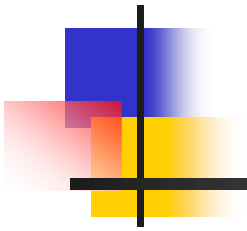
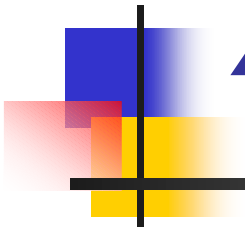


Engenharia de Software

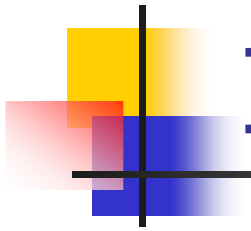
Prof. Grinaldo Oliveira

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas



Aula Inicial

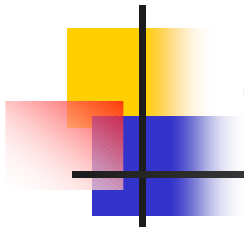
Introdução a Engenharia de Software



Introdução

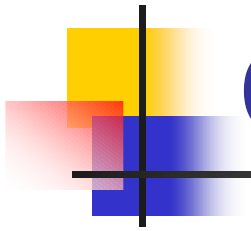
- "O Software ultrapassou o Hardware como chave para o sucesso de muitos sistemas baseados em computador"

(Pressman, pg. 3, 1992)



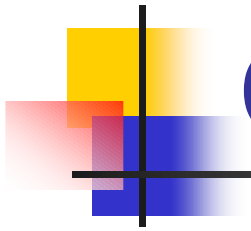
O que é Software?

- Definição - Software é:
  - *1º* - instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados;
  - *2º* - estruturas de dados que permitem a manipulação das informações;
  - *3º* - documentos que descrevem a operação e uso dos programas.



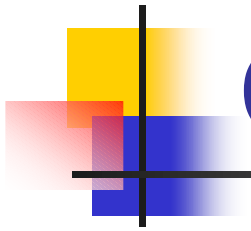
Características do Software - 1

- O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico:
  - Custos são concentrados no trabalho de engenharia.
  - Projetos não podem ser geridos como projetos de manufatura.
  - “Fábrica de Software!”



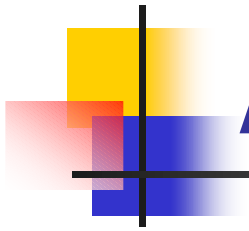
Características do Software - 2

- Software não desgasta!
  - Software não é sensível aos problemas ambientais que fazem com que o hardware se desgaste.
  - Toda falha indica erro de projeto ou implementação: manutenção do SW é mais complicada que a do HW.



Características do Software - 3

- A maioria dos softwares é feita sob medida e não montada a partir de componentes existentes.
- Situação esta mudando:
  - Orientação a objetos.
  - Reusabilidade é o "Santo Graal"(diminui custos e melhora projetos).

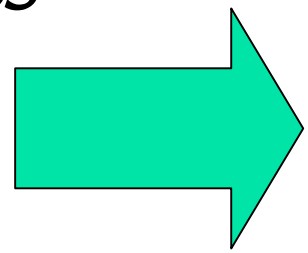


A importância do Software

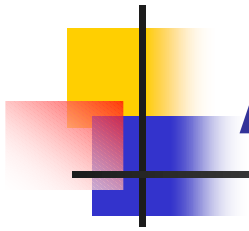
- Durante as 3 primeiras décadas da era do computador, o principal desafio era desenvolver um **HARDWARE** de baixo custo e alto desempenho.
- O hoje o desafio é melhorar a qualidade (e reduzir os custos) das soluções baseadas em **SOFTWARE**!

O Software é o que faz a diferença!!!

- *Completeza* da informação
- *user-friendlyness*
- *web-enhanced*
- inteligência
- funcionalidade
- compatibilidade
- suporte



Tornam 1 produto melhor que outro

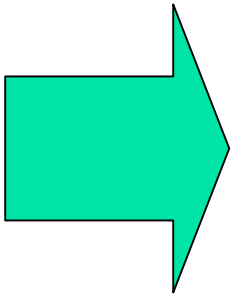


Aplicações de Software

Pressman, página 20

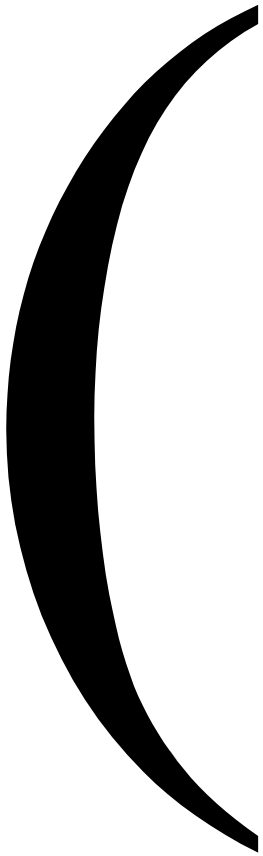
- Software Básico
- Software de Tempo Real
- Software Comercial
- Software Científico ou de Engenharia
- Software Embutido
- Software de Computador Pessoal
- Software de Inteligência Artificial

A evolução do Software

Computação 

- Nova Revolução Industrial (*Toffler*)
- 3a. Onda

Parêntesis: Revolução Industrial Primeira Onda

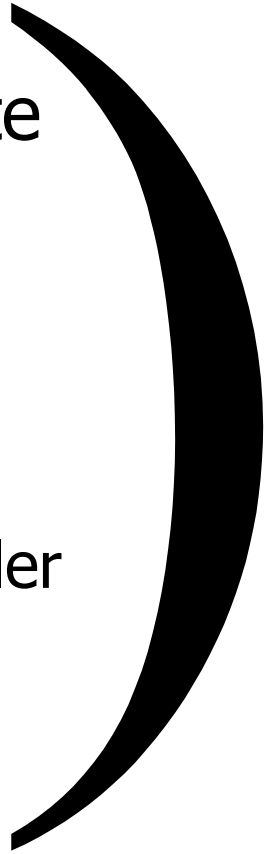


- Ferro (Darby, 1709)
- Máquina a vapor:
  - Inventada (Newcomen, 1712)
  - Aperfeiçoada (WATT, 1766 - ’69 -’82)
- Mecanização da indústria têxtil:
  - Tear Mecânico (Kay, 1722)
  - Maquina de fiar (Hargreaves, 1764)
- Aspectos sociais, políticos e econo

Têxteis, Carvão e Ferro

Parêntesis: Revolução Industrial Segunda Onda

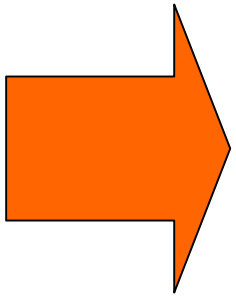
- Aço (Bessemel, 1856 e 1885 - Liga)
- Locomotiva a Vapor (Rede de Transporte - 1830)
- Máquina de Costura (SINGER,1851)
- Motor a combustão interna:
  - Primeiro eficiente (OTTO, 1876)
  - Produção automobilística em massa (Daimler e Benz, 1896)
- Desemprego e fim da escravidão



Revolução Industrial: Terceira Onda

- Energia Nuclear (Fermi, 1942)
- Uso Industrial/Comercial da Eletricidade
- Computadores Eletrônicos (ENIAC 1946)
- Transistor (Shockley, et al., 1948)

Sociedade Industrial



transformação

Sociedade da Informação

Filosofando...

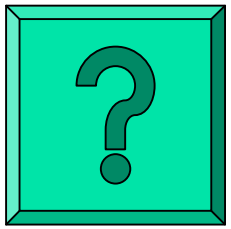
- A mudança de uma sociedade industrial para uma baseada na informação é uma *Radical Mudança Econômica*:
  - Material tem menos valor e Informação tem mais valor

Antes: quanto menos pessoas possuísse algo, maior o valor.

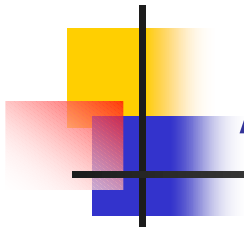
Hoje: quanto mais pessoas possuem algo, maior o valor.

Filosofando ... Exemplo!

- Cite as características dos sistemas operacionais que você conhece.
- Compare os sistemas:
  - Unix
  - Windows
  - MacOS

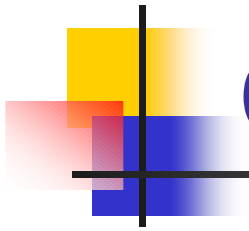


O Windows vende mais porque é mais fresquinho ou é mais fresquinho porque vende mais???



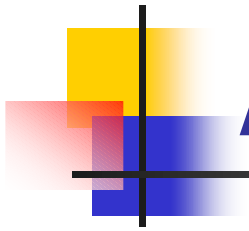
A evolução do software

- Software é dividida em 5 Eras:
  - Primeiros anos 1950 - 1965
  - Segunda Era 1965 - 1975
  - Terceira Era 1975 - 1985
  - Quarta Era 1985 – 1990
  - Quinta Era 1990 - ...



Os primeiros anos... (1950-1965)

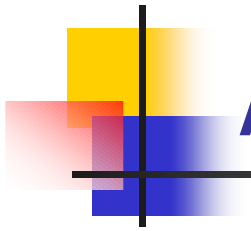
- Softwares eram projetados para aplicações específicas.
- Computação era para especialistas – quem usava a aplicação era o próprio desenvolvedor.
- Poucos recursos de hardware disponíveis.
- Documentação inexistente.



A Segunda Era... (1965-1975)

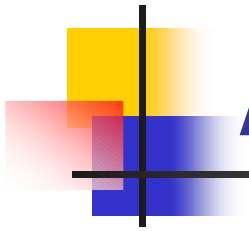
- Sistemas multiusuários.
- Surgiram os primeiros SGBDs.
- Advento das "softwares houses."
- Reaproveitamento de bibliotecas de softwares.
- Manutenção absorveu recursos em índices alarmantes.

CRISE DO SOFTWARE!



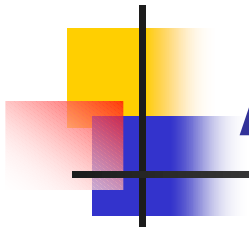
A Terceira Era... (1975-1985).

- Computador tornou-se objeto popular.
- Disseminação das redes locais.



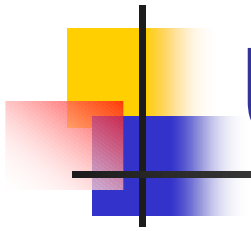
A Quarta Era... (1985-1990).

- Arquitetura Cliente-Servidor.
- Aplicativos com interface gráfica.
- Sistemas especialistas.



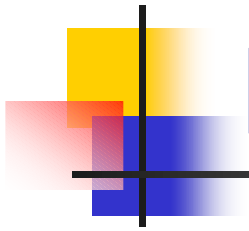
A Quinta Era... (1990-???)

- Sistemas Distribuídos.
- Aplicações WEB.
- Agentes Móveis.
- Sistemas voltados para Gestão do Conhecimento.



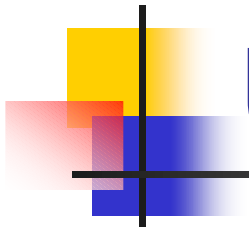
Uma Crise no horizonte

- A industria de Software tem tido uma “crise” que a acompanha há quase 30 anos:
  - Aflição Crônica != Crise
- Problemas não se limitam ao software que não funciona adequadamente, mas abrange:
  - desenvolvimento, testes, manutenção, suprimento, etc.



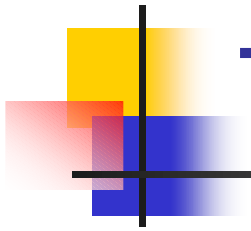
E quais são as causas?

- Gerentes sem experiência em softwares;
- Pouco treinamento;
- Resistência a mudanças;
- Existência de certo "mitos":
  - Administrativos:
    - "Se estamos atrasados nos prazos, podemos adicionar mais programadores e tirar o atraso".
  - Cliente:
    - "Uma declaração geral dos objetivos é o suficiente para começar a escrever programas – podemos preencher os detalhes mais tarde".
  - Profissional:
    - "A única coisa a ser entregue em um projeto bem-sucedido é o programa funcionando".



Uma Crise no horizonte

- Conjunto de problemas encontrados no desenvolvimento e manutenção de software:
  - Estimativas de custos e prazos vencidos;
  - Insatisfação do cliente;
  - Qualidade de software inferior à esperada;
  - Dificuldade de manutenção.



Therac-25

- Equipamento de Radioterapia.
- Entre 1985 e 1987 se envolveu em 6 acidentes, causando mortes por overdoses de radiação.
- Software foi adaptado de uma antecessora, Therac-6:
  - falhas por falta de testes integrados
  - falta de documentação

Denver International Airport

- Custo do projeto: US$ 4.9 bilhões
  - 100 mil passageiros por dia
  - 1,200 vôos
  - 53 milhas quadradas
  - 94 portões de embarque e desembarque
  - 6 pistas de pouso / decolagem

Denver International Airport

- Erros no sistema automático de transporte de bagagens (*misloaded*, *misrouted*, *jammed*):
  - Atraso na abertura do aeroporto com custo total estimado em US$360 Milhões
- 86 milhões para consertar o sistema

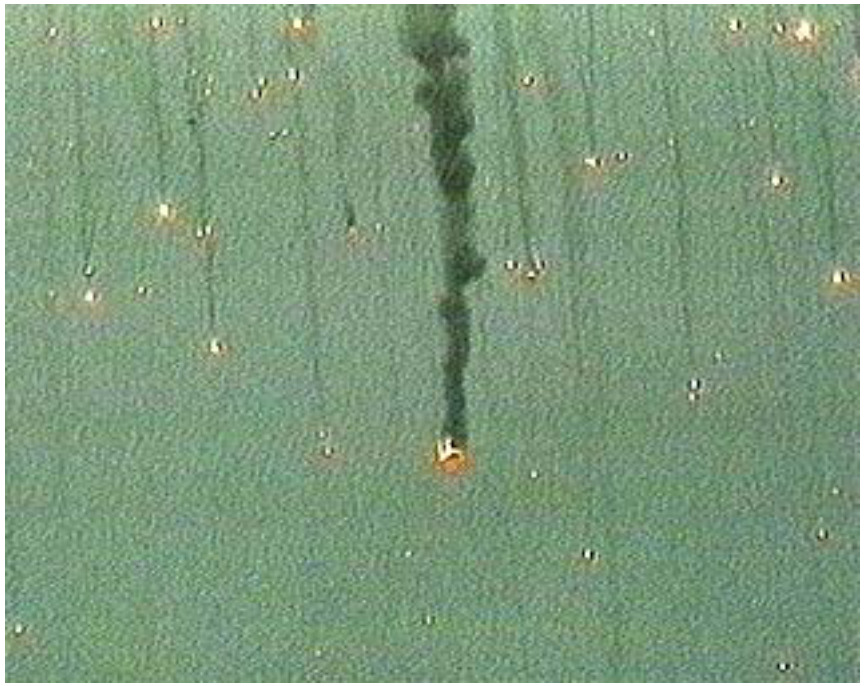
Ariane 5

Ariane 5

- Projeto da Agência Espacial Européia que custou:
  - 10 anos.
  - US$ 8 Bilhões.
- Capacidade 6 toneladas.
- Garante supremacia européia no espaço.

Vôo inaugural em 4/junho/1996

Resultado



- Explosão 40 segundos após a decolagem.
- Destruição do foguete e carga avaliada em US$ 500 milhões.

O que aconteceu? (I)

- Fato: o veículo detonou suas cargas explosivas de autodestruição e explodiu no ar. Por que?
- Porque ele estava se quebrando devido às forças aerodinâmicas. Mas por que?
- O foguete tinha perdido o controle de direção (atitude). Causa disso?
- Os computadores principal e back-up deram shut-down ao mesmo tempo.

O que aconteceu? (II)

- Por que o Shut-down? Ocorrera um *run time error* (out of range, overflow , ou outro) e ambos computadores se desligaram. De onde veio este erro?
- Um programa que convertia um valor em ponto flutuante para um inteiro de 16 bits recebeu como entrada um valor que estava fora da faixa permitida.

Especificamente:O que faltou?

strict precondition 1:

{

Set."x"=FLPT and Set."y"=INT16

and -32768 <= x <= +32767

}

program code:

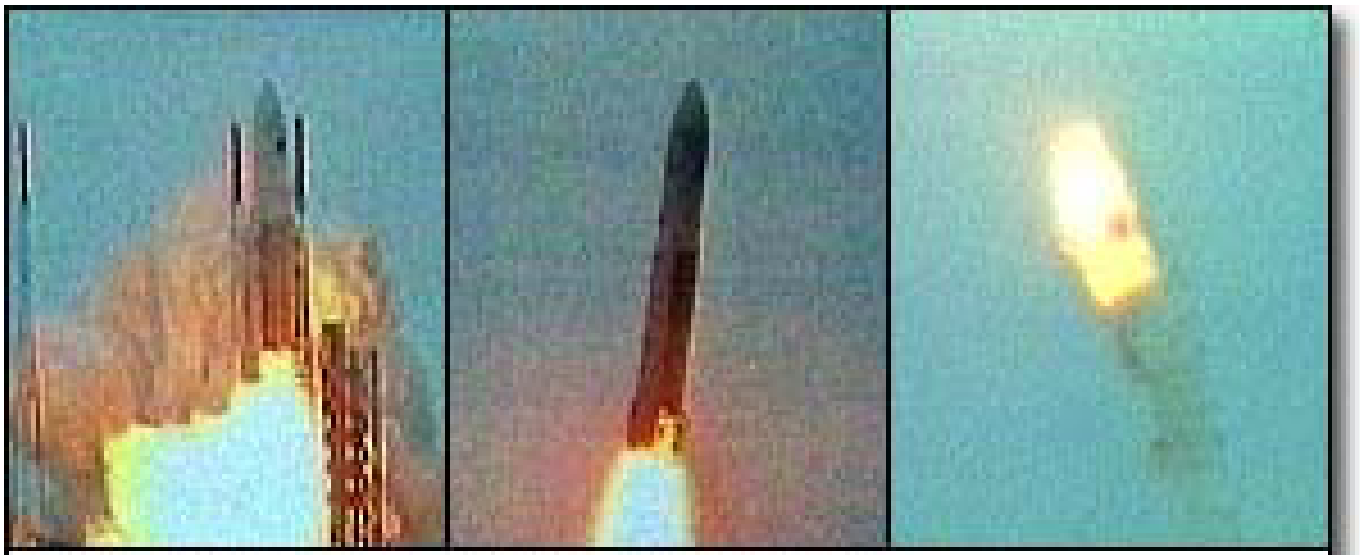
y := int(x);

postcondition:

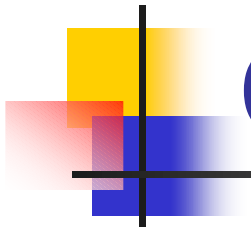
{Set."x"=FLPT and Set."y"=INT16 and y=int(x)}

Ironia...

- O resultado desta conversão não era mais necessário após a decolagem...

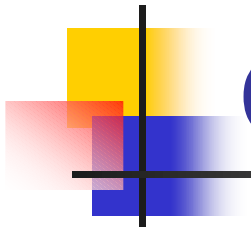


The rocket exploded seconds after launching



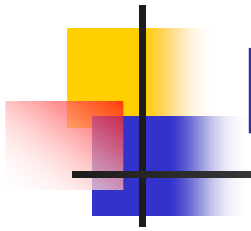
Quais são os problemas?

- A sofisticação do software ultrapassou nossa capacidade de construção.
- Nossa capacidade de construir programas não acompanha a demanda por novos programas.
- Nossa capacidade de manter programas é ameaçada por projetos ruins.



Como resolver estes problemas?

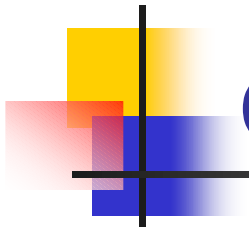
ENGENHARIA DE SOFTWARE!!!



Engenharia de Software

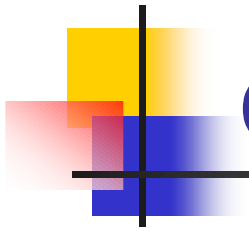
- "Engenharia de Software é o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais". [PRESSMAN, 1992].

- Abordagem sistemática, disciplinada e possível de ser medida para o desenvolvimento, operação e manutenção do software.
- METODOLOGIA !!!!!!!!!!!!!



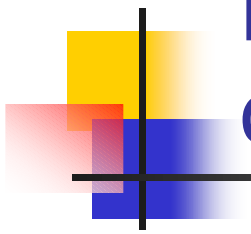
O que compõe a Engenharia de Software?

- Métodos:
  - Indicam ”como fazer”.
  - Abrangem tarefas relacionadas ao planejamento e estimativa do projeto, análise de requisitos do software, projeto da estrutura de dados, codificação do software, além de testes e manutenção.
- Ferramentas:
  - Apoio automatizado aos métodos.
  - Ferramentas CASE (Computer-Aided Software Engineering).



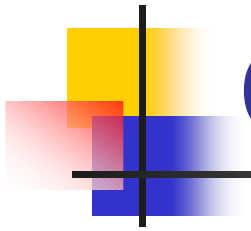
O que compõe a Engenharia de Software?

- Procedimentos:
  - Elo de ligação entre os métodos e as ferramentas.
  - Definem um conjunto de itens de suma importância no desenvolvimento do software:
    - a sequência em que os métodos serão aplicados;
    - os produtos que devem ser entregues (documentos, relatórios, etc.);
    - controles de qualidade;
    - marcos de referência que permitam ao gerente controlar o progresso do projeto.



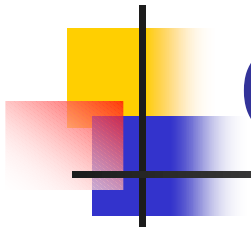
Perguntas que Engenharia de Software quer responder:

- Porque demora tanto para concluir um projeto (não cumprimos prazos)?
- Porque custa tanto (uma ordem de magnitude a mais)?
- Porque não descobrimos os erros antes de entregar o software ao cliente?
- Porque temos dificuldade de medir o progresso enquanto o software está sendo desenvolvido?



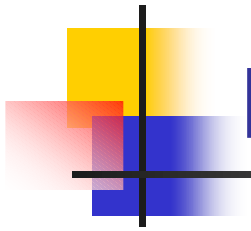
Causas óbvias

- Não dedicamos tempo para coletar dados sobre o desenvolvimento do software - resulta em estimativas “a olho”.
- Comunicação entre o cliente e o desenvolvedor é muito fraca.
- Falta de testes sistemáticos e completos.



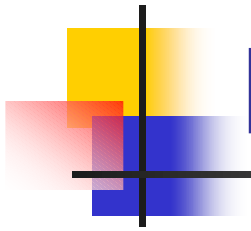
Causas menos óbvias

- O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico (característica 1).
- Gerentes sem *background* em desenvolvimento de SW.
- Profissionais recebem pouco treinamento formal.
- Falta investimento (em ES).
- Falta métodos e automação.



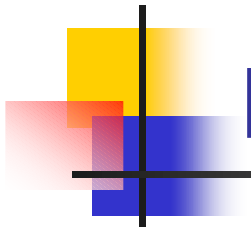
Mitos do Software - Administrativos

- Um manual oferece tudo que se precisa saber.
- Computadores de última geração solucionam problemas de desenvolvimento.
- Se estamos atrasados, basta adicionar programadores e tirar o atraso (chamado "*conceito de hordas de mongois*").



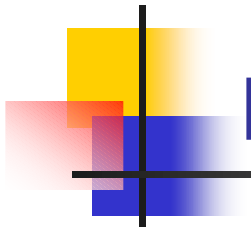
Mitos do Software - do Cliente

- Uma declaração geral é suficiente para começar a escrever programas.
- Mudanças podem ser facilmente acomodadas em um projeto



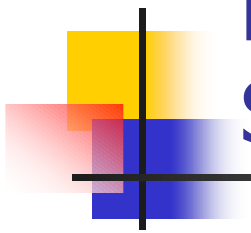
Mitos do Software - do Profissional

- Um programa está terminado ao funcionar.
- Quanto mais cedo escrever o código, mais rápido terminarei o programa.
- Só posso avaliar a qualidade de um programa em funcionamento.
- A única coisa a ser entregue em um projeto é o programa funcionando.



Engenharia de Software: Paradigmas

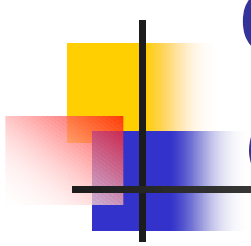
- A engenharia de Software compreende um conjunto de etapas envolvendo métodos, ferramentas e procedimentos.
- Essas etapas são conhecidas como Paradigmas da Engenharia de Software ou Modelos de Processos.
- Escolhido com base em alguns fatores:
  - a natureza do projeto e da aplicação;
  - os métodos e as ferramentas a serem usados;
  - os controles e os produtos que precisam ser entregues.



Principais Paradigmas de Engenharia de Software

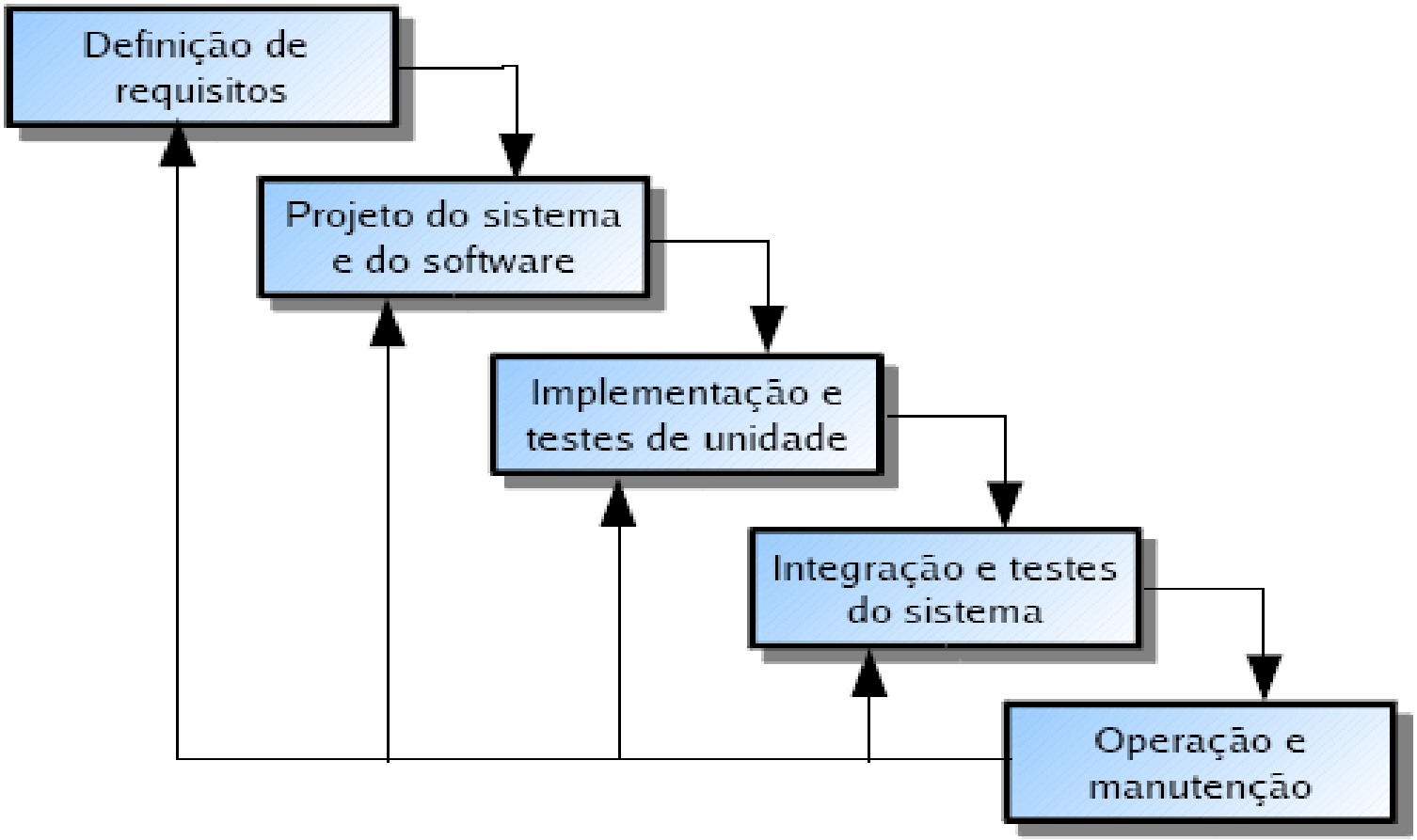
- Cascata
- Prototipação
- Evolutivo
  - Incremental
  - Espiral
- RAD
- Métodos Formais

Ciclo de Vida Clássico: modelo *Cascata (Waterfall)*

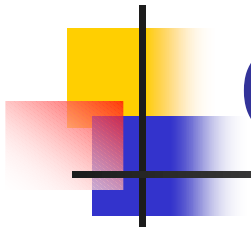


- É o mais antigo e utilizado paradigma da Engenharia de Software.
- Origina-se do ciclo da engenharia convencional
- Requer uma abordagem sistemática, seqüencial ao processo de desenvolvimento do software.
- Divide o processo de construção em fases ou etapas:
  - Uma etapa só tem início ao final da etapa anterior;
  - O resultado da etapa anterior serve como entrada para a etapa subseqüente.

Ciclo de Vida Clássico (II)

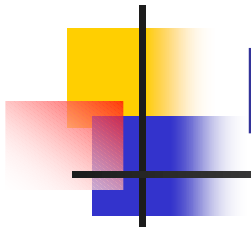


de Sommerville, 1995.



Ciclo de Vida Clássico (III)

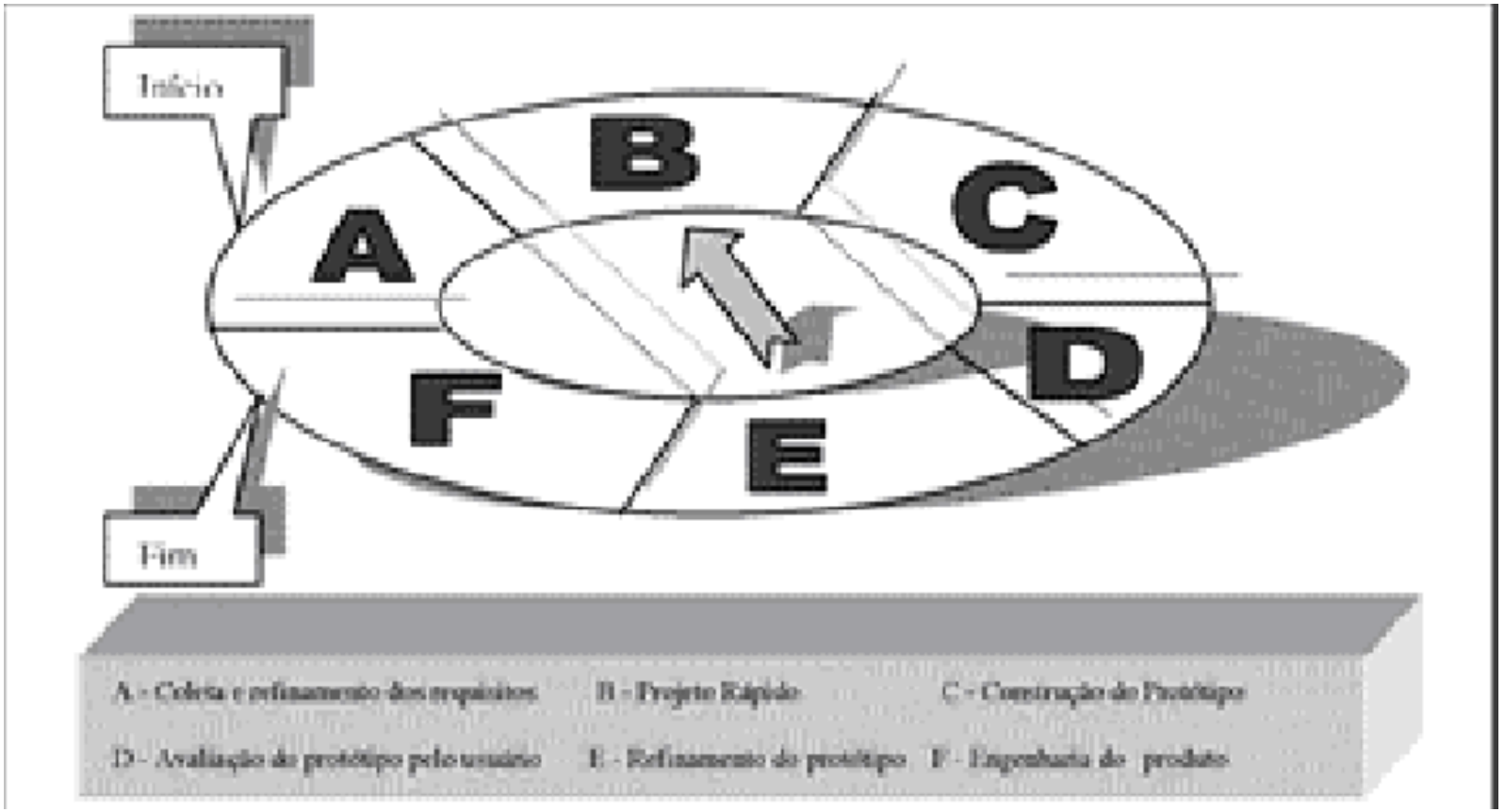
- Problemas:
  - projetos reais não seguem um fluxo seqüencial: dificuldade de acomodar mudanças depois de iniciado.
  - Dificuldade de declaração de todas as exigências pelo cliente.
  - Paciência!

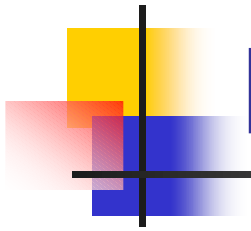


Prototipação

- Tem como objetivo direcionar o processo para a construção de um protótipo que ajude o cliente e o analista a entender o que deverá ser feito.
- Indicado quando existem dificuldades para definir as características e o comportamento adequados.
- Assim como o modelo em cascata, possui uma seqüência de eventos

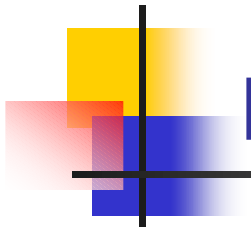
Prototipação





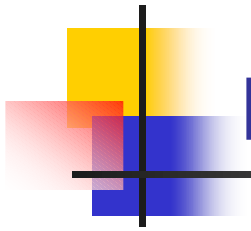
Prototipação

- Tudo começa com a coleta e refinamento dos requisitos (cliente e analista trabalhando juntos) – (A).
- Após esse passo, o analista elabora um "projeto rápido" com base nos aspectos visíveis ao usuário - (B).
- Com base nesse projeto rápido, o analista constrói o protótipo -(C).
- a etapa subseqüente, o cliente avalia o protótipo construído - (D).
- Ajustes são efetuados com base na avaliação do cliente – (E).
- Produto concluído - Descartá-lo ou aproveitar partes (F).



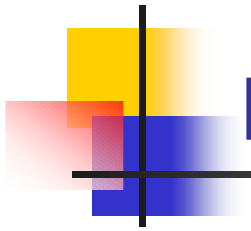
Problemas do Modelo de Prototipação

- Cliente dificilmente entende que o protótipo não é funcional e que não são necessários apenas "alguns acertos" para colocá-lo em produção.
- O analista muitas vezes faz concessões de implementação a fim de colocar o protótipo em funcionamento rapidamente.
- Mesmo assim...
  - A prototipação é um paradigma eficiente da Engenharia de Software. A chave é definir-se as regras do jogo no começo da prototipação – e respeitá-las no final.



Modelos Evolutivos

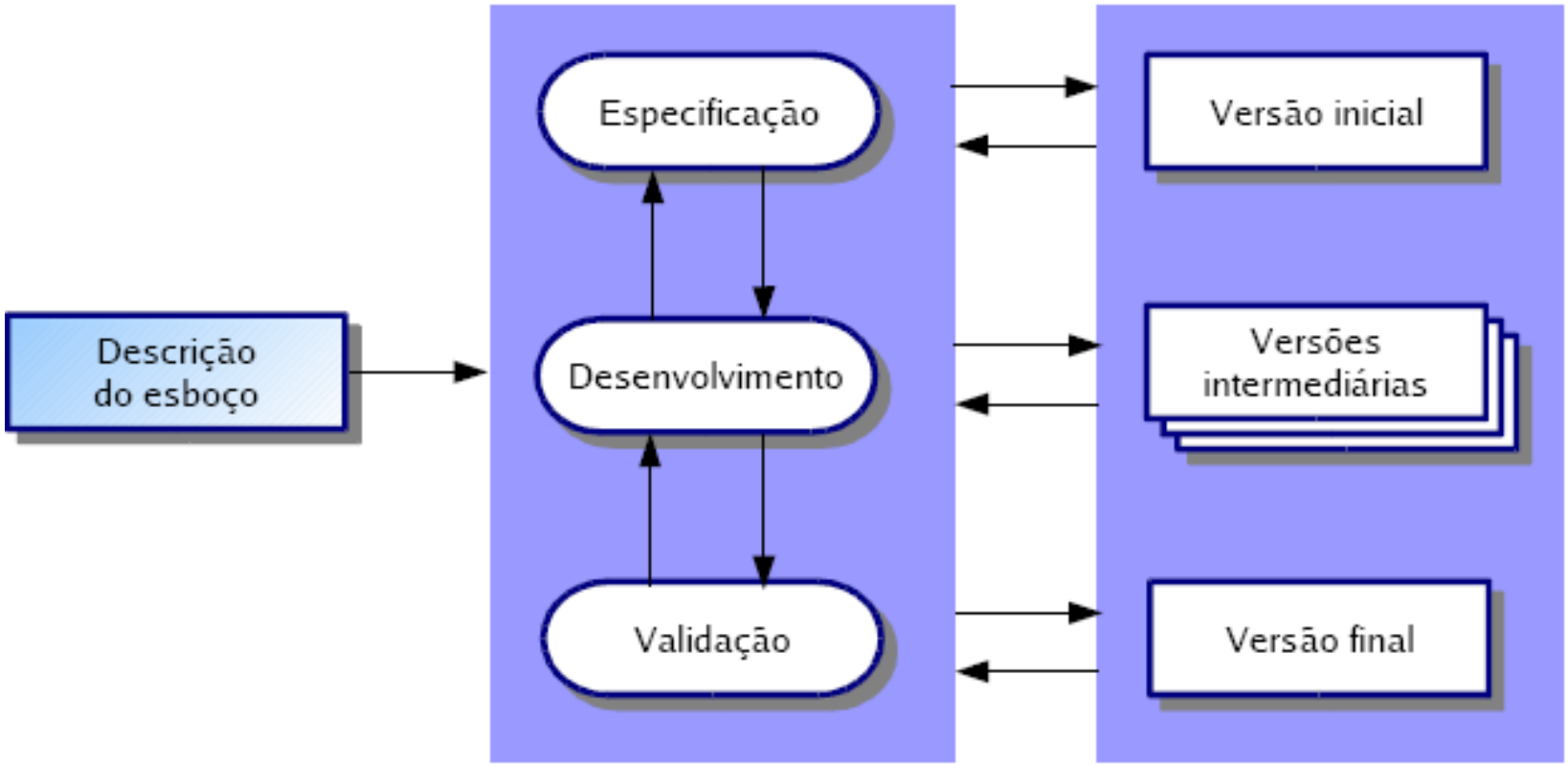
- São modelos de processo que acomodam a evolução do produto com o tempo.
- O objetivo é trabalhar com os clientes e evoluir para um sistema final a partir de uma especificação genérica inicial.
- O desenvolvimento inicia com as partes do sistema que estão compreendidas.
- É necessário o uso de ferramentas específicas para apoiar o processo de desenvolvimento do software;
- Trabalha-se também com protótipos descartáveis:
  - O objetivo é compreender os requisitos do sistema. O protótipo se concentra em fazer experimentos com partes dos requisitos que estejam mal compreendidas

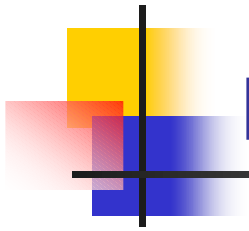


Modelos Incremental

- Ao invés de entregar o sistema de uma única vez, quebra-se o desenvolvimento e a entrega em incrementos, com cada incremento entregando parte da funcionalidade requerida.
- Os requisitos do usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, ainda que os requisitos para incrementos posteriores continuem a evoluir.
- Para que o modelo incremental funcione, é necessário que a arquitetura da aplicação permita a entrega de partes independentes.

Modelo Incremental (Sommervile 2003)



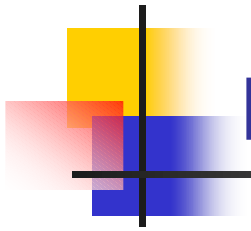


Problemas do Modelo Incremental

- Falta de visibilidade do processo: dificuldade de planejamento e gerenciamento.
- Os sistemas freqüentemente possuem pouca estrutura
- Podem ser exigidas habilidades especiais (p.ex.em linguagens para desenvolvimento rápido)

Mesmo assim...

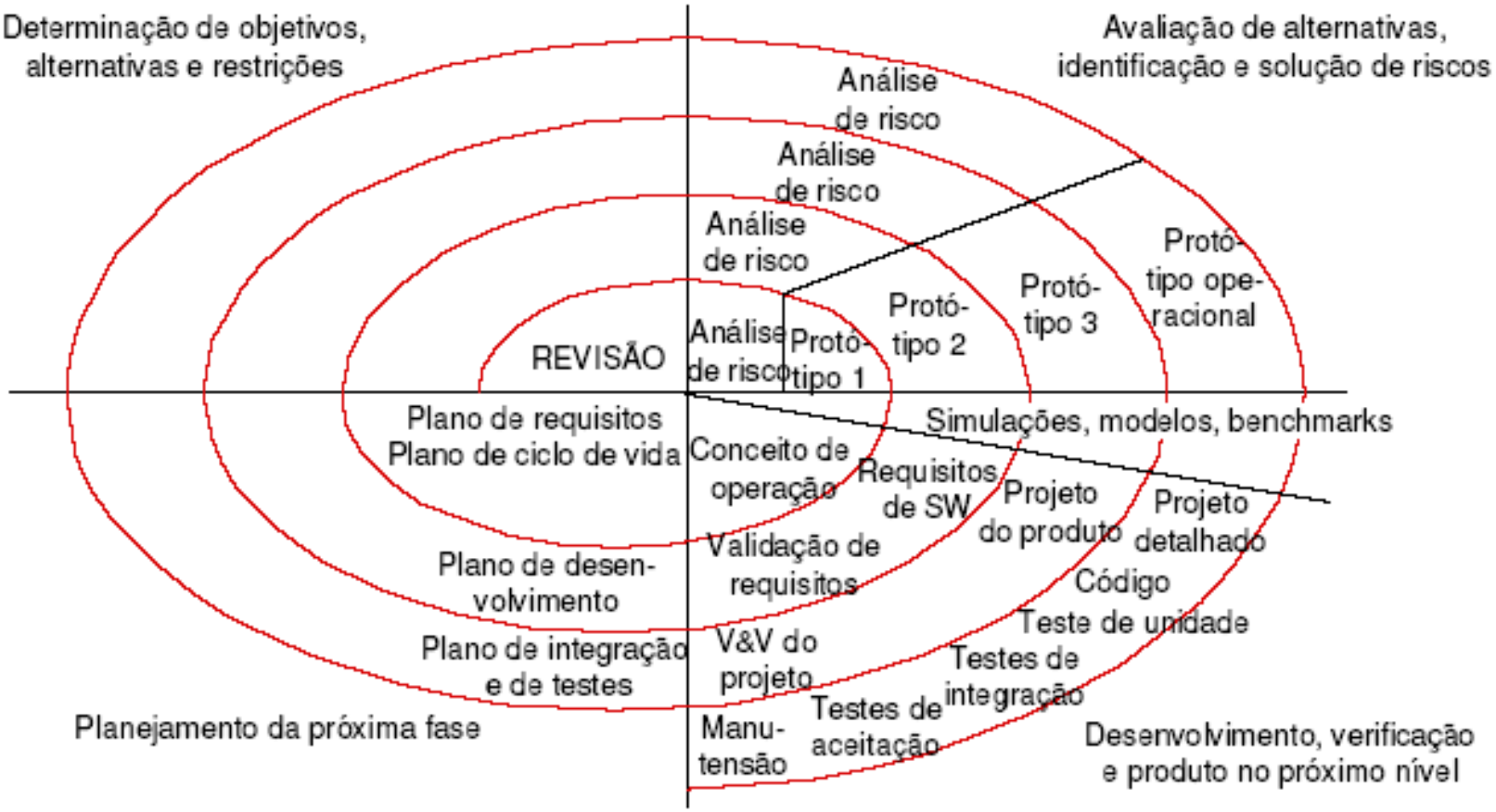
- O modelo evolutivo é indicado para:
  - sistemas interativos pequenos ou de médio porte;
  - partes de sistemas grandes (p.ex., a interface com o
  - usuário);
  - para sistemas de vida curta.

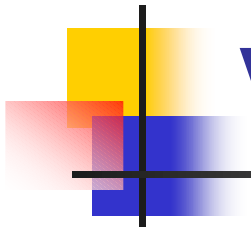


Modelo em Espiral

- O processo é representado como uma espiral. Cada ciclo da espiral é uma fase do processo
- Cada ciclo determina quatro etapas fundamentais:
  - Definição de objetivos, alternativas e restrições.
  - Análise e redução de riscos
  - Desenvolvimento e validação
  - Planejamento do próximo ciclo
- No modelo espiral, não há fases fixas pré-definidas. Elas são definidas de acordo com os objetivos.
- É um meta-modelo: qualquer modelo pode ser derivado a partir do modelo espiral.

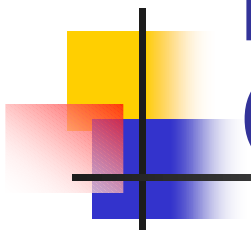
Modelo Espiral de Boehm (1988)





Vantagens do modelo Espiral

- Abordagem mais realista para o desenvolvimento de software em grande escala.
- Devido ao foco dado à análise de riscos, permite que o gerente e o cliente acompanhem e gerenciem os riscos de uma forma mais segura.
- No entanto...
  - O modelo espiral exige considerável experiência na avaliação de riscos. Se um grande risco não for descoberto, problemas ocorrerão.
  - Evidencia-se a importância de uma boa Gerência de Projetos.

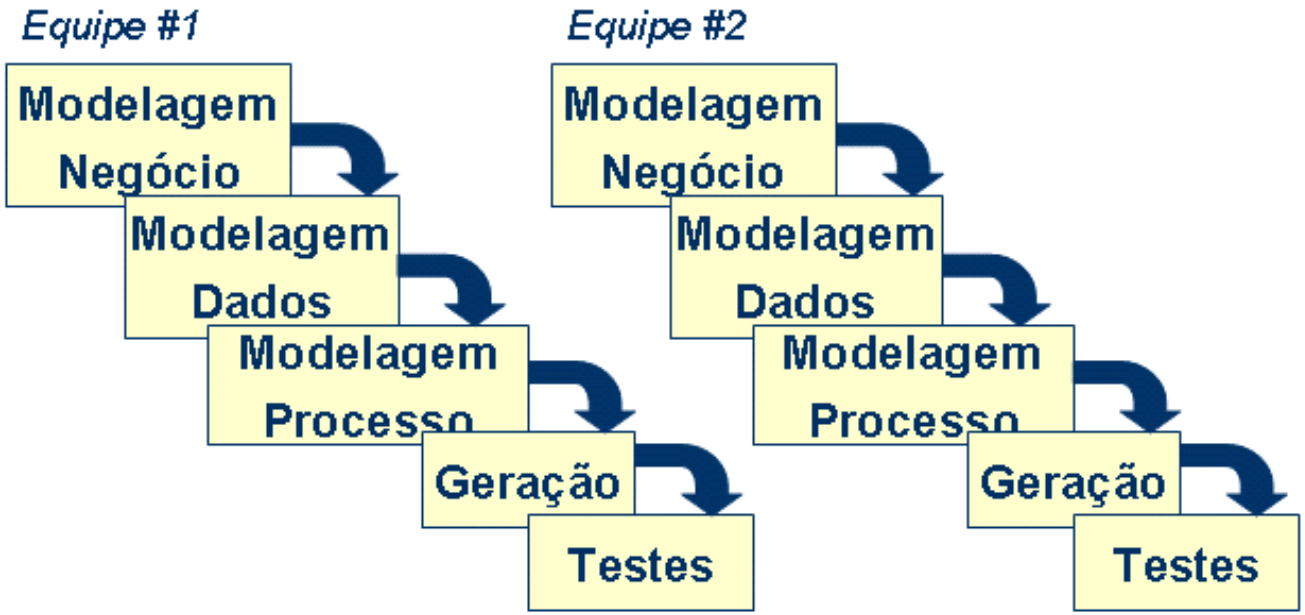


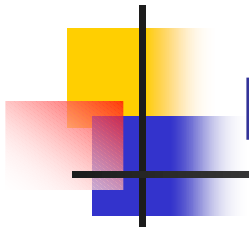
Modelo RAD (Rapid Application Development)

- Adaptação do modelo Cascata para “alta velocidade”.Para obter esse aumento de velocidade, a construção é baseada na divisão em componentes independentes.

- Cada equipe desenvolve uma funcionalidade independente, de forma paralela. Depois essas funções são integradas.

Modelo RAD
(Rapid Application Development)





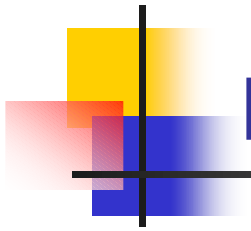
Problemas do Modelo RAD

- Apropriada apenas para projetos rápidos e modulares.

- Riscos técnicos elevados invalidam o RAD (novas tecnologias, integração com outros sistemas).

Mesmo assim...

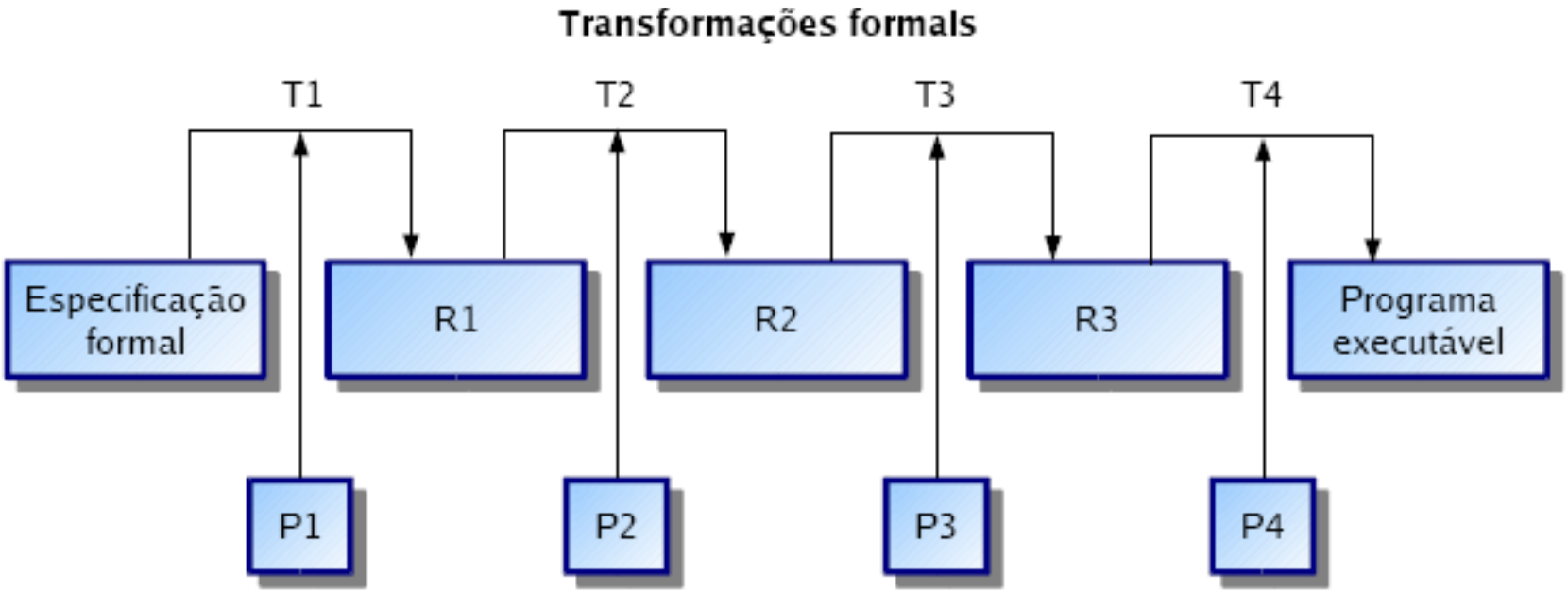
- Se uma aplicação comercial pode ser modularizada de modo a permitir que cada função principal possa ser completada em menos de três meses, é uma forte candidata a RAD.

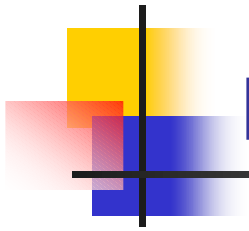


Modelo de Métodos Formais

- Envolve atividades que levam a uma especificação matemática do software.
- A especificação de requisitos, informal ou semi-formal, é redefinida em uma especificação formal detalhada, utilizando uma notação matemática.
- As atividades de design, implementação e testes são substituídas por um processo transformacional.
- Essas transformações devem ser matematicamente corretas, baseadas em provas-de-correção.
- É adequado a sistemas com exigências rigorosas de segurança, confiança e garantia.

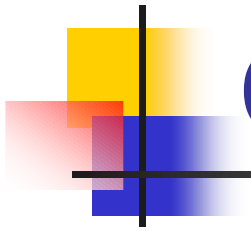
Modelo de Métodos Formais





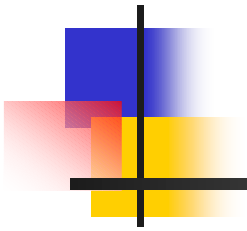
Problemas do Modelo de Métodos Formais

- As provas são longas e impraticáveis para sistemas de grande porte.
- Necessidade de habilidades especiais e treinamento para aplicar a técnica
- Dificuldade de especificar formalmente alguns aspectos do sistema, tais como a interface com o usuário



Conclusão

- Software é elemento chave para o sucesso. Mas:
  - Software não é hardware.
  - Software não é fácil.
  - Software mata.
  - Precisamos de ajuda.



Engenharia de Software

Prof. Grinaldo Oliveira

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas