Engenharia de Sistemas de Informação I

Modelagem do Processo e Ciclo de Vida de Software

Marcos L. Chaim
Escola de Artes, Ciências e Humanidades – EACH
Universidade de São Paulo – USP

11-08-2007

Processo

- Envolve uma série de etapas que envolvem atividades, restrições e recursos para alcançar a saída desejada.
- Utiliza ferramentas e técnicas.

Processo

- Inclui:
  - Todas as principais atividades do processo.
  - Recursos; está sujeito a um conjunto de restrições (como um cronograma);
  - Produtos intermediários e finais;
  - Subprocessos, com hierarquia ou organizados de algum modo;
  - Critérios de entrada e saída para cada atividade;
  - Seqüência de atividades, de modo que a ordem de execução de uma para outra seja clara;
  - Conjunto de diretrizes que explicam os objetivos de cada atividade;
  - Restrições e controles para cada atividade, recurso ou produto.

Processo

Razões para modelar um processo:

- Formar um entendimento comum;
- Encontrar inconsistências, redundâncias e omissões;
- Encontrar e avaliar atividades propostas mais adequadas aos objetivos;
- Fazer um processo geral para uma situação particular na qual ele será utilizado,

Processo de Desenvolvimento

- Um processo de desenvolvimento de software compreende todas as atividades necessárias para definir, desenvolver, testar e manter um produto de software.
- Atividades fundamentais que são comuns a todos os processos de software

    1. *Especificação de software*: clientes e engenheiros definem o software a ser produzido e as restrições para a sua operação.
    2. *Desenvolvimento de software*: o software é projetado e programado.
    3. *Validação de software*: na qual o software é verificado para garantir que é o que o cliente deseja.
    4. *Evolução de software*: o software é modificado para se adaptar às mudanças dos requisitos do cliente e do mercado

Software is Hard

- Porcentagem de projetos que terminam dentro do prazo estimado: 10%
- Porcentagem de projetos que são descontinuados antes de chegarem ao fim: 25%
- Porcentagem de projetos acima do custo esperado: 60%
- Atraso médio nos projetos: um ano. Fonte: Chaos Report (1994)

Processo de Desenvolvimento

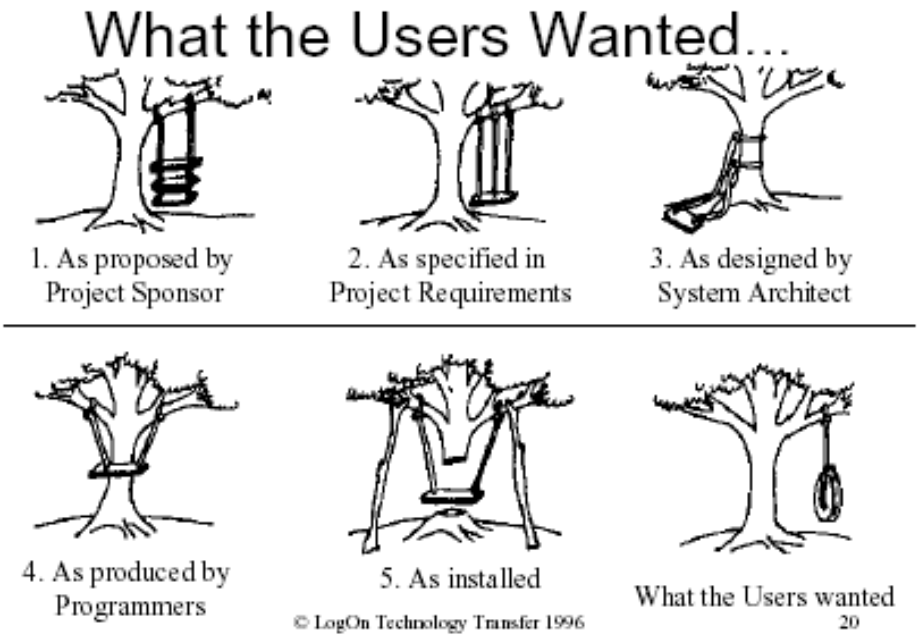
- Exemplos de processos de desenvolvimento
  - ICONIX, RUP, XP, UML Components & Catalysis
- Alguns objetivos de um processo de desenvolvimento são:
  - Definir *quais* as atividades a serem executadas ao longo do projeto;
  - Definir *quando, como e por quem* tais atividades serão executadas;
  - Prover *pontos de controle* para verificar o andamento do desenvolvimento;
  - Padronizar a forma de desenvolver software em uma organização.

Participantes do Processo

- Gerentes de projeto
- Analistas
- Projetistas
- Arquitetos de software
- Programadores
- Clientes
- Avaliadores de qualidade
- entre outros...

Participação do Usuário

A participação do usuário durante o desenvolvimento de um sistema é extremamente importante.



Atividades típicas

- Levantamento de requisitos
- Análise de requisitos
- Projeto
- Implementação
- Verificação & Validação (Testes)
- Implantação
- Evolução
  - Concentra-se nas modificações existentes nos sistemas de software para atender novos requisitos.

Requisitos

- Levantar requisitos
  - Enfatiza a busca junto ao cliente, seus sistemas e documentos, todas as informações possíveis sobre:
    - *Requisitos funcionais* - as funções que o sistema deve executar.
    - *Requisitos não funcionais* - as restrições sob as quais o sistema deve operar.
    - *Requisitos de domínio* - as características e as restrições do domínio da aplicação do sistema. Podem ser requisitos funcionais ou não funcionais.
- Organizar requisitos
  - Enfatiza a estruturação dos requisitos para que possam ser abordados nos ciclos de desenvolvimento

Análise de Requisitos

- Para criar o software de uma aplicação, é necessária uma descrição do problema e dos seus requisitos:
  - o que é o problema? o que o sistema deve fazer?
- Análise: O QUÊ o sistema deve fazer para satisfazer os requisitos
  - enfatiza a investigação do problema e dos requisitos
  - ex: sistema para biblioteca - quais são os processos de negócio relacionados a seu uso?
  - O que o sistema vai fazer - permitir consultas, empréstimo,..?
- Pode-se dizer que o resultado da análise é o enunciado do problema, e que o projeto será a sua resolução.
  - Problemas mal enunciados podem até ser resolvidos, mas a solução não corresponderá às expectativas.

Análise de Requisitos

- A qualidade do processo de análise é importante porque um erro de concepção resolvido na fase de análise tem um custo;
- Na fase de projeto tem um custo maior;
- Na fase de implementação maior ainda, e
- Na fase de implantação do sistema tem um custo relativamente astronômico.

Projeto

- Para criar o software de uma aplicação, também é necessário ter:
  - descrições de alto nível da solução lógica
  - descrições detalhadas da solução lógica
  - descrições de como a solução lógica atende os requisitos e as restrições
- Projeto: COMO o sistema deve ser organizado para satisfazer os requisitos
  - enfatiza uma solução lógica para o problema
  - cria soluções de alto nível e detalhadas
  - ex: sistema para biblioteca - como o sistema vai registrar um empréstimo?

Projeto

- A fase de projeto enfatiza a proposta de uma solução que atenda os requisitos da análise.
- Então, se a análise é uma investigação para tentar descobrir o que o cliente quer, o projeto consiste em propor uma solução com base no conhecimento adquirido na análise.

Implementação

- A utilização de técnicas sistemáticas nas fases de análise e projeto faz com que o processo de geração de código possa ser automatizado.
- Neste caso, cabe ao programador dominar as características específicas das linguagens, ferramentas, frameworks e estruturas de dados para adaptar o código gerado aos requisitos indicados quando necessário.

Verificação & Validação

Destina-se a mostrar que um software está em conformidade com sua especificação e que atende às expectativas do cliente que está adquirindo o software.

- Inspeções e Revisões: a cada estágio do processo de software, desde a definição de requisitos de usuário até a implementação do software.
- No entanto, maior parte dos custos de validação incorrem após a implementação, quando o software é testado.

Verificação & Validação: Testes

- Teste de componente (ou unidade). Os componentes individuais são testados para garantir que operem corretamente. Cada componente é testado independentemente, sem os outros componentes do sistema.
- Teste de integração. Os componentes são integrados para compor o sistema. Esse processo está relacionado com a busca de erros que resultam das interações não previstas entre os componentes e problemas de interface de componentes.
- Teste de sistema. O sistema é testado para assegurar que o projeto foi corretamente construído, fornecendo as saídas corretas de acordo com a especificação.

Verificação & Validação: Testes

- Teste de aceitação. Este é o estágio final do processo de teste, antes que o sistema seja aceito para o uso operacional. O sistema é testado com os dados fornecidos pelo cliente do sistema, em vez de dados simulados de teste.

Modelos de Processo de Software

A maioria dos processos de software é baseada em um dos quatro modelos gerais ou paradigmas de desenvolvimento de software

- Modelo em cascata e suas variantes (com prototipação; modelo V)
- Desenvolvimento iterativo e incremental.
- Modelo formal
  - Um modelo matemático do sistema é formalmente transformado em uma implementação.
- Engenharia de software baseada em componentes
  (*CBSE - Component Based Software Engineering*)
  - Supõe que partes do sistema já existem. O processo de desenvolvimento concentra-se mais na integração dessas partes do que no seu desenvolvimento a partir do início.

Modelo em cascata

Ver Figura 1

Modelo em cascata

Problemas: Ver Figura 2

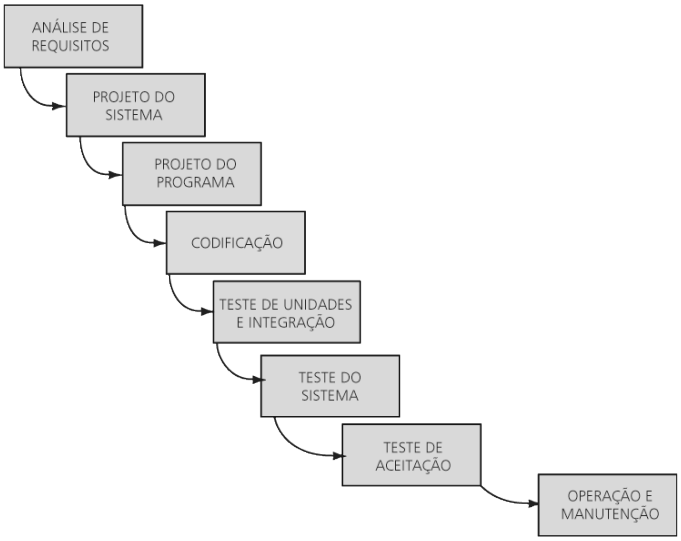


Figura 1: Modelo em cascata.

Modelo em cascata

Problemas:

- projetos raramente seguem um fluxo seqüencial;
- difícil de definir todas as restrições a priori;
- primeira versão em um estágio tardio;
- apesar dos problemas, foi largamente utilizado;
- melhor que não ter uma sistemática de desenvolvimento;

Modelo em cascata com prototipação

Problemas: Ver Figura 2

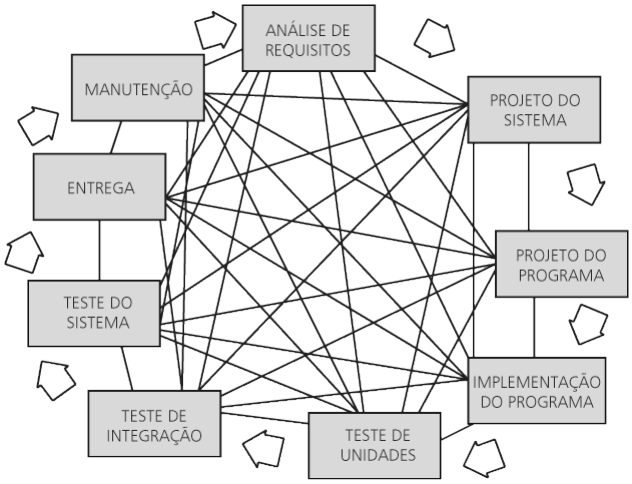


Figura 2: Processo de desenvolvimento na realidade.

Modelo em cascata com prototipação

Protótipo:

- é um produto parcialmente desenvolvido, que possibilita aos clientes e desenvolvedores examinarem certos aspectos do sistema e decidir se eles são ou não apropriados ou adequados para o produto acabado.
- pode ser composto de uma pequena parte do alguns requisitos-chave para assegurar que seja consistentes, viáveis e práticos; se não forem, as revisões serão feitas no estágio dos requisitos.
- para partes do projeto podem ser feitas protótipos; eles ajudam ao projetista a avaliar estratégias alternativas de projeto e decidir qual é a melhor para cada projeto.

Modelo em cascata com prototipação

Protótipo:

- freqüentemente, a interface do usuário é construída e testada como um protótipo.
- principais problemas são tratados e corrigidos bem antes de serem oficialmente validados no teste de sistema.

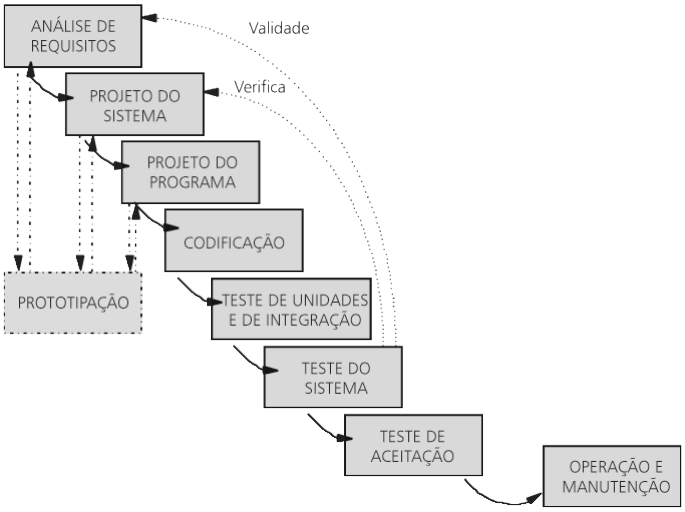


Figura 3: Modelo cascata com prototipação.

Modelo em V

- É uma variação do modelo em cascata, que demostra como as atividades de teste estão relacionadas coma a análise e projeto.
- A codificação forma o vértice do V, com a análise e projeto à esquerda e teste e a manutenção à direita. Ver Figura 4

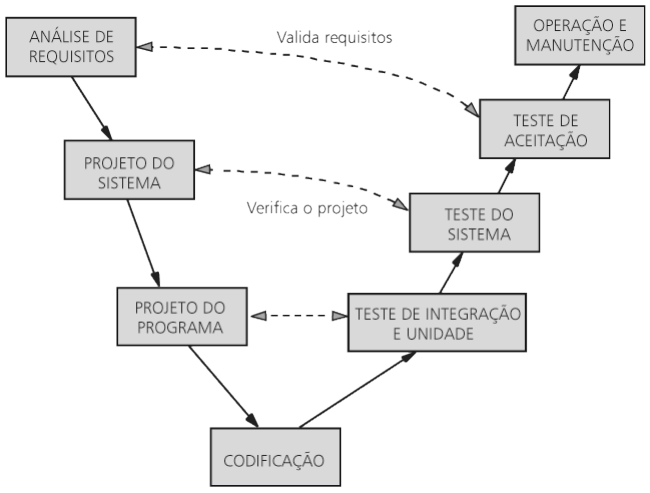


Figura 4: Modelo em V.

Modelo em V

- Teste de unidade e de integração garantem a qualidade dos programas.
- Teste de sistema deve verificar o projeto do sistema.
- Teste de aceitação, conduzido mais pelo cliente do que pelo desenvolvedor, valida os requisitos.
- A conexão entre os lados esquerdo e direito do modleo em V implica que, caso sejam encontrados problemas durante a verificação e a validação, o lado esquerdo do V pode ser executado novamente.
- Torna explícitas iterações e repetições de trabalho, ocultas no modelo cascata.
- Enfoque na atividade e na sua correção.

Prototipação

Ver Figura 5

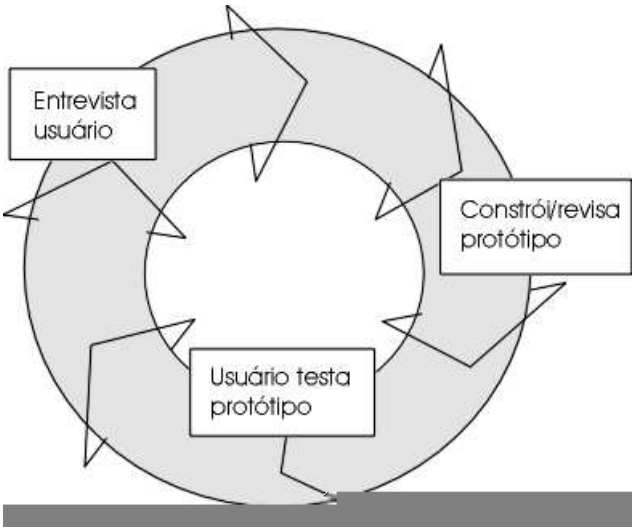


Figura 5: Modelo prototipação.

Prototipação

Benefícios:

- idéias confusas podem ser identificadas;
- entendimento errado pode ser esclarecido;
- complementar idéias vagas;
- utilidade do sistema antes de pronto.

Prototipação

Problemas:

- usuários vêem o protótipo como produto final e pensam que o produto está praticamente pronto;
- ferramentas/linguagem provisórias podem se tornar definitivas — inércia;
- manutenção de um protótipo tende a ser problemática: falta de estrutura porque não foi preparado para ser a versão final.

Modelo Iterativo e incremental

- **Iterativo**
  - o sistema de software é desenvolvido em vários passos similares.
- **Incremental**
  - Em cada passo, o sistema é estendido com mais funcionalidades.

- Divide o desenvolvimento de um produto de software em ciclos.
- Em cada ciclo de desenvolvimento, podem ser identificadas as fases de análise, projeto, implementação, testes e implantação.
- Cada ciclo considera um subconjunto de requisitos.
- Esta característica contrasta com a abordagem clássica, na qual as fases são realizadas uma única vez.

Modelo iterativo

Divide o desenvolvimento de um produto de software em ciclos.

Ver Figura 6

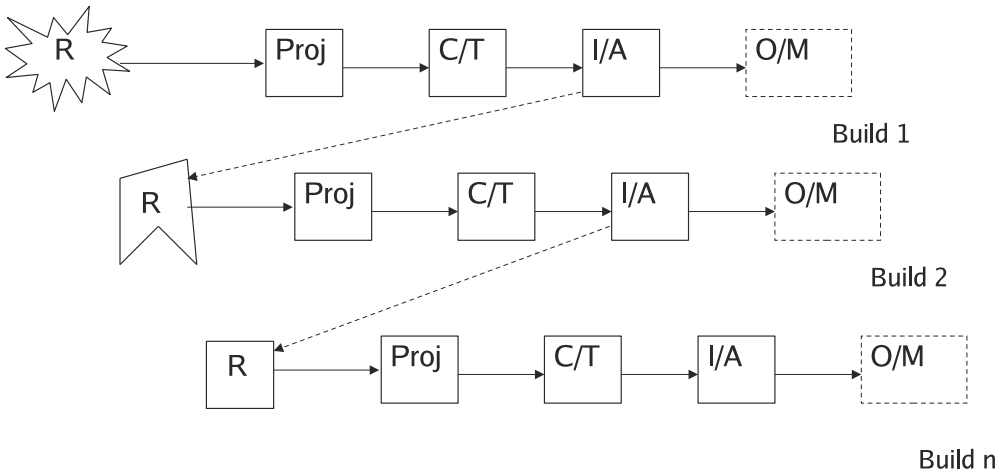


Figura 6: Modelo iterativo.

Modelo incremental

Divide o desenvolvimento de um produto de software em ciclos.

Ver Figura 8

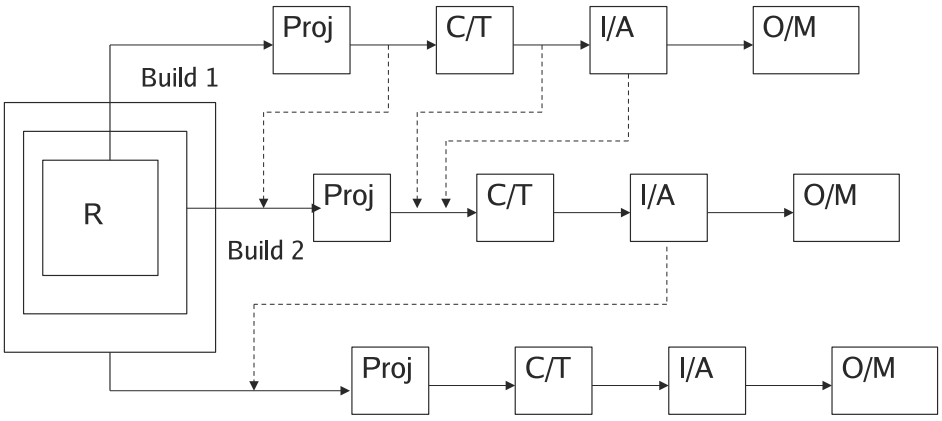


Figura 7: Modelo incremental.

Modelo espiral

Ver Figura 8

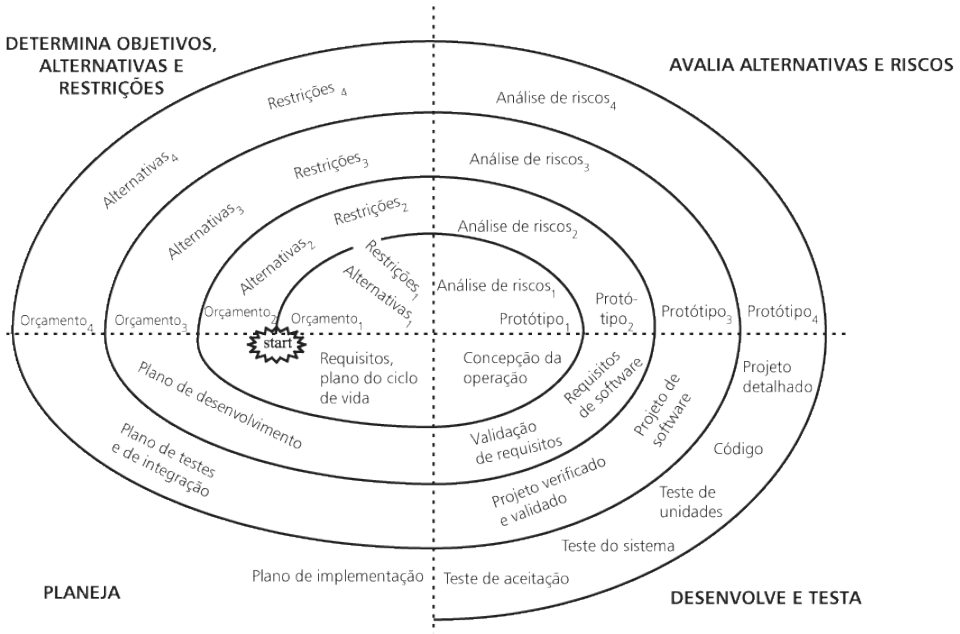


Figura 8: Modelo espiral de Boehm.

Modelo Iterativo e incremental

- Incentiva a participação do usuário.
- Riscos do desenvolvimento podem ser mais bem gerenciados.
  - Um risco de projeto é a possibilidade de ocorrência de algum evento que cause prejuízo ao processo de desenvolvimento, juntamente com as conseqüências desse prejuízo.
  - Influências: custos do projeto,cronograma, qualidade do produto, satisfação do cliente, etc.
- Mais difícil de gerenciar: possibilidade de ciclos em paralelo.

Cronograma de execução

O cronograma de execução dependerá dos seguintes fatores:

1. Tempo total estimado para o projeto (em hora/pessoa)
2. Tempo disponível (em semanas ou meses)
3. Tamanho e estruturação da equipe

Exemplo retirado do livro:

Raul S. Wazlawick, Análise e Projeto de Sistemas de Informação Orientados a Objetos, Campus, 2004

Ciclo	Casos de Uso	Entidades	Consultas	Observações	Esforço estimado
1	Emprestar Fita(550)	-	-	Neste ciclo não será implantado o mecanismo de persistência	550 horas
2	Devolver Fita (300)	-	-	Implementar mecanismo de persistência(300)	600 horas
3	Reservar Fita (270)	Fita, Cliente e Reserva(300)	-	-	570 horas
4	-	Emprestimo (100)	todas (400)	-	500 horas

Cronograma de execução

Analistas, projetistas e programadores a disposição (quantos forem necessários)

Dias:	1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-90
Ciclo 1	análise	projeto	implem.	testes				
Ciclo 2		análise	projeto	implem.	testes			
Ciclo 3			análise	projeto	implem.	testes		
Ciclo 4				análise	projeto	implem.	testes	
Implantação								implantação

1 Analista-projetista e 1 programador-testador a disposição

Dias:	1-20	21-40	41-60	61-80	81-100	101-20	121-40	141-60	161-80	181-200	201-20
Ciclo 1	A	P	I	T							
Ciclo 2			A	P	I	T					
Ciclo 3					A	P	I	T			
Ciclo 4							A	P	I	T	
Implantação											X

Exemplos retirados do livro:

Raul S. Wazlawick, Análise e Projeto de Sistemas de Informação Orientados a Objetos, Campus, 2004.

Modelo Formal

Um modelo matemático do sistema é formalmente transformado em uma implementação.

- Problemas
  - Apenas especialistas são capazes de aplicar a técnica. Requer treinamento em linguagens formais.
  - É difícil formalmente especificar alguns aspectos do sistema tais como interface humano-computador (IHC).
- Aplicação
  - Sistemas críticos especialmente aqueles onde a segurança deve ser assegurada/garantida antes do sistema ser posto em operação.

Desenvolvimento baseado em componentes

Abordagem baseada em reuso para definição, implementação e composição de componentes independentes não acoplados nos sistemas.

- Um componente é uma unidade de software cuja funcionalidade e dependências são completamente definidas por um conjunto de interfaces públicas.
  - Interfaces providas - serviços que o componente oferece
  - Interfaces requeridas - serviços que os componente necessita para prover os seus serviços (interfaces providas)

Desenvolvimento baseado em Componentes

Ver Figural 9.

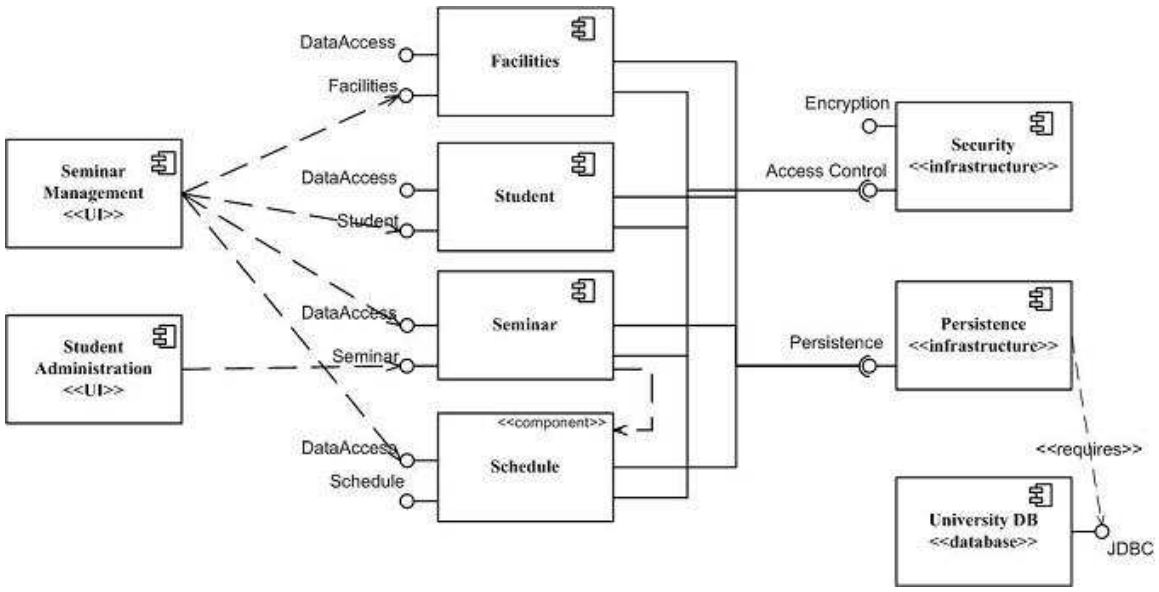


Figura 9: Desenvolvimento baseado em componentes.

Desenvolvimento baseado em Componentes

- Abstração: explícita separação entre a especificação e a implementação das funcionalidades. Os detalhes de implementação não são visíveis aos usuários das funcionalidades.
  - Flexibilidade e facilidade de manutenção - incorporação de mudanças ou inclusão de novas funcionalidades com mínimo impacto nas outras partes do sistema.
- Composição uniforme: A composição é adquirida através de conectores que reconhecem as interfaces dos componentes e fazem a adaptação de interfaces.
  - Interoperabilidade
  - Reusabilidade

Resumo

- Processo e processo de desenvolvimento.
- Modelos de ciclo de vida: modelo cascata, cascata com prototipação e modelo em V.
- Modelo baseado em prototipação.
- Modelo interativo e incremental.
- Model formal.
- Desenvolvimento baseado em componentes.

Notas de aula baseadas nas referências: [1] e [2].

Exercícios

1. Qual o modelo de processo a empresa que você trabalha utiliza?
2. Quais os benefícios e as desvantagens de se utilizar cada tipo de modelo de processo descrito nete capítulo?
3. Como cada modelo lida com uma mudança significativa nos requisitos no final do desenvolvimento?
4. Uma organização de desenvolvimento deveria adotar um único modelo de processo de desenvolvimento para todo software que ela desenvolve. Discuta os prós e contras.
5. Qual dos modelos propostos acomoda melhor a mudança de requisitos?

*

Referências

[1] Shari Lawrence Pfleeger. *Engenharia de Software: Teoria e prática*. Prentice-Hall, São Paulo, 2a. edition, 2004.

[2] Ian Sommerville. *Engenharia de Software*. Pearson Addison-Wesley, São Paulo, 8a. edition, 2007.