

Engenharia de Software I

Aula – 4



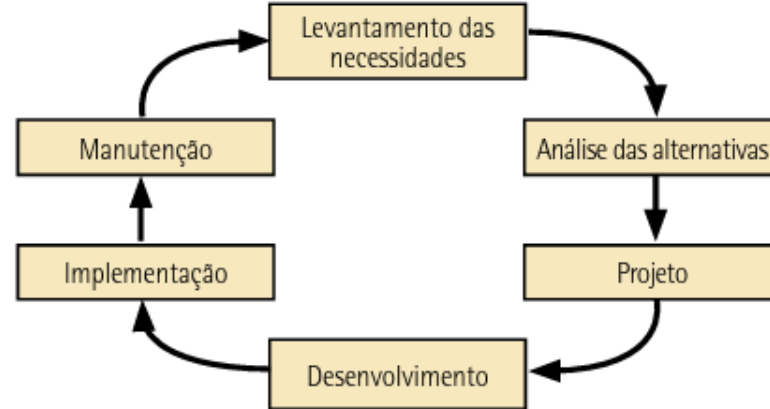
Prof. Salatiel Luz Marinho



salatiel.marinho@docente.unip.br


MODELOS DE CICLO DE VIDA DE SOFTWARE

De acordo com Gordon e Gordon (2006), o ciclo de vida do desenvolvimento de sistemas (Systems Development Life Cycle – SDLC), conhecido também como ciclo de vida do software, refere-se aos estágios de concepção, projeto, criação e implementação de um Sistema de Informação (SI). Um desdobramento possível para SDLC é mostrado na figura a seguir.




MODELOS DE CICLO DE VIDA DE SOFTWARE


Analisando-se as atividades propostas para o ciclo SDLC, com base em Gordon e Gordon (2006), tem-se o exposto a seguir.

- Levantamento de necessidades:
 - essa atividade compreende diversas tarefas para o entendimento e o levantamento das necessidades da área de negócio, dos clientes ou dos usuários que precisam de um software de automação de suas atividades;
 - tem como resultados mais importantes a lista de requisitos do sistema a ser construído, as informações que serão tratadas e armazenadas e, se necessário, um protótipo das interfaces entre os usuários e o sistema de software.
- 


MODELOS DE CICLO DE VIDA DE SOFTWARE

- Análise de alternativas:
 - essa atividade consiste na identificação e na avaliação de alternativas sistêmicas que melhor atendam aos requisitos do software a ser construído;
 - seria interessante, para que essa atividade fosse padronizada e a organização possuísse uma arquitetura de referência, que a base fosse para todas as soluções de tecnologia de todos os sistemas da empresa.
 - Projeto:
 - trata da construção das especificações detalhadas para o projeto selecionado;
 - essas especificações incluem projeto das interfaces, banco de dados, características físicas do sistema, tais como número, tipos e localizações das estações de trabalho, hardware de processamento, cabeamento e os dispositivos de rede; deve especificar os procedimentos para testar o sistema completo antes da instalação.
- 

MODELOS DE CICLO DE VIDA DE SOFTWARE

- Desenvolvimento:
 - inclui o desenvolvimento ou a aquisição do software, a provável aquisição do hardware e o teste do novo sistema.
 - Implementação:
 - ocorre após o sistema ter passado satisfatoriamente por testes de aceitação. O sistema é transferido do ambiente de desenvolvimento para o ambiente de produção;
 - o sistema antigo (se existir) deve migrar para o novo.
 - Manutenção:
 - refere-se a todas as atividades relacionadas a um sistema depois que ele é implementado, devendo incluir atividades como a correção de software que não funcione corretamente, a adição de novos recursos aos sistemas em resposta às novas demandas dos usuários etc.
- 


O Modelo Codifica-remenda

- O modelo codifica-remenda significa um processo de ciclo de vida de software mais caótico. Partindo apenas de uma especificação ou simplesmente de uma reunião de trabalho, os desenvolvedores começam imediatamente a codificar, remendando, à medida que os erros vão sendo descobertos. Não existe nenhum processo definido e nenhum é seguido.
 - Esse modelo, provavelmente e infelizmente, seja o ciclo de vida mais usado na atualidade. Para alguns desenvolvedores, ele é atraente porque não exige nenhuma sofisticação técnica ou gerencial. Todavia, é considerado um modelo de alto risco, impossível de ser gerenciado e que não permite assumir compromissos confiáveis.
 - Esse modelo persistiu durante algumas décadas, e, como não existia separação de papéis, todos faziam um pouco de tudo. Às vezes, uma única pessoa analisava, codificava e testava um sistema inteiro.
- 

O Modelo Codifica-remenda

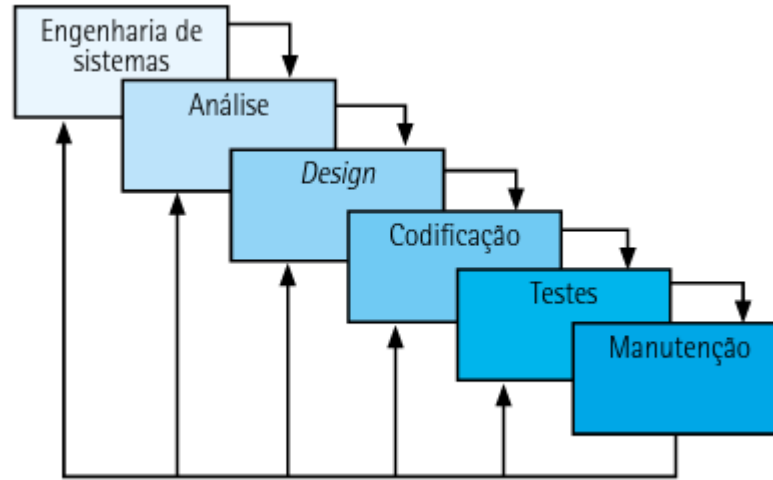
- Na busca de eliminar os problemas causados pelo método codifica-remenda, foram sendo criadas as metodologias conhecidas hoje como tradicionais, sendo a mais conhecida a chamada cascata, que é, também, referida como o modelo clássico de desenvolvimento de software.

O Modelo Cascata


- O modelo clássico ou cascata, que também é conhecido por abordagem top-down, surgiu na década de 1970. Até meados da década de 1980, foi o único modelo com aceitação geral; derivado de modelos de atividade de engenharia com o fim de estabelecer ordem no desenvolvimento de grandes produtos de software. Comparado com outros modelos, este é mais rígido e menos administrativo. Um dos mais importantes, é referência para muitos outros, servindo de base para projetos modernos. A versão original desse modelo foi melhorada e retocada ao longo do tempo, e este continua a ser muito utilizado hoje em dia.
 - Grande parte do sucesso do modelo cascata está no fato de ser orientado para a documentação. No entanto, esta abrange mais do que arquivo e texto: incluindo representações gráficas e mesmo simulações. Uma abordagem que incorpora processos, métodos e ferramentas deve ser utilizada pelos criadores de software.
- 

O Modelo Cascata

- O paradigma do ciclo de vida clássico demanda uma abordagem sistemática e sequencial para o desenvolvimento de software. Começa no nível do sistema e progride, através da análise, do design, da codificação, do teste e da manutenção, como apresentado na figura a seguir.

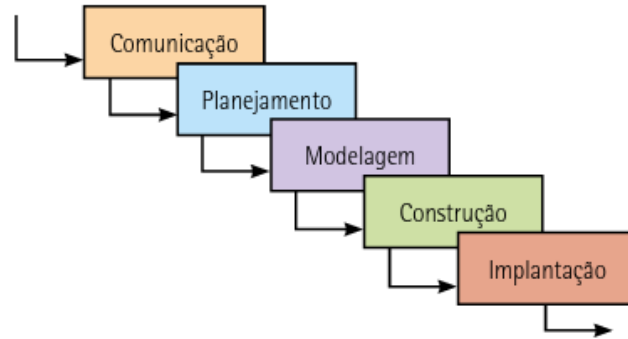


O Modelo Cascata


- Problemas encontrados no ciclo clássico de desenvolvimento de software são fartamente documentados pelos autores consagrados da engenharia de software:
 - Os projetos reais raramente seguem o fluxo sequencial que o modelo propõe; ocorrem interações, bem como voltas a níveis anteriores, provocando problemas na aplicação do paradigma;
 - Frequentemente os usuários têm dificuldade de estabelecer explicitamente todos os requisitos do software, acompanhada das incertezas naturais que existem no início de muitos projetos;
 - Uma versão do software somente estará disponível quando todo o sistema for definido e desenvolvido; qualquer alteração pode ocasionar um desastre no desenvolvimento do sistema; isso requer paciência dos usuários.
- 

O Modelo Incremental

- De acordo com Pressman (2006), Sommerville (2007), Paula Filho (2003) e outros autores, o modelo incremental seria a aplicação do modelo cascata por diversas vezes em um mesmo projeto. Isso ocorre ao se dividir o desenvolvimento de um sistema complexo em pequenas partes, o que pode ocorrer de forma sequencial, parte a parte ou em paralelo, com equipes diferentes desenvolvendo partes diferentes. Para apresentar o modelo, é utilizada uma visão do ciclo de vida em cascata considerada por Peres, Laskoski e Radaelli (2014), mostrada na próxima figura.

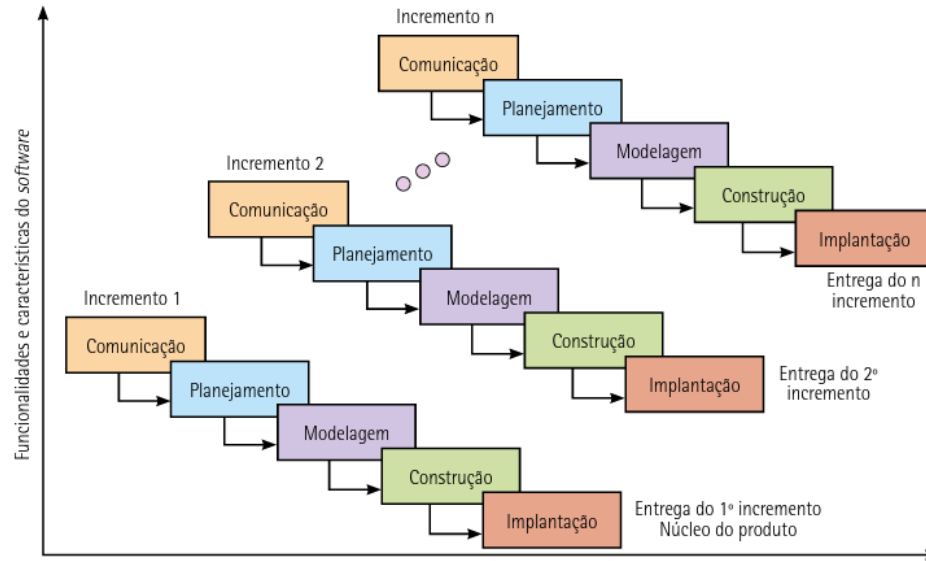


O Modelo Incremental


- As fases desse modelo, de acordo com Peres, Lakosky e Radaelli (2014, p. 2) significam:
 - **Comunicação:** consiste de atividades para o levantamento dos requisitos do sistema e envolve a comunicação entre os desenvolvedores e o cliente;
 - **Planejamento:** consiste de atividades para o estabelecimento de um plano de desenvolvimento do sistema ou projeto de software. Descreve, também, as técnicas a serem conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos de trabalho a serem produzidos e um cronograma;
 - **Modelagem:** composta de atividades que incluem a criação de modelos que permitem ao desenvolvedor e ao cliente entender melhor os requisitos do software e o projeto que vai satisfazer a esses requisitos;
 - **Construção:** essa fase combina a geração de código e os testes necessários para revelar erros no código implementado;
 - **Implantação:** o software é entregue ao cliente, que avalia o produto e fornece feedback com base na avaliação.
- 

O Modelo Incremental


- Se essas fases se repetirem e pequenos pedaços do software forem sendo liberados ou agrupados para serem entregues ao cliente, teremos o que se chama de modelo incremental, mostrado na figura a seguir.




O Modelo Incremental

- Vantagens apontadas pelos autores para o modelo incremental em relação ao modelo em cascata puro:
 - entregas parciais facilitam a identificação e a correção de erros entre os componentes do software; necessidades não especificadas nas fases iniciais podem ser desenvolvidas nos incrementos;
 - cada iteração produz um conjunto de itens utilizáveis (se possível);
 - os feedbacks de iterações anteriores podem ser usados nos próximos incrementos;
 - os incrementos podem ser desenvolvidos por menos profissionais; • a entrega dos incrementos permite o cumprimento do prazo especificado.
- 


O Modelo Incremental

- Como todo modelo conceitual, porém, o modelo incremental apresenta algumas desvantagens:
 - o número de iterações não pode ser definido no início do processo, o que pode não ser aceito pelo cliente;
 - o fim do processo também não pode ser previamente definido;
 - o gerenciamento e a manutenção do sistema completo podem se tornar complexos, principalmente, se ocorrem durante o desenvolvimento dos incrementos;
 - o gerenciamento do custo do projeto é mais complexo, em razão do número de iterações que pode acabar com a verba disponível.
- 

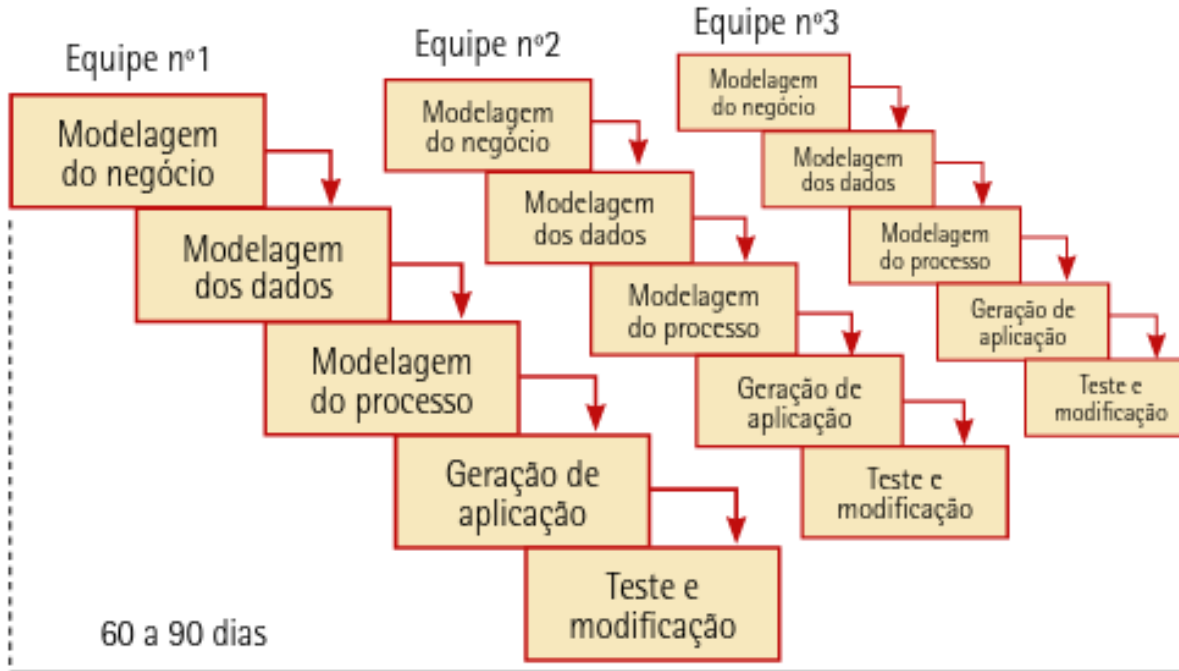
O Modelo Rapid Application Development (RAID)

- Na busca de produtividade e qualidade no processo de desenvolvimento de sistemas, especialistas e autores da engenharia de software, baseados nos problemas do ciclo clássico (cascata), na evolução das técnicas estruturadas, no impacto das linguagens de programação orientadas a objetos e no surgimento dos conceitos de qualidade, no final da década de 1980 e início da década de 1990, propuseram um novo paradigma denominado de Rapid Application Development (RAD).
 - A metodologia RAD propõe um conjunto de elementos que criaram um novo paradigma incluindo a prototipação interativa, o desenvolvimento espiral e o uso intensivo de ferramentas de automação do desenvolvimento (CASE), com uso de linguagens de quarta geração, orientação a objetos e modelos padrão de desenvolvimento, como a Unified Modeling Language (UML).
- 


O Modelo Rapid Application Development (RAID)

- O RAD foi originalmente utilizado por James Martin, em seu livro RAD, publicado em 1991. De acordo com Ramos (2009), tem-se sobre o modelo RAD:
 - É sequencial linear e enfatiza um ciclo de desenvolvimento extremamente curto;
 - O desenvolvimento rápido é obtido usando uma abordagem de construção baseada em componentes;
 - Os requisitos devem ser entendidos corretamente, e o alcance do projeto deve ser restrito;
 - É usado principalmente para aplicações de sistemas de informação;
 - Cada função principal pode ser direcionada para uma equipe RAD separada e, então, integrada para formar o todo.
- 

O Modelo Rapid Application Development (RAID)



O Modelo Rapid Application Development (RAID)

- Alguns autores consideram que nem todos os tipos de aplicação são apropriados para o modelo RAD. Para que este seja aplicado corretamente, as seguintes considerações devem ser levadas em conta:
 - A aplicação deve permitir uma efetiva modularização, com ciclos curtos de desenvolvimento;
 - Se a aplicação exigir alto desempenho e se este for obtido sintonizando as interfaces dos componentes, a abordagem RAD poderá não funcionar a contento;
 - A prototipação interativa e viva, aliada ao desenvolvimento incremental e não linear, são aspectos extremamente positivos para atingir os objetivos do desenvolvimento rápido;
 - A adoção da orientação a objetos ou o uso de linguagens de quarta geração é também um fator altamente positivo, principalmente, na aplicação de reuso de código;
- 

O Modelo Rapid Application Development (RAID)

- O modelo RAD propõe-se a ser um método leve, flexível e adaptável às novas tecnologias emergentes que considerem o RAD um processo organizado, gerenciado e padronizado.




O Modelo Evolucionários

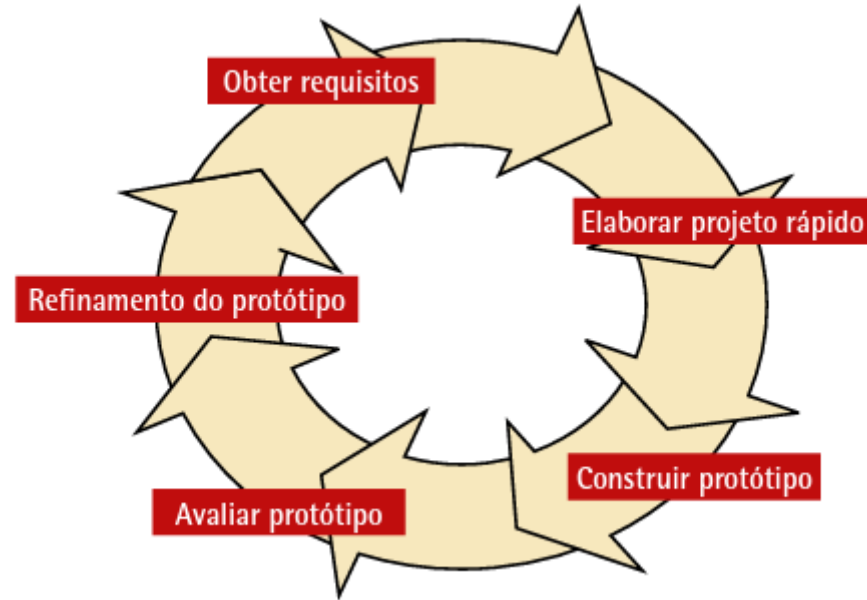
- Os modelos evolucionários ou modelos evolutivos, como o próprio nome sugere, são os explicitamente projetados para acomodar um produto de software que evolui com o tempo. A cada iteração, os modelos evolucionários ou evolutivos têm por objetivo produzir uma versão melhor e mais completa do software. Dois modelos se encaixam nessa definição: o de prototipagem e o espiral.



O Modelo de Prototipagem

- A prototipação é uma ferramenta que pode ser usada em qualquer um dos modelos apresentados até agora. Essa técnica auxilia o engenheiro de software e o cliente a entenderem melhor o que deve ser construído quando os requisitos estão confusos. Um protótipo é uma espécie de versão preliminar do software. Pode ser um software ou um plano no papel e concentra-se na representação dos aspectos do software que são visíveis para o cliente.
 - O objetivo é entender os requisitos do usuário e, assim, obter melhor definição dos requisitos do sistema. Possibilita que o analista crie um modelo (protótipo) do software que será construído. Esse protótipo é apropriado para quando o cliente não tiver definido detalhadamente os requisitos, mas tiver prazo curto para o desenvolvimento. O protótipo dá maior assertividade ao projeto com relação às necessidades do cliente ou usuário.
- 

O Modelo de Prototipagem



O Modelo de Prototipagem

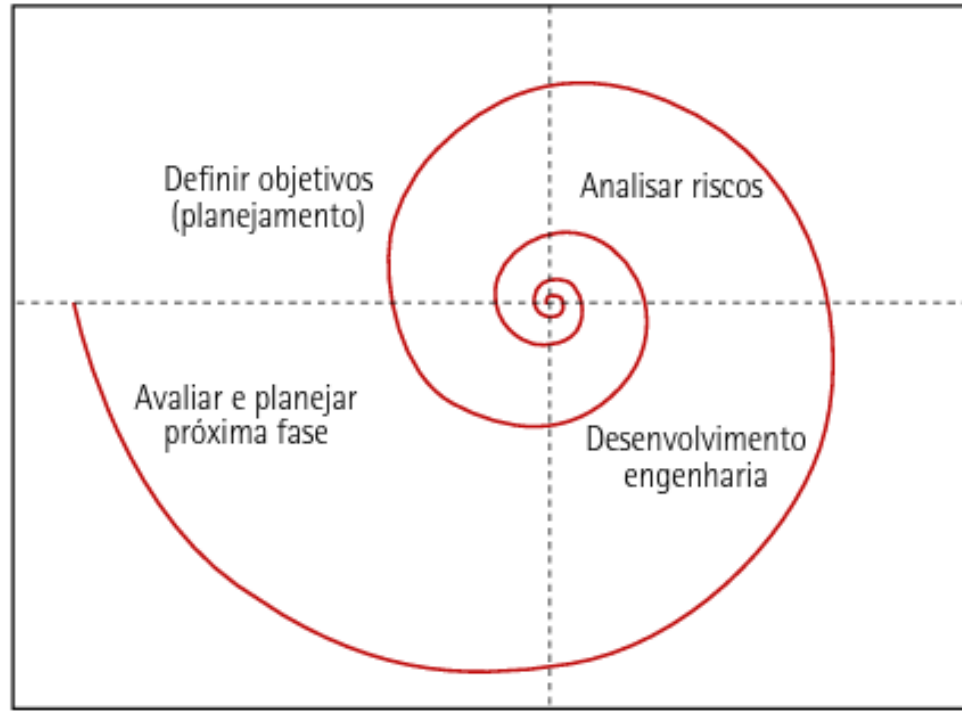
- A função da atividade obter requisitos é permitir que desenvolvedor e cliente definam os objetivos gerais do software, identifiquem quais requisitos são conhecidos e as áreas que necessitam de definições adicionais; elaborar projeto rápido é aquela em que se faz a representação dos aspectos do software que são visíveis ao usuário por meio de abordagens de entrada e formatos de saída; a atividade construir protótipo é a implementação rápida do projeto; avaliar protótipo é quando o desenvolvedor e o cliente avaliam o protótipo; e a atividade refinamento do protótipo é o momento em que o desenvolvedor refina os requisitos do software a ser desenvolvido na tecnologia final.
- Percebe-se claramente que a intenção desse modelo é apoiar o entendimento do software a ser construído e que o resultado da prototipagem não pode ser considerado a aplicação final que será colocada em produção ou operação. Quando todos os requisitos estiverem identificados e detalhados, e as interfaces, totalmente definidas, o protótipo deverá ser descartado, e a versão de produção, construída, considerando os critérios de qualidade.

O Modelo Espiral


- O modelo espiral foi proposto por Barry Boehm, em 1986, e tem como objetivo acoplar a natureza iterativa da prototipação aos aspectos controlados e sistemáticos do modelo em cascata; é dividido em uma série de atividades de trabalho ou regiões de tarefas e combina as características positivas da gerência de baselines (conjunto de documentos associados ao processo).




O Modelo Espiral




O Modelo Espiral

- O modelo espiral é uma evolução dos modelos clássicos e da prototipagem. Valoriza os pontos positivos desses modelos, desprezando os considerados negativos. Organiza o desenvolvimento como um processo iterativo em que vários conjuntos com quatro fases se sucedem, até que seja obtido o sistema ou software final.
 - Um novo ciclo se inicia (primeira fase) com a determinação de objetivos, alternativas e restrições, em que ocorre o comprometimento dos interessados e o estabelecimento de uma estratégia para alcançar os objetivos. Na segunda fase, a avaliação de alternativas, bem como a identificação e a solução de riscos se executam numa análise de riscos. Na terceira fase, ocorre o desenvolvimento do produto de software. Nesse quadrante, outros modelos podem ser empregados, inclusive, o modelo cascata. Na quarta e última fase, o produto é avaliado e prepara-se para iniciar um novo ciclo.
- 


Os Modelos Especializados

- Um modelo de processo especializado leva em conta muitas das características de um ou mais modelos tradicionais, tais como o modelo em cascata, o espiral, o evolucionário, entre outros. Todavia, existe a necessidade de aplicação, em produtos de softwares muito específicos, de uma abordagem mais especializada de engenharia de software, ou, quando definida, de uma forma mais restritiva. Os autores organizam esses modelos especializados em modelo de desenvolvimento, baseado em componentes, e modelo de métodos formais.
 - Com relação ao modelo de desenvolvimento baseado em componentes, tem-se:
 - O modelo baseado em componentes tem como ênfase criar sistemas de software que envolvam a composição de componentes, permitindo que sejam adicionadas, adaptadas, removidas e substituídas partes do sistema sem que seja necessária sua completa substituição; é a implementação do conceito da reusabilidade no desenvolvimento de software, tão comum em outras engenharias;
- 


Os Modelos Especializados

- Esse tipo de desenvolvimento auxilia a manutenção dos sistemas, visto que foca a integração de novos componentes já prontos ou, então, a atualização dos já existentes;
 - Dessa forma, essa abordagem enfatiza a criação ou a adaptação de componentes para que sejam utilizados em diversos sistemas; com isso, temos como resultado a reutilização, que busca flexibilizar o desenvolvimento;
 - Um tipo de desenvolvimento que utiliza essa filosofia é o formado pelos componentes de software comercial de prateleira, ou Commercial Off-The-Shelf (COTS), componentes desenvolvidos por vendedores que os oferecem como produtos e disponibilizam as suas funcionalidades juntamente com interfaces bem-definidas que permitem que esses componentes sejam integrados ao software a ser desenvolvido;
 - Nesse caso, a equipe de desenvolvimento do sistema ou da aplicação conhece pouco ou nada sobre o funcionamento interno de um componente, porém é fornecida uma interface externa bem-definida com a qual se deve trabalhar;
- 


Os Modelos Especializados

- esses componentes podem ser comprados, e a sua principal desvantagem, na maioria dos casos, é que não há código-fonte disponível; assim, a definição de como usar o componente é dada pelo fabricante e deve ser documentada, além de ser oferecido suporte na instalação e no uso desses componentes;
 - o que se procura em um componente é algo quase independente e uma parte substituível de um sistema que tem função bastante clara; os componentes possuem interface e, também, empregam regras de herança oriundas da tecnologia de objetos;
 - O modelo de desenvolvimento baseado em componentes possui uma abordagem iterativa e evolucionária; a essência é desenvolver aplicações comerciais ou científicas a partir de componentes de software pré-empacotados;
 - Uma característica importante nesse modelo é que as atividades de modelagem e construção começam com a identificação de possíveis candidatos a componentes, que podem ser projetados como módulos de software convencionais, como classes ou pacotes;
- 

Os Modelos Especializados

- De acordo com Medeiros, E. (2004), o modelo de desenvolvimento baseado em componentes possui as seguintes etapas: diversos produtos baseados em componentes existentes no mercado são pesquisados e avaliados; os itens de integração de componentes são considerados; projetase uma arquitetura de software para acomodar os componentes; integram-se os componentes à arquitetura; e realizam-se todos os testes para assegurar a funcionalidade adequada; todas essas etapas são independentes da tecnologia utilizada para criar os componentes.
- 

Os Modelos Especializados

- Com relação ao modelo de métodos formais, tem-se:
 - O modelo de métodos formais possui um conjunto de atividades que conduzem a uma especificação matemática formal do software e possibilita a especificação, o desenvolvimento e a verificação de um sistema baseado em computador por meio da aplicação de uma rigorosa notação matemática;
 - Os métodos formais oferecem três diferentes níveis de atividades:
 - nível 0: em que o software é descrito por meio de uma especificação formal que será usada como base para a implementação do sistema; esse nível é opcional e deve ser de custo-benefício menor;
 - nível 1: em que o desenvolvimento e a verificação formal são utilizados para produzir um software de maneira mais formal; esse nível é mais apropriado para sistemas de alta integridade que necessitem de segurança ou confiança;
- 

Os Modelos Especializados

- nível 2: em que provadores de teoremas ou simuladores podem ser utilizados, a fim de conduzir testes completos das propriedades de um sistema, de forma mais automatizada; esse nível é mais apropriado em sistemas nos quais o custo provocado por erros é extremamente alto.
- A vantagem dos métodos formais durante o desenvolvimento é que eles oferecem a eliminação de diversos problemas encontrados em outros modelos, como a ambiguidade, a incompletude e a inconsistência;
- Todos esses problemas podem ser descobertos e corrigidos mais facilmente com a aplicação da análise matemática; os métodos formais, quando utilizados durante o projeto, servem para verificar a programação, possibilitando, assim, a descoberta e a correção de erros que poderiam passar despercebidos;
- Em razão da sua característica, os modelos de métodos formais são mais utilizados quando se desenvolvem softwares que possuem fator crítico de segurança, ou quando a empresa pode sofrer pesados problemas econômicos caso ocorram erros no software;
- Alguns exemplos de software em que os métodos formais são aplicados são os sistemas de controle de aeronaves, engenharia aeroespacial e equipamentos médicos.

Dúvidas

