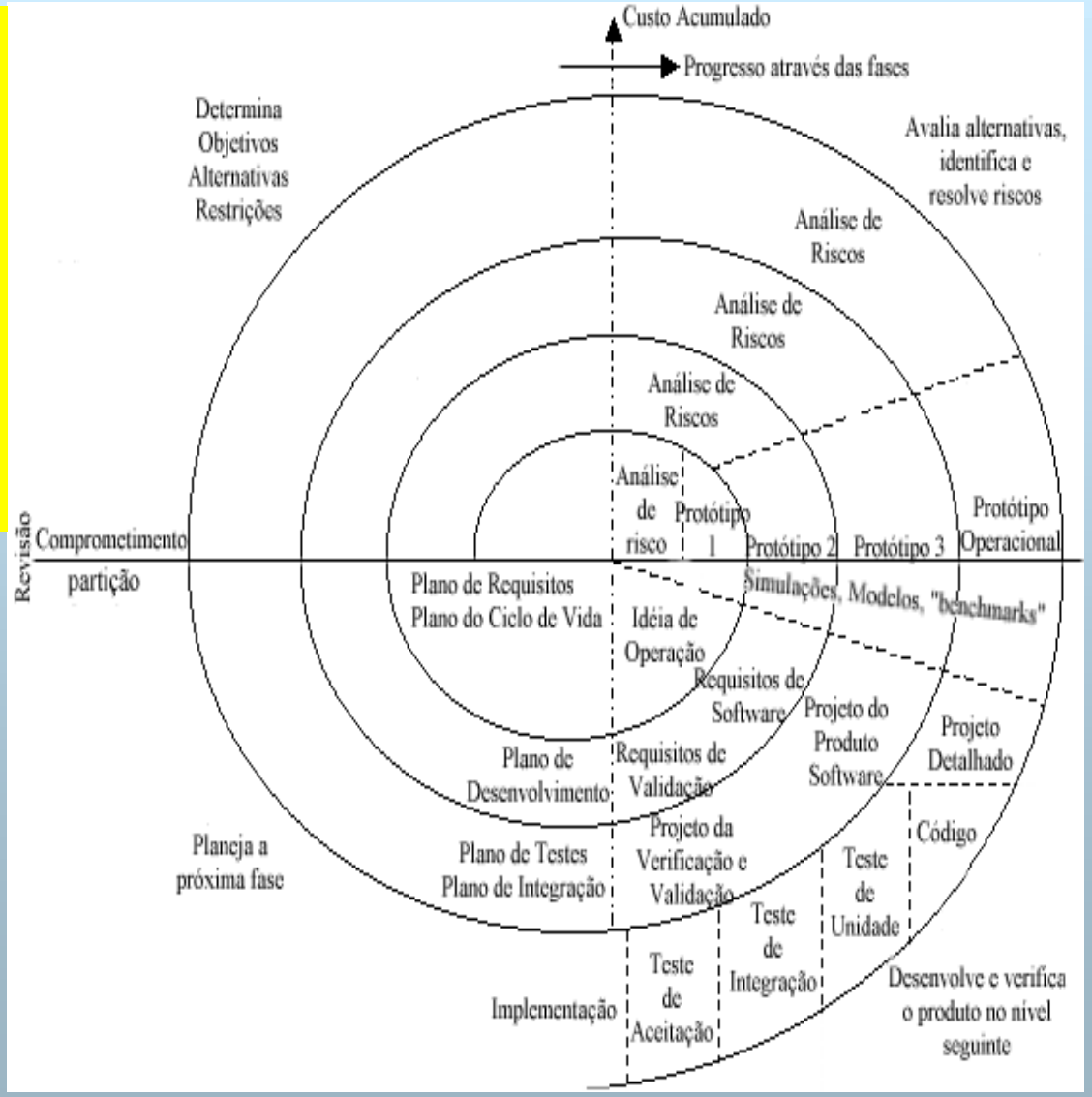


Modelo Espiral (cont.)

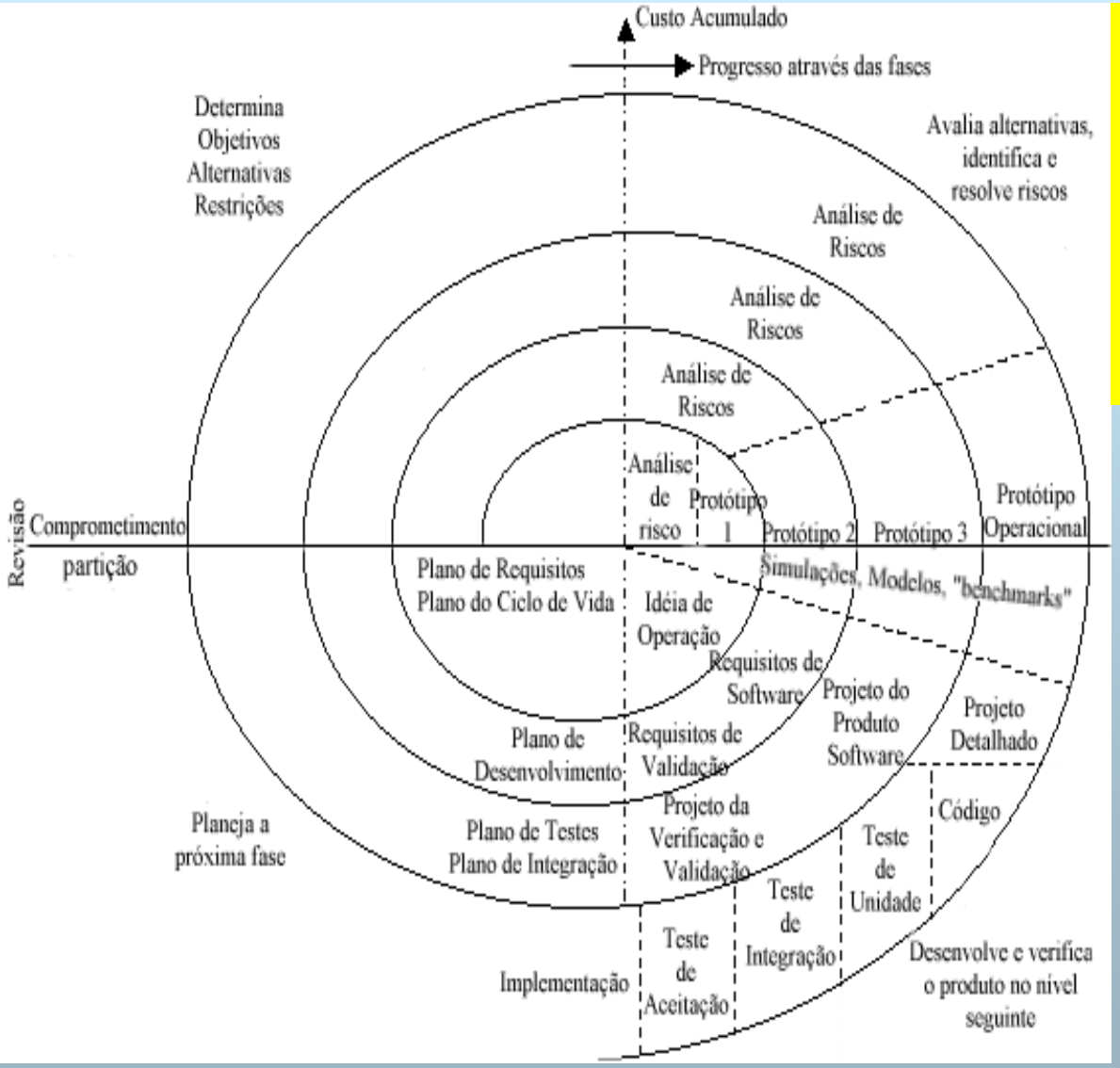


Modelo Espiral (cont.)

Na Definição de Objetivos, desempenhos, funcionalidade, entre outros objetivos, são levantados.
Visando alcançar esses objetivos são listadas alternativas e restrições, e cria-se um plano gerencial detalhado.

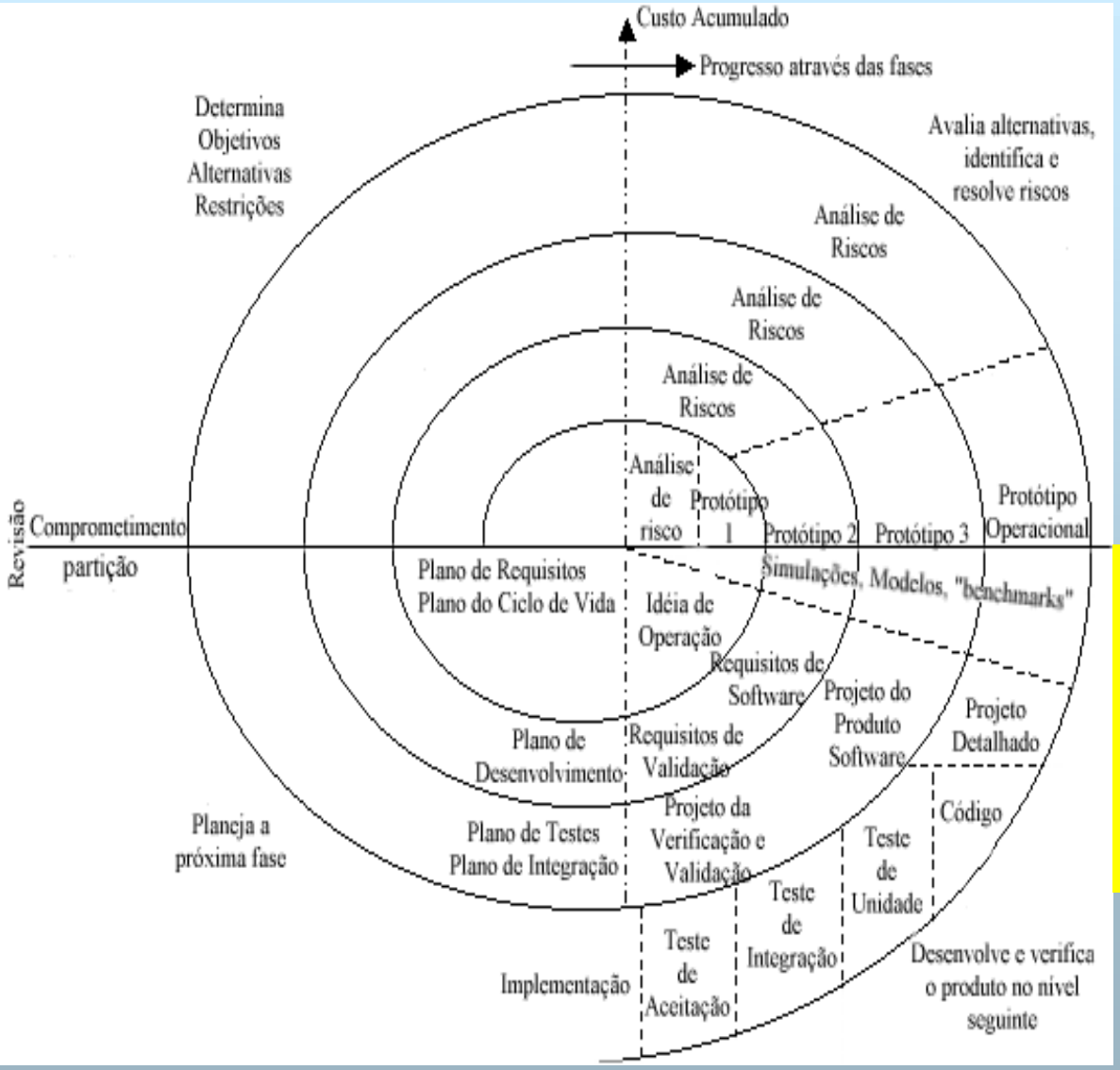


Modelo Espiral (cont.)



Na Análise de Riscos, as alternativas, restrições e riscos anteriormente levantados são avaliados.
Neste setor (porém não apenas neste) protótipos são utilizados para ajudar na análise de riscos.

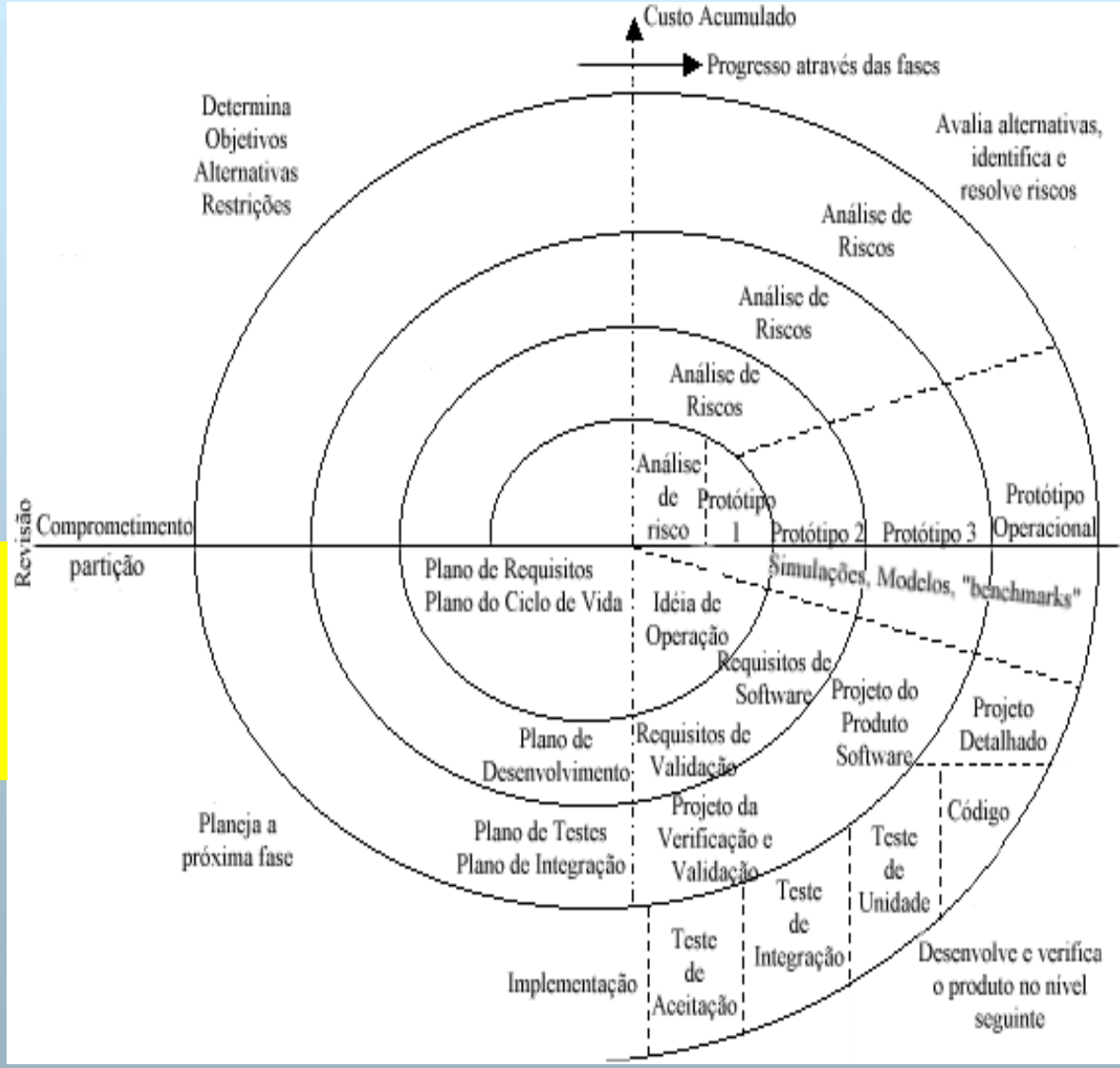
Modelo Espiral (cont.)



No Desenvolvimento e Validação um modelo apropriado para o desenvolvimento do sistema é escolhido, de acordo com o risco analisado no setor anterior (cascata, interativo,...).

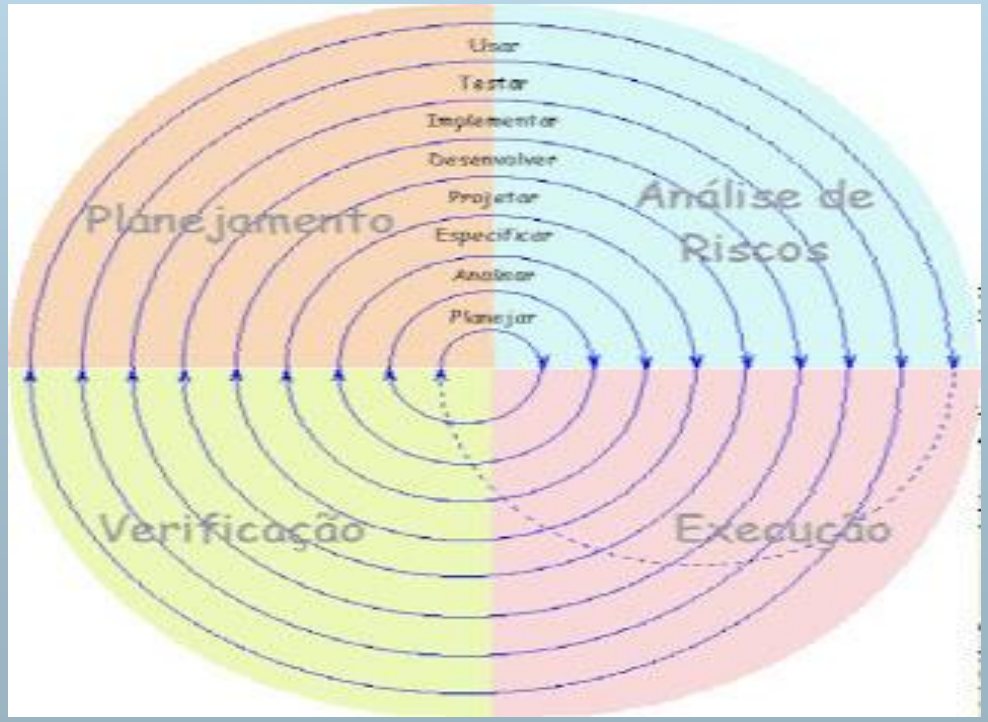
Modelo Espiral (cont.)

No Planejamento da Próxima Fase ocorre a revisão do projeto e a decisão de partir para a próxima fase.



Modelo Espiral (cont.)

- Neste modelo, apenas o início é definido.
  - A evolução e amadurecimento dos requisitos demandam tempo ajustável (assim como custo).
  - Isto torna o sistema difícil de ser vender ao cliente e exige um alto nível de gerenciamento em todo o processo.



Modelo Espiral (cont.)

- Foco principal do modelo é no gerenciamento de riscos.
- A cada ciclo do modelo:
  - O conhecimento aumenta;
  - O planejamento é refinado; e
  - O produto gerado no ciclo anterior é evoluído (não é jogado fora).

- Cada ciclo evolui o sistema, mas não necessariamente entrega um software operacional.
  - Modelo em papel.
  - Protótipo.
  - Versões do produto.
  - Etc.