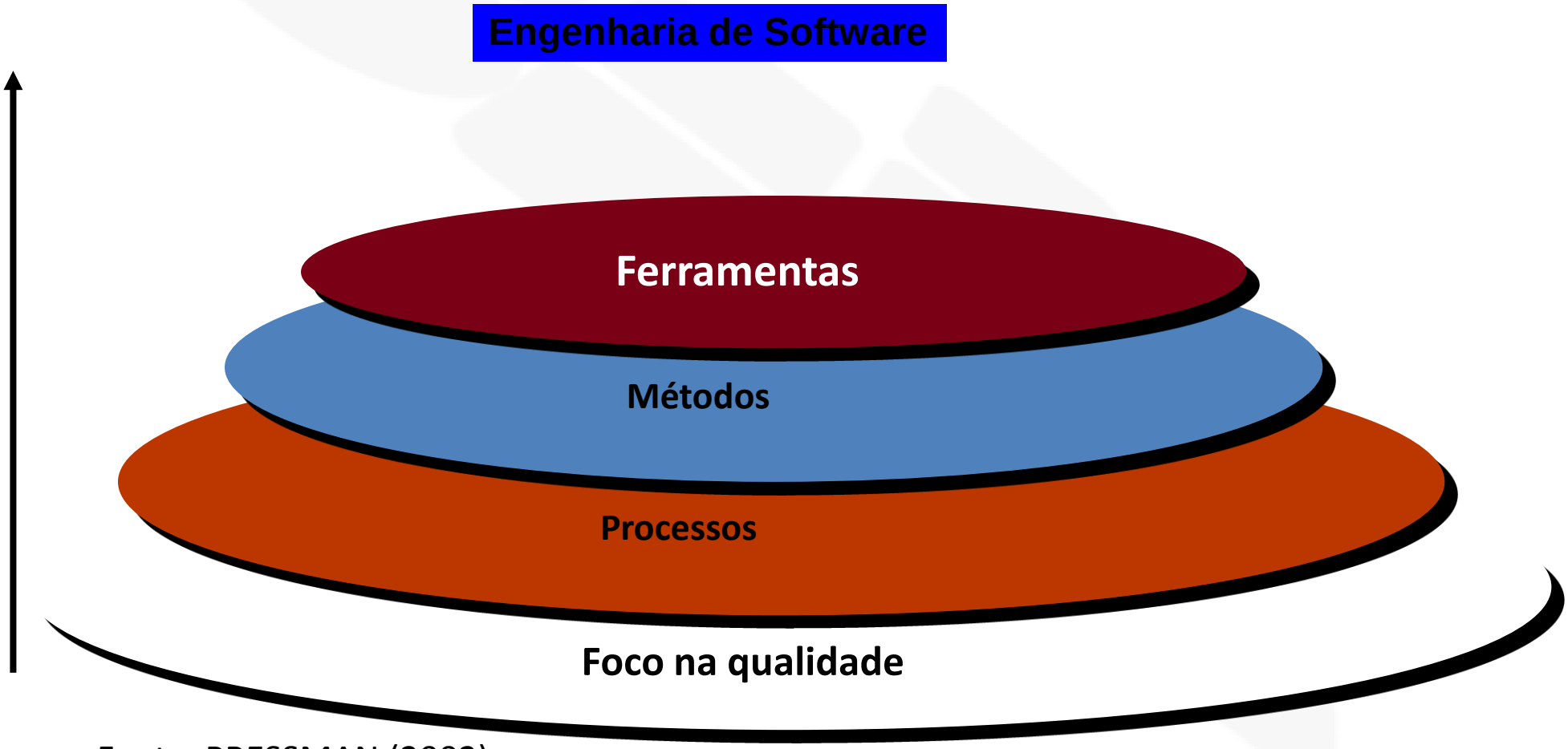
INF014 – Análise e Projeto de Sistemas

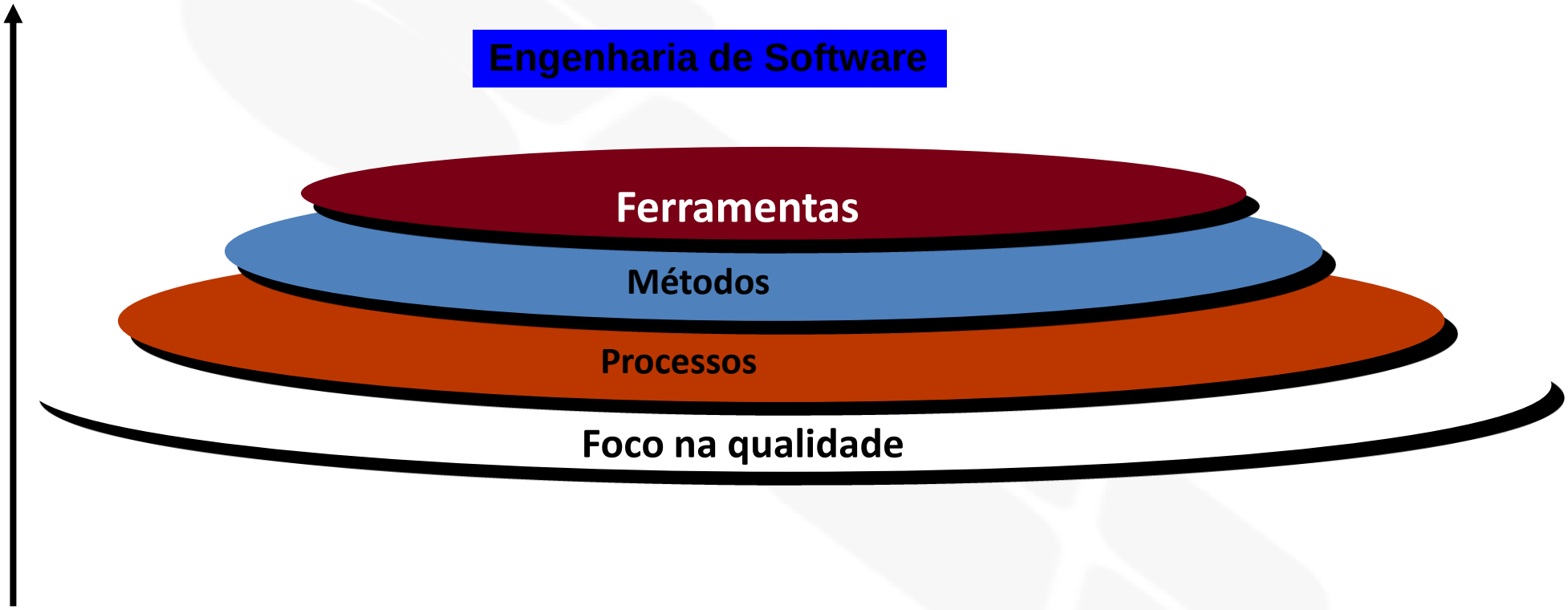
Ciclos de vida e Processos de Software

Maurício Pitangueira
antoniomauricio@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas



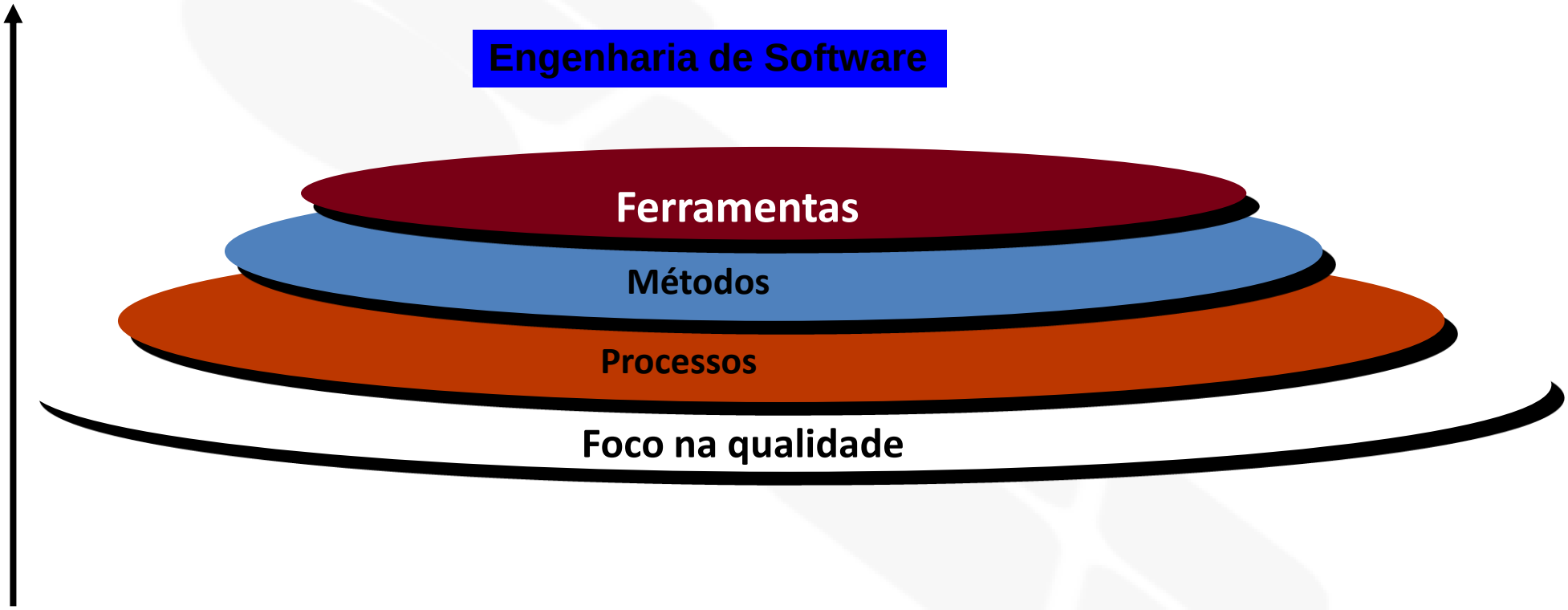
Fonte: PRESSMAN (2002)



Foco na qualidade: cultura de um processo contínuo de aperfeiçoamento - desenvolvimento de abordagens cada vez mais amadurecidas para a ESW

Processos: permite o desenvolvimento racional e oportuno de software para computador. Define áreas-chaves(base p/ o controle gerencial de projetos) - produtos de trabalho, marcos, etc..

Fonte: PRESSMAN (2002)



Métodos de Engenharia: fornecem a técnica de como fazer para construir software.

ferramentas: fornecem apoio automatizado ou semi-automatizado para o processo e para os métodos

Fonte: PRESSMAN (2002)

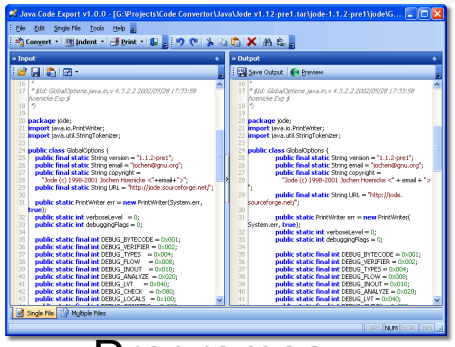
- Engenharia: análise, projeto, construção, verificação e a gestão de elementos técnicos
  - Qual o problema a ser resolvido?
  - Que características do elemento são usadas para resolver o problema?
  - Como o elemento (e a solução) serão realizados?
  - Como o elemento vai ser construído?
  - Que abordagem será usada para descobrir erros que foram cometidos no projeto e na construção do elemento?
  - Como o elemento será mantido a longo prazo, quando correções, adaptações e aperfeiçoamentos forem solicitados pelos usuários?

- Faz-se a engenharia de software aplicando-se as três fases distintas, que focalizam a definição, o desenvolvimento e a manutenção
- **DEFINIÇÃO:** SE CONCENTRA NO QUÊ
  - que informação deve ser processada
  - Que função e desempenho são desejados
  - Que comportamento deve ser esperado do sistema
  - Que interfaces devem ser estabelecidas
  - Que restrições de projeto existem
  - Que critérios de validação são necessários para definir um sistema bem-sucedido

- Faz-se a engenharia de software aplicando-se as três fases distintas, que focalizam a definição, o desenvolvimento e a manutenção
- **DESENVOLVIMENTO:** FOCALIZA O COMO
  - Definir como os dados devem ser estruturados
  - Como a função deve ser implementada dentro da arquitetura do sw
  - Como os detalhes procedimentais devem ser implementados
  - Como as interfaces devem ser caracterizadas
  - Como o projeto deve ser traduzido em uma linguagem de programação
  - Como o teste vai ser realizado

- Faz-se a engenharia de software aplicando-se as três fases distintas, que focalizam a definição, o desenvolvimento e a manutenção
- **MANUTENÇÃO:**
  - FOCALIZA AS MODIFICAÇÕES ASSOCIADAS COM A CORREÇÃO DE ERROS, AS ADAPTAÇÕES NECESSÁRIAS, À MEDIDA QUE O AMBIENTE DE SW EVOLUI. TIPOS:
    - Correção modifica o sw para corrigir defeitos
    - Adaptação resulta em modificações no sw para acomodar mudanças no seu ambiente externo
    - Aperfeiçoamento aprimora o sw além dos requisitos funcionais originais
    - Prevenção faz modificações nos programas, de modo que possam ser mais facilmente corrigidos, adaptados e melhorados

- Acompanhamento e controle de projeto de sw
- Revisões técnicas formais
- Garantia de qualidade de sw
- Gestão de configuração de sw
- Preparação e produção de documentos
- Gestão de reutilização
- Medição
- Gestão de risco

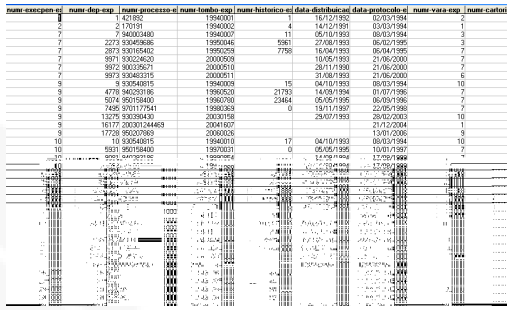


Programas

+

Documentação

+



Dados

Simplesmente
“FAZER”

OU

ENGENHARIA DE SOFTWARE

www.sei.cmu.edu/
www.rspa.com/spi/
www.swebok.org

[redacted]

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Processo de Desenvolvimento de Software

Atividades

- Garantia de qualidade;
- Gerência de Configuração;
- Gerência de Riscos;
- Métricas;
- Estimativas;
- Revisões Técnicas Formais.

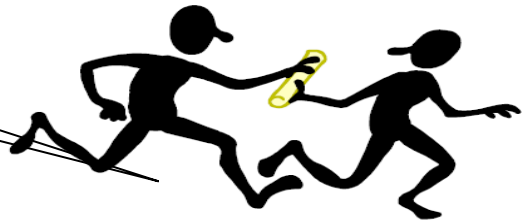
Outros Processos Contidos no Processo Principal

- !)
- Un oftware de ati eva à proc n produto de so
- b b!d jb! n ef!

! fn!f p b!p!

Motivação

Processo bom, resultado bom!



Processo ruim, resultado ruim!

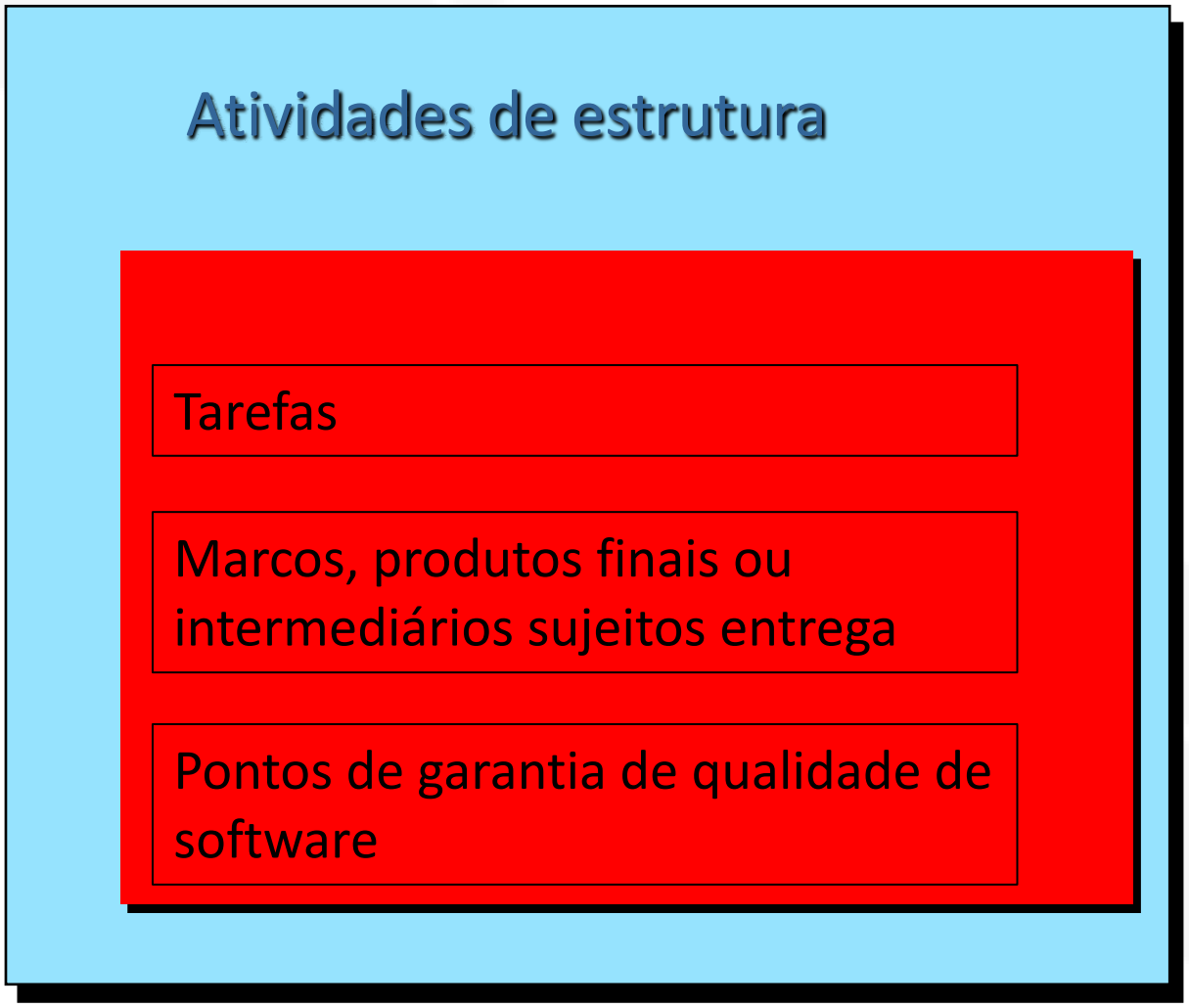
- Embora existam muitos processos de software diferentes, algumas atividades fundamentais são comuns a todos eles, como:
  - Especificação de software: a funcionalidade e as restrições sobre sua operação devem ser definidas
  - Projeto e Implementação de software: o software que atenda à especificação deve ser produzido
  - Validação de software; o sw deve ser validado para garantir que ele faça o que o cliente deseja
  - Evolução de software: o sw deve evoluir para atender às necessidades mutáveis do cliente

Processo bom, resultado bom!

Processo ruim, resultado ruim!

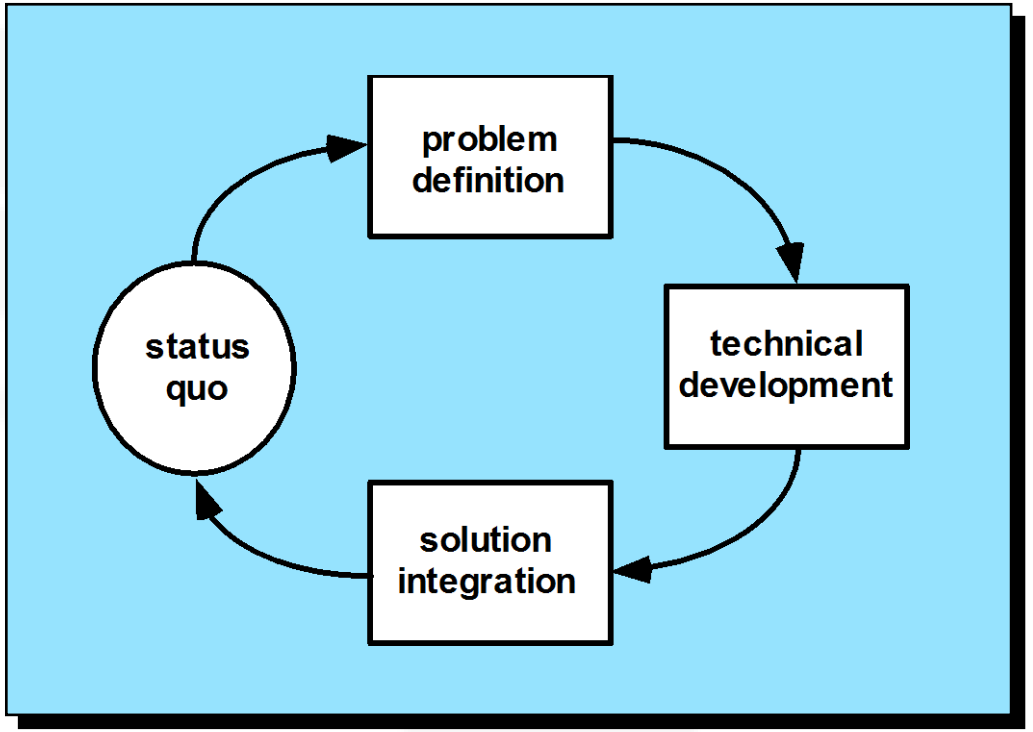
- O que é?
  - Série de passos previsíveis que o ajuda a criar a tempo um resultado de alta qualidade
- Quem faz
  - Os engenheiros de siftware e seus gerentes adaptam o processo a suas necessidades e depois o seguem
- Por que é importante?
  - Porque fornece estabilidade, controle e organização para uma atividade que pode, se deixada sem controle, tornar bastante caótica
- Qual é o produto do trabalho?
  - São os programas, documentos e dados produzidos em consequência das atividades de engenharia de software;
- Como garanto que fiz corretamente?
  - A qualidade, pontualidade e viabilidade são os melhores indicadores da eficácia do processo usado.

- 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

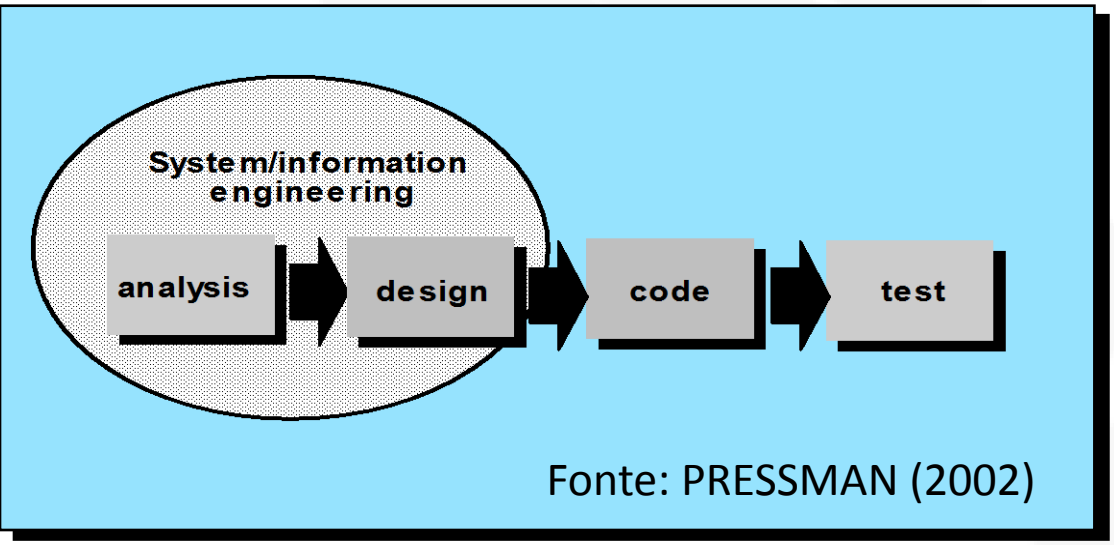

Atividades de estrutura
Tarefas
Marcos, produtos finais ou intermediários sujeitos entrega
Pontos de garantia de qualidade de software

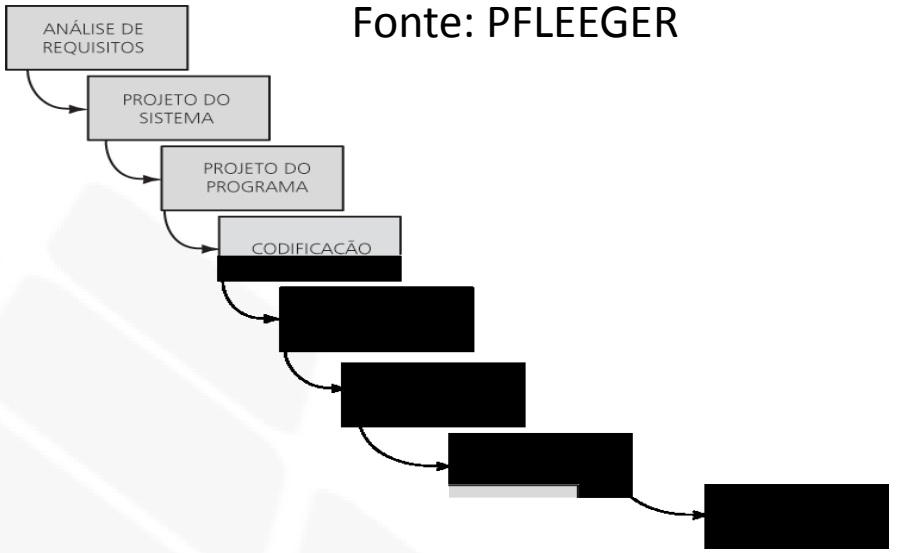
- **Modelo de processo de software** é uma representação abstrata de um processo de software
- Estratégia de desenvolvimento que abrange as camadas de processo, métodos e ferramentas.
- É escolhido com base na natureza do projeto e da aplicação, nos métodos e ferramentas a serem usados, e nos controles e nos produtos intermediários e finais que são requeridos.
- Pode ser encarado como um ciclo de solução de problema
- Razões p/ se modelar um processo
  - Formar um entendimento comum/ encontrar inconsistências, redundâncias e omissões/ encontrar e avaliar propostas mais adequadas aos objetivos

- Alguns modelos :
  - Seqüencial linear/ Prototipagem/ RAD/ Evolucionário(incremental, espiral e espiral ganha-ganha)/ desenvolvimento concorrente/ baseado em componentes



Fonte: PRESSMAN (2002)



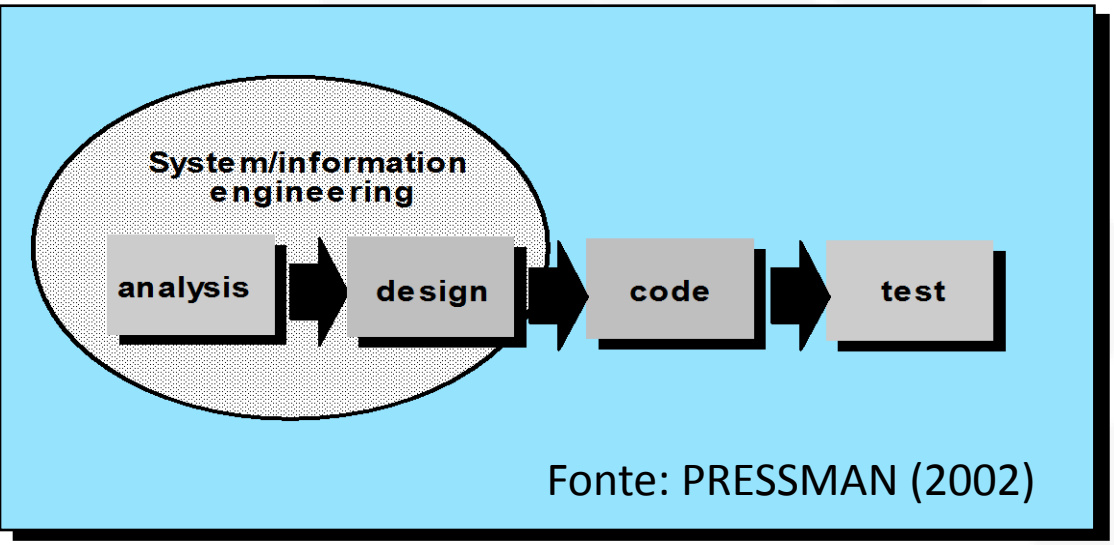


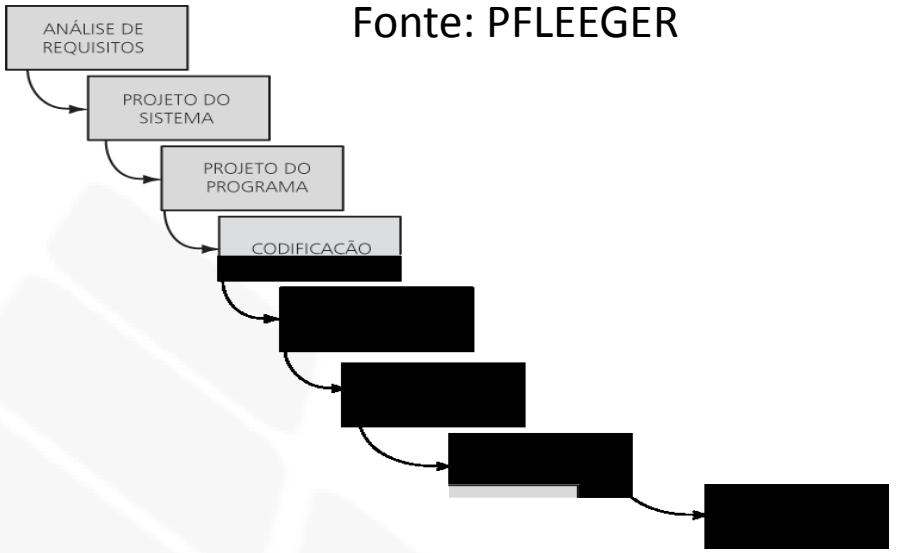
Fonte: PFLEEGER

- Aspectos do modelo seqüencial
  - **Análise**: visão essencial quando o software deve fazer interface com outros elementos (hardware, pessoas e banco de dados. Envolve a coleta de requisitos em nível do sistema, pequena quantidade de projeto e análise de alto nível.

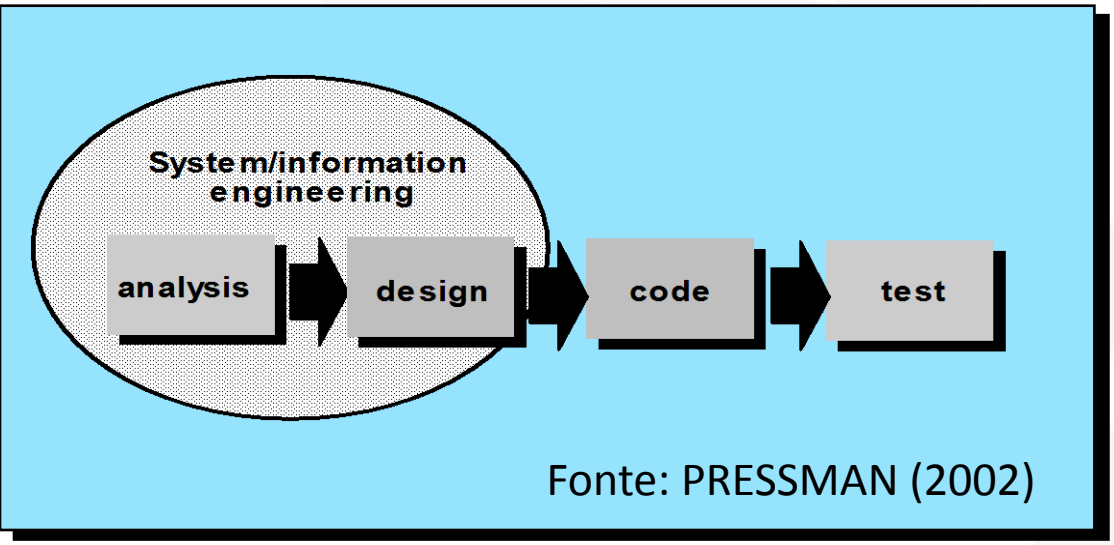
O processo de definição de requisitos é intensificado e focalizado especificamente no software. Aqui. O engenheiro de sw deve conhecer o domínio da informação do sw tanto quanto a função necessária, o comportamento, o desempenho e a interface.

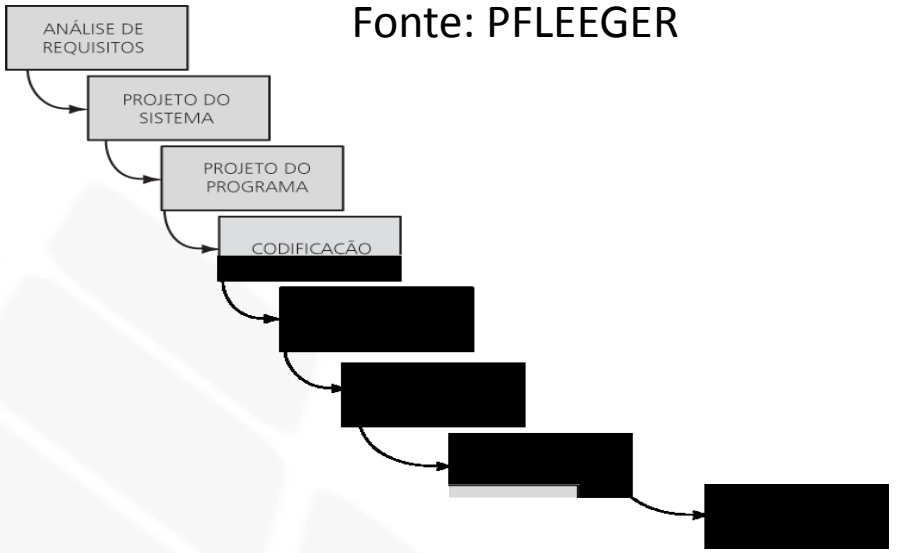
Os documentos do sistema são documentados e revistos com o cliente.





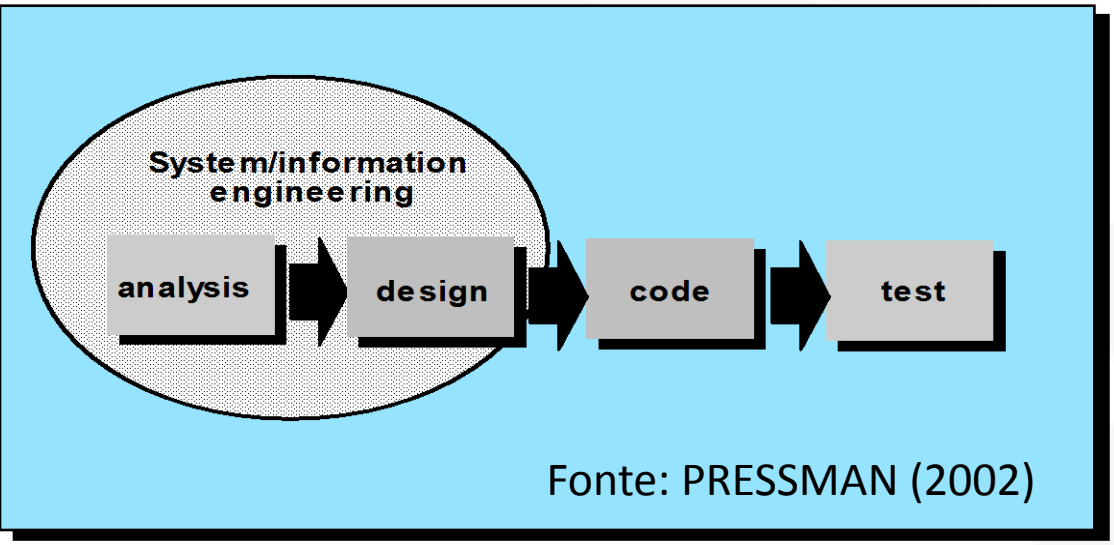
- Aspectos do modelo seqüencial
  - **Projeto**: tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie. Se concentra em 4 atributos do programa:
    - Estrutura de dados;
    - Arquitetura de software;
    - Detalhes procedimentais(algorítmicos)
    - Representações da interface



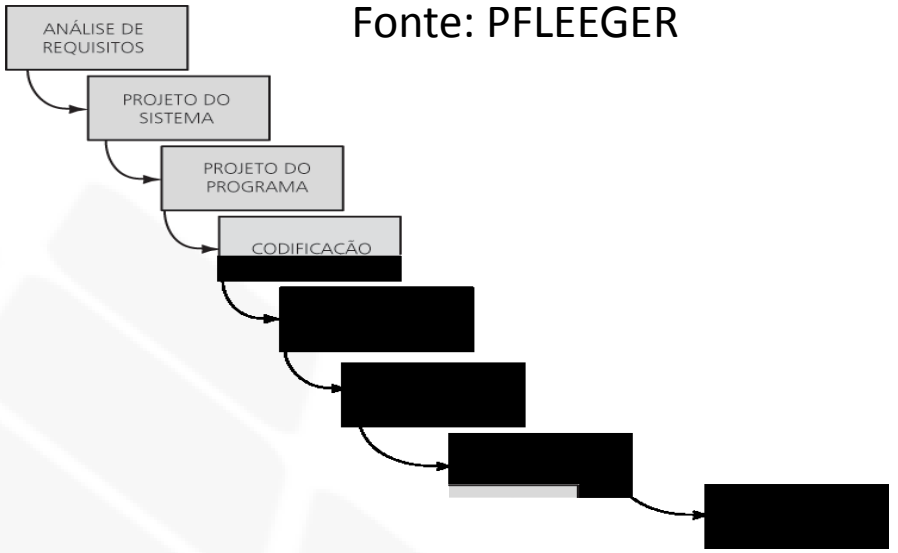


Fonte: PFLEEGER

- Aspectos do modelo seqüencial
  - **Codificação** tradução das representações do projeto para uma linguagem "artificial" resultando em instruções executáveis pelo computador

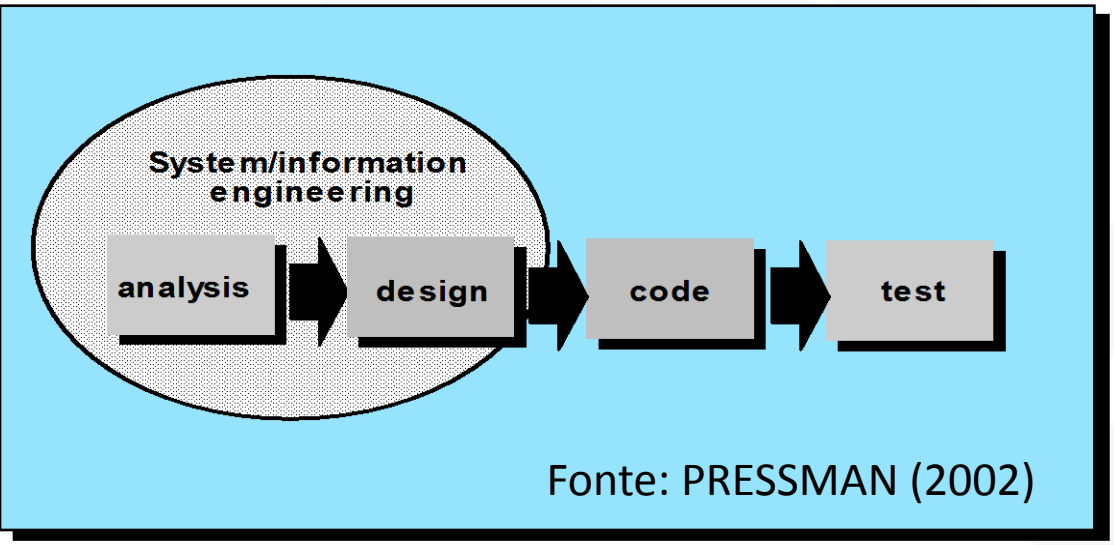


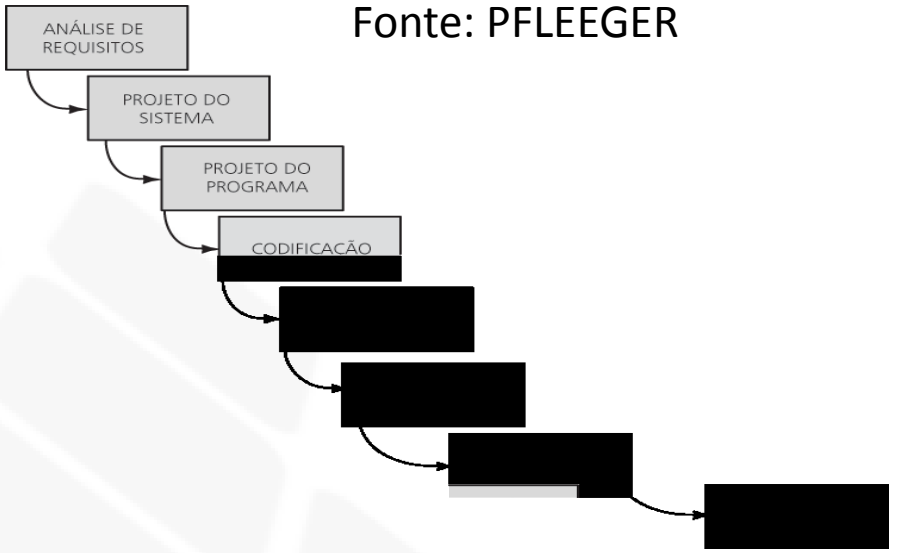
Fonte: PRESSMAN (2002)



Fonte: PFLEEGER

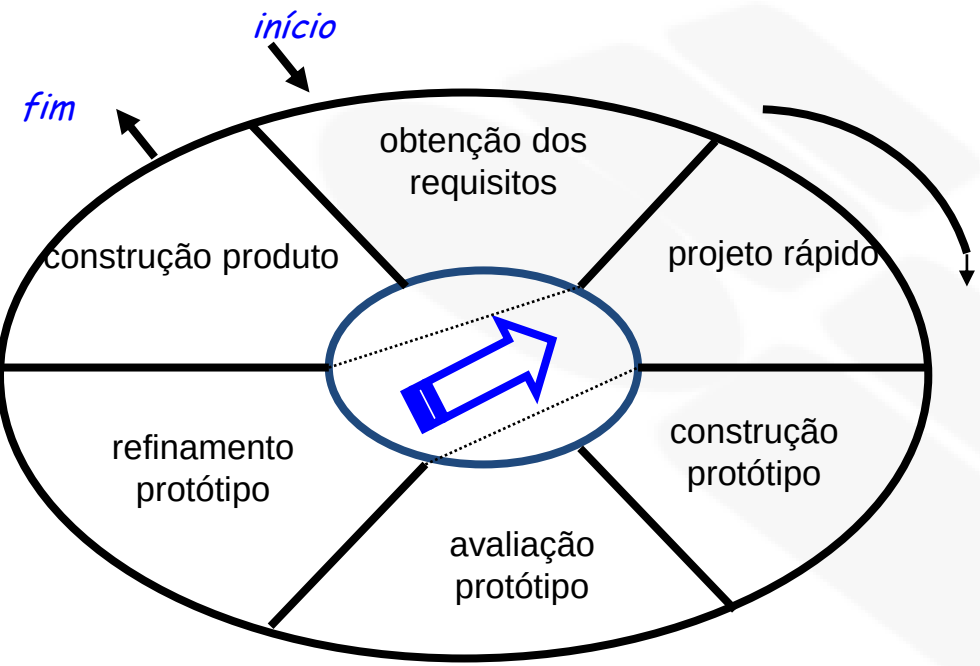
- Aspectos do modelo seqüencial
  - **Testes:** concentram-se nos aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas e nos aspectos funcionais externos, para descobrir erros e garantir que a entrada definida produza resultados que concordem com os esperados

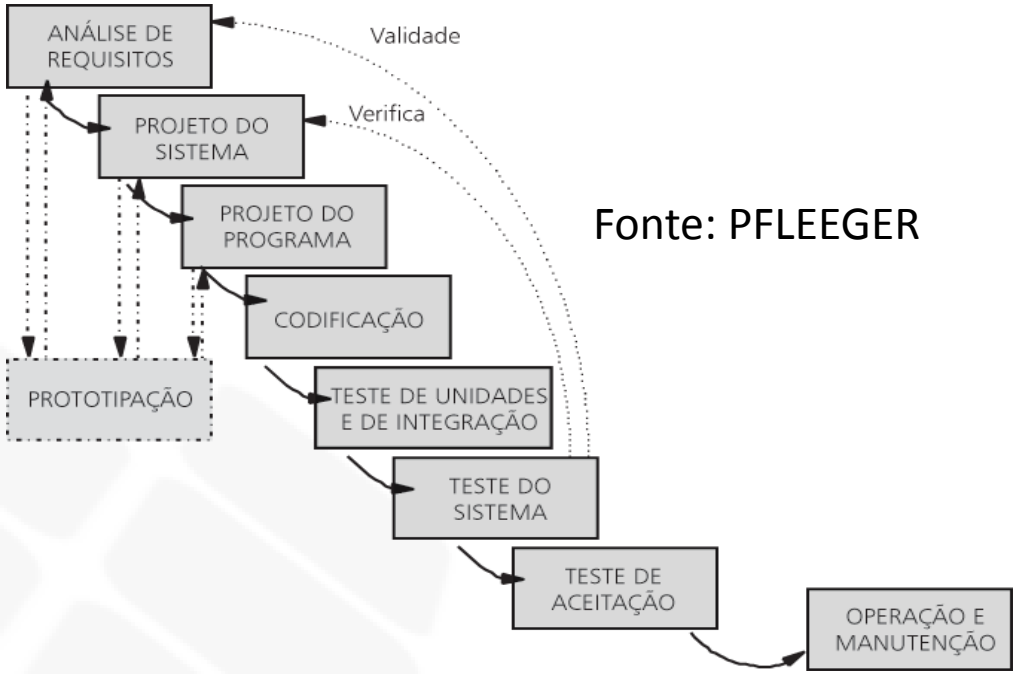




Fonte: PFLEEGER

- Aspectos do modelo seqüencial (-)
  - modificações podem causar confusão à medida que a equipe de projeto prossegue
  - projetos reais raramente seguem o fluxo seqüencial que o modelo propõe
  - Difícil p/ o cliente estabelecer todos os requisitos explicitamente (incerteza natural no começo)
  - Cliente precisa ter paciência
  - Leva a **estados de bloqueio** (membros da equipe esperam outros membros completarem suas tarefas)





Fonte: PFLEEGER

Obtenção dos requisitos: desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais

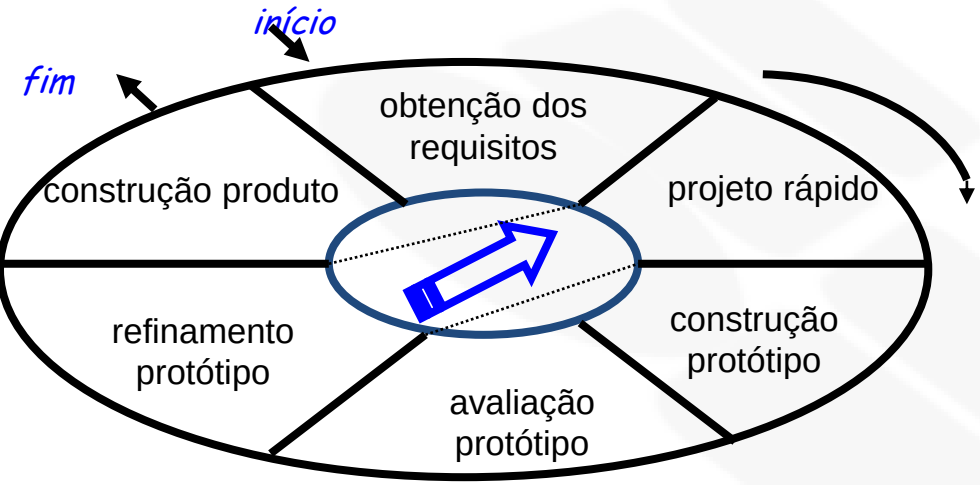
Projeto rápido: representação dos aspectos do software que são visíveis ao usuário (abordagens de entrada e formatos de saída)

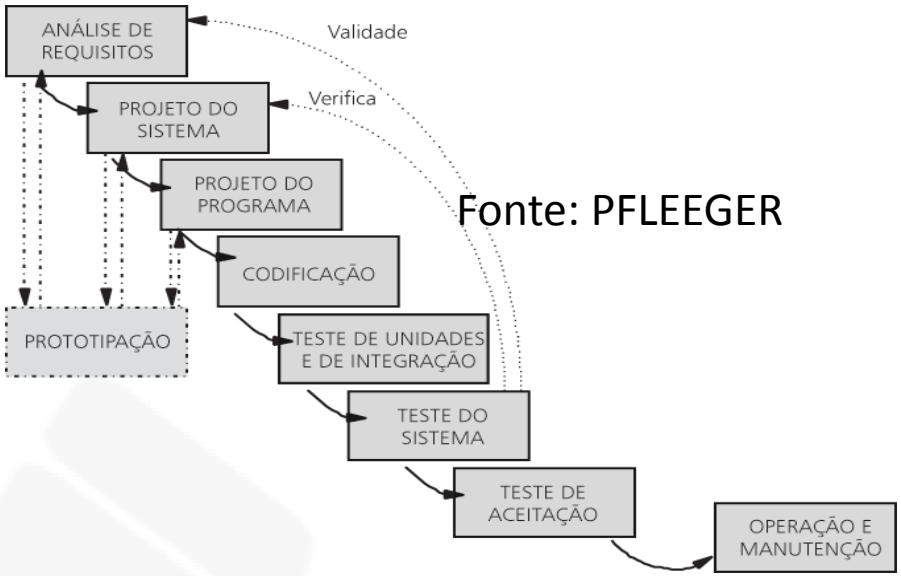
Construção protótipo: implementação do projeto rápido

Avaliação do protótipo: cliente e desenvolvedor avaliam o protótipo

Refinamento do protótipo

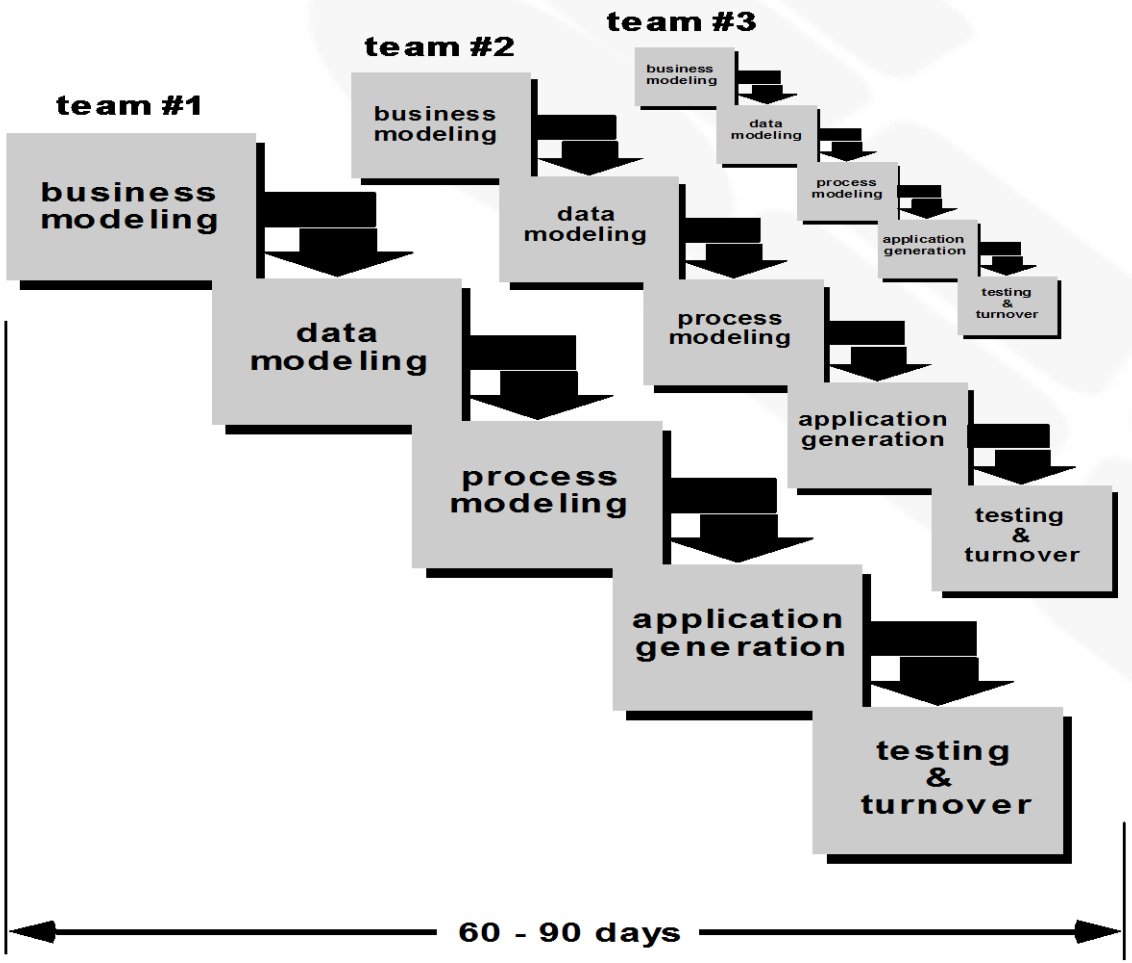
iteração





Fonte: PFLEEGER

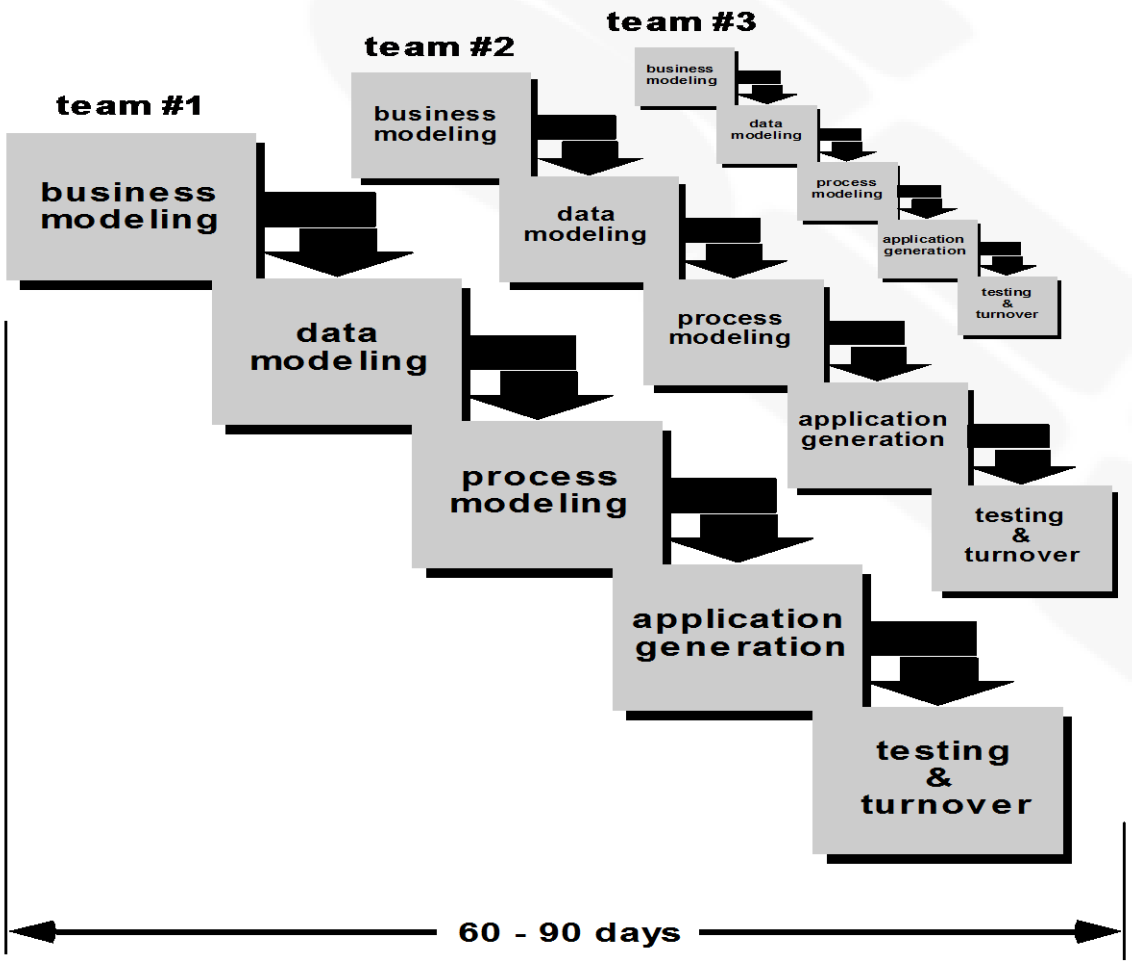
- Aspectos do modelo de prototipagem
  - (+) identificar mais detalhadamente requisitos de entrada, processamento e saída
  - (+) Enfrentar a insegurança na eficiência de um algoritmo, da adaptabilidade de um SO ou da interface
  - Idealmente, o protótipo serve como um mecanismo para a identificação dos requisitos de software
  - (-) O cliente vê o que parece ser uma versão executável do sw e ignora outros aspectos
  - (-) O desenvolvedor freqüentemente faz concessões na implementação a fim de conseguir rapidamente um protótipo executável (quais os impactos?)
  - OBS: resista a pressão para aperfeiçoar um protótipo mal feito dentro de uma linha de produção. O resultado quase sempre é de baixa qualidade.



- É uma adaptação de alta velocidade do modelo sequencial linear, no qual o desenvolvimento rápido é conseguido no uso de construção baseada em componentes
- Se os requisitos são bem compreendidos e o objetivo do projeto é restrito, o processo RAD permite a uma equipe de desenvolvimento criar um sistema plenamente fucnional

Fonte: PRESSMAN (2002)

RAD



Modelagem do negócio: o fluxo de informação entre as funções do negócio é modelado de forma a responder às seguintes questões:

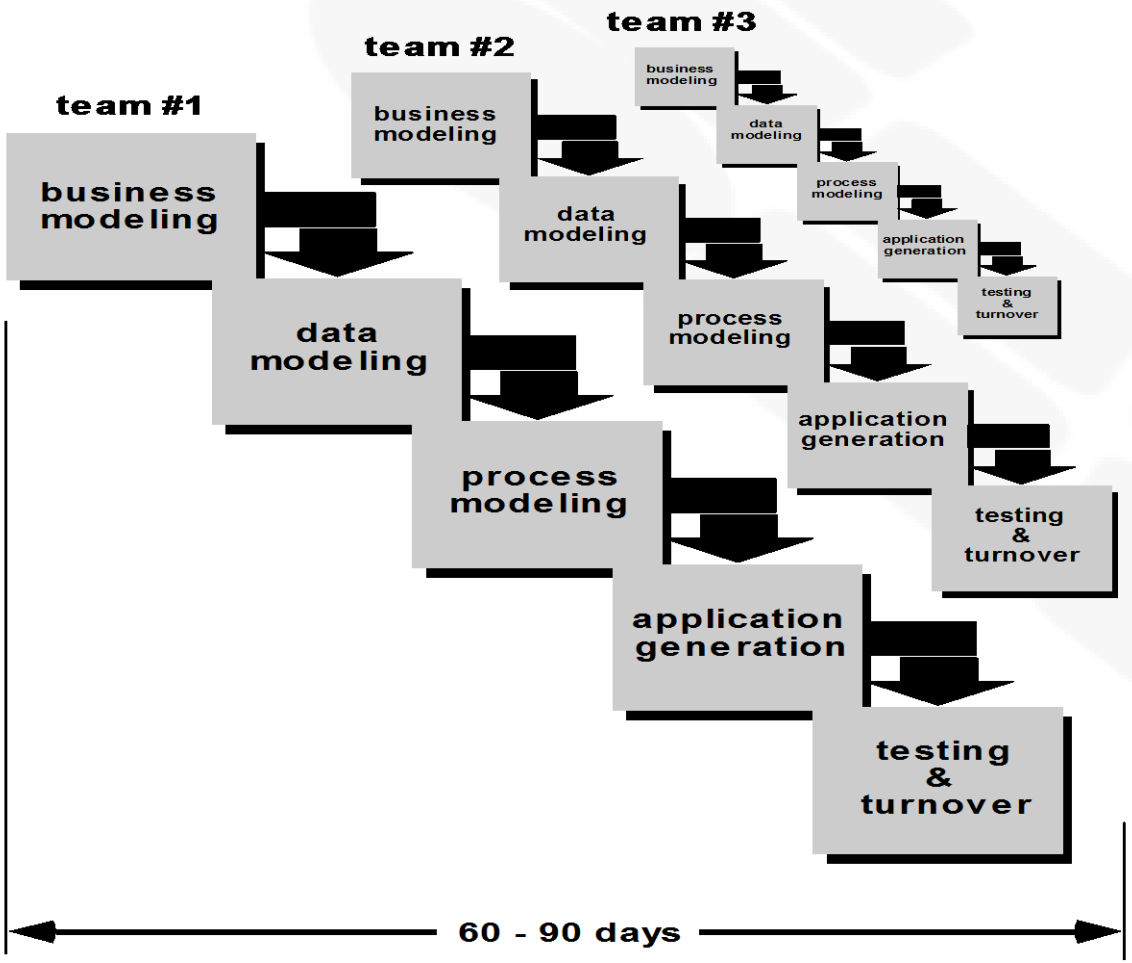
- Que informação dirige o processo de negócio?
- Que informação é gerada?
- Quem a gera?
- Para onde vai a informação?
- Quem a processa?

Modelagem dos dados: o fluxo de informação , definido como parte da fase de modelagem é refinado num conjunto de objeto de dados, que são necessários para dar suporte ao negócio

- As características de cada objeto (atributos) são identificadas e as relações entre esses objetos são definidas

Fonte: PRESSMAN (2002)

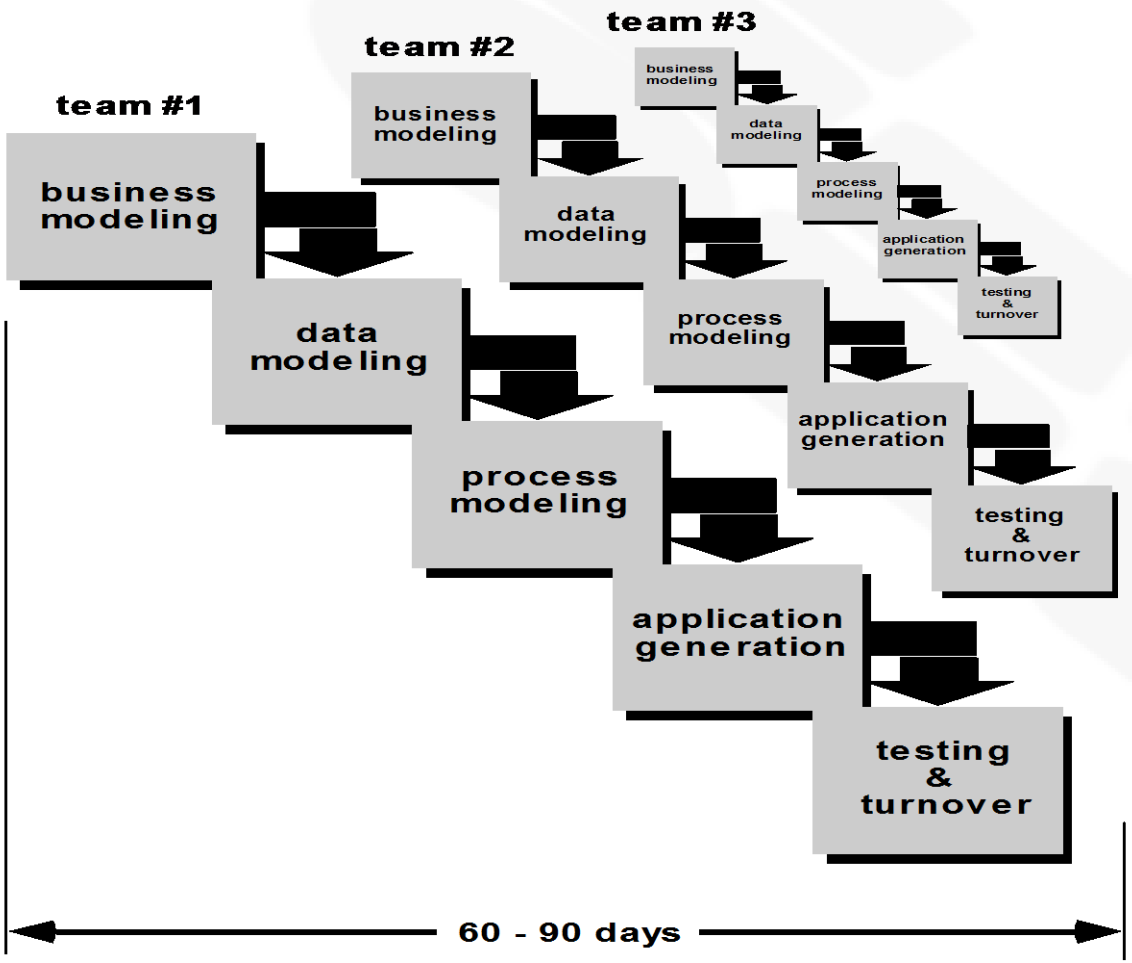
RAD



- **Modelagem do processo:** objetos da fase anterior são transformados para conseguir o fluxo de informação necessário para implementar uma função do negócio.
  - Descrições do processamento são criadas para adicionar, modificar, descartar ou recuperar um objeto de dados.
- **Geração da aplicação**: este processo trabalha para reusar componentes de programas existentes(quando possível) ou criar componentes reusáveis(quando necessário).
  - Ferramentas automatizadas são usadas para facilitar a construção do software
- **Teste e entrega**: os componentes novos devem ser testados e todas as interfaces devem ser exaustivamente exercitadas

Fonte: PRESSMAN (2002)

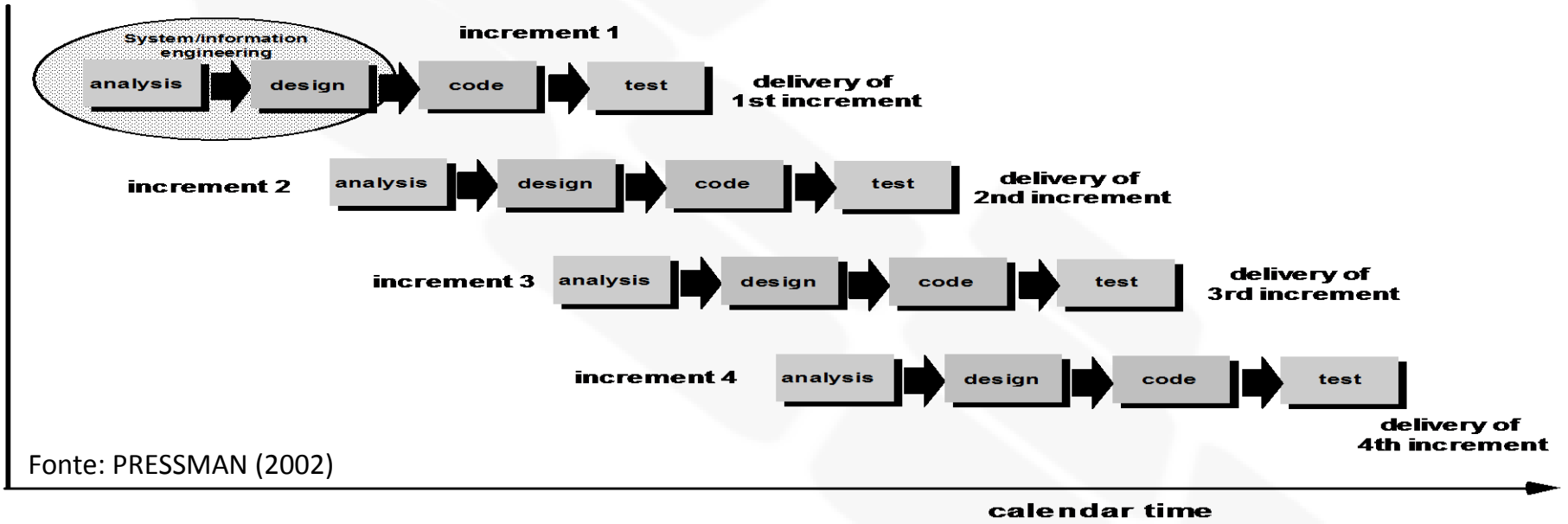
RAD



- Para projetos grandes, mas mensuráveis, O RAD exige rh suficientes para criar um nº adequado de equipes RAD
- Exige compromisso
- Se o sistema não puder ser adequadamente modularizado, a construção dos componentes, necessária para o RAD, será problemática
- Quando riscos técnicos forem elevados, O RAD não é adequado
  - Isso ocorre quando uma nova aplicação faz uso intenso de uma nova tecnologia ou quando o novo sw exige um alto grau de interoperabilidade com programas de computadores existentes

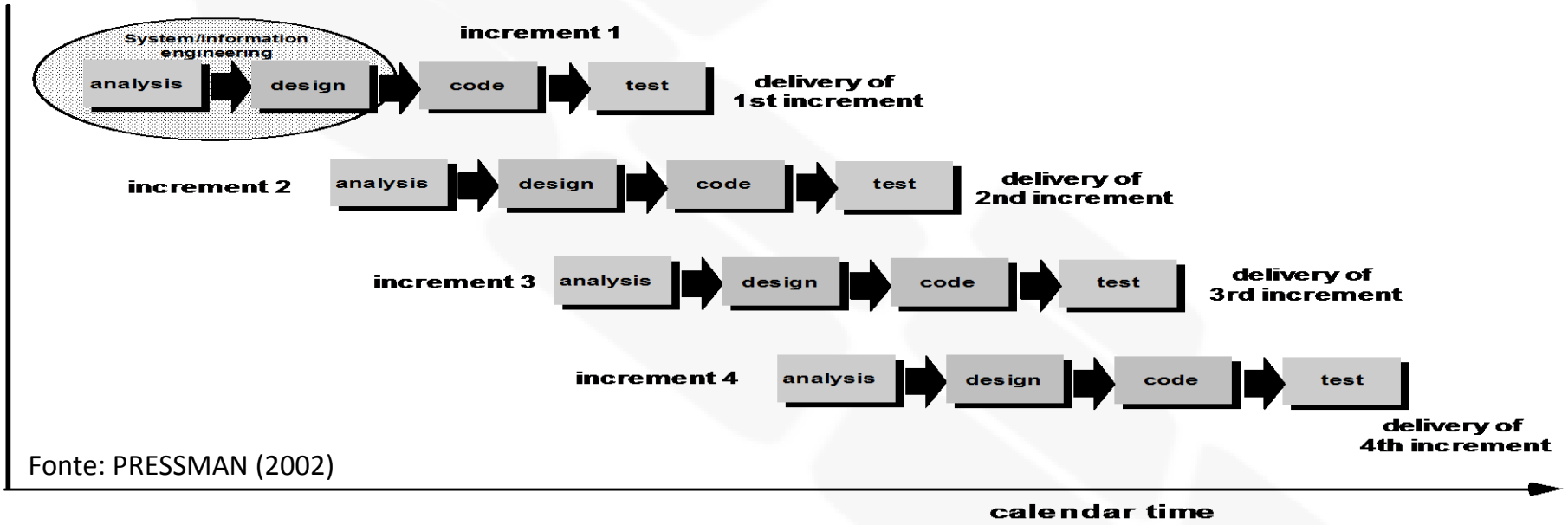
Fonte: PRESSMAN (2002)

RAD



Fonte: PRESSMAN (2002)

- Combina elementos do sequencial linear coma filosofia interativa da prototipagem
- Cada sequencial linear produz um “incremento” factível do sw
- Difrentemente da prototipagem, este modelo objetiva a eleboração de um produto operacional a cada incremento.



Fonte: PRESSMAN (2002)

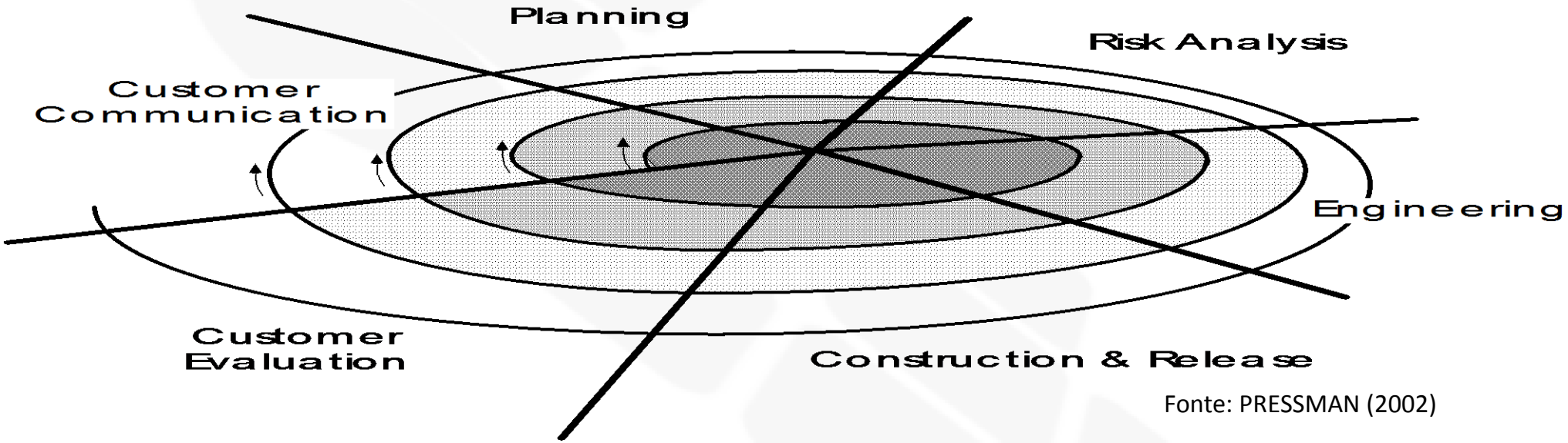
Núcleo do produto no 1º produto/

Elaboração de um produto operacional a cada incremento/

Incrementos acumulativos/

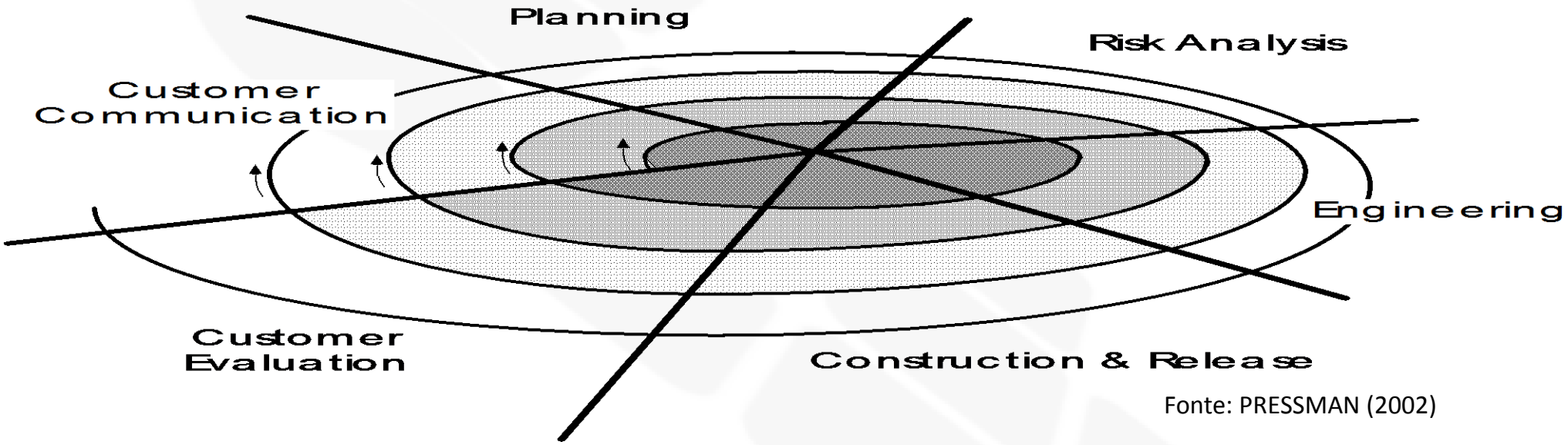
útil quando há mão-de-obra escassa/

incrementos podem ser planejados para gerir os riscos técnicos



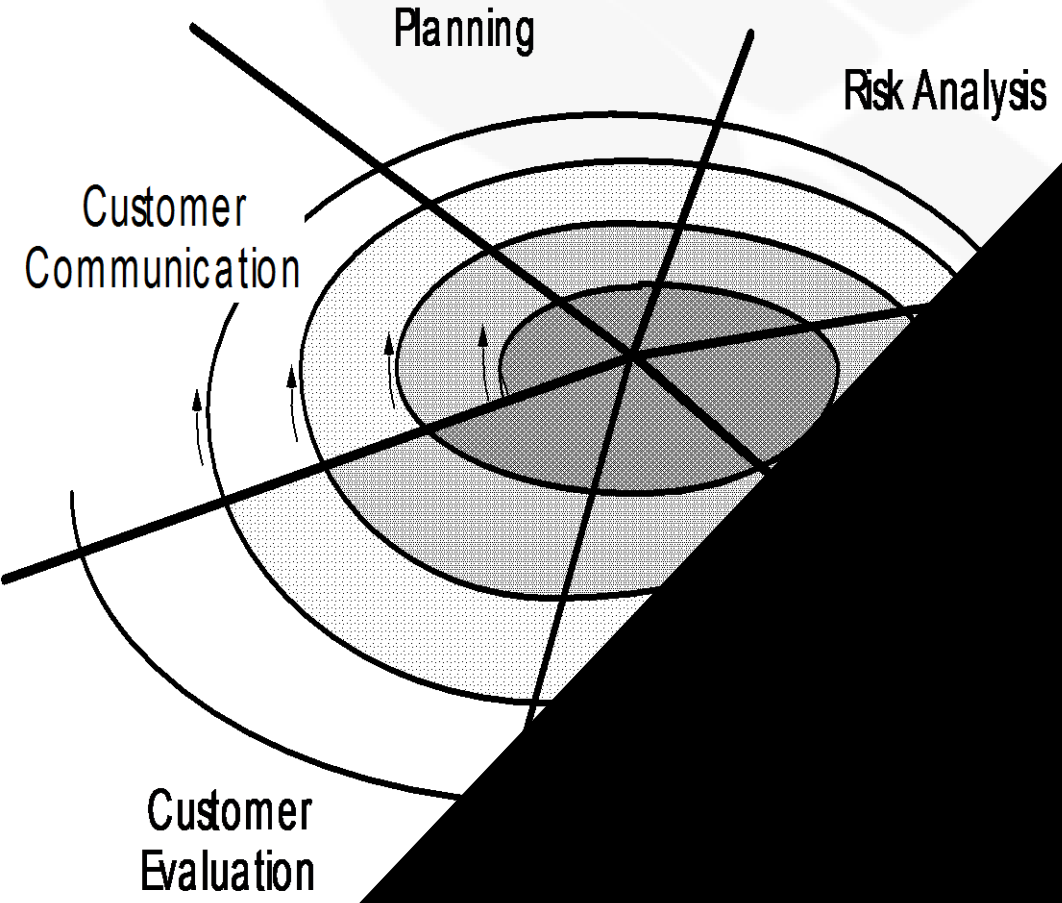
Fonte: PRESSMAN (2002)

- **Comunicação com o cliente**: tarefas necessárias para estabelecer efetiva comunicação entre o desenvolvedor e o cliente
- **Planejamento**: tarefas necessárias para definir recursos, prazos e outras informações relacionadas ao projeto
- **Análise de risco**: tarefas necessárias para avaliar os riscos, tanto técnicos quanto gerenciais
- **Engenharia**: tarefas necessárias para construir uma ou mais representações da aplicação
- **Construção e liberação**: tarefas necessárias para construir, testar, instalar e fornecer apoio ao usuário
- **Avaliação pelo cliente**

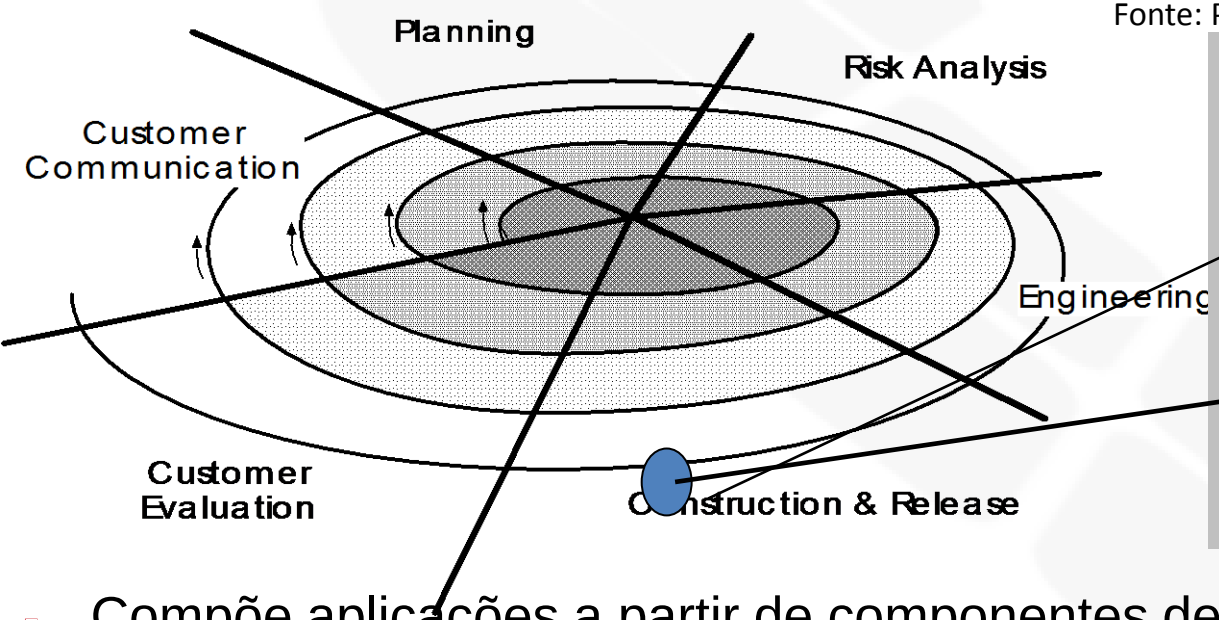


Fonte: PRESSMAN (2002)

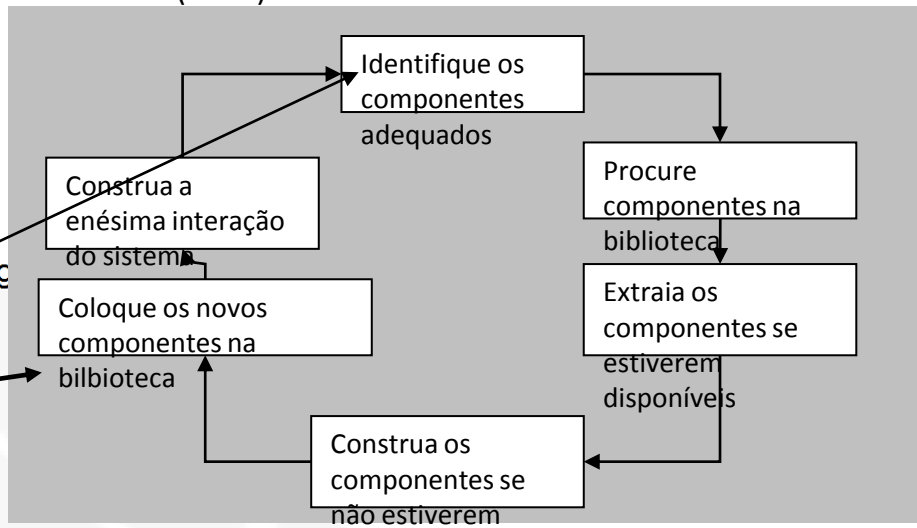
- Aspectos do modelo espiral
- Combina a natureza iterativa da prototipagem com os aspectos controlados e sistemáticos do linear
  - (+) potencial para o desenvolvimento rápido de versões incrementais do sw
  - (+) adaptado p/ aplicação ao longo da vida do sw
  - (-) pode ser difícil convencer os clientes (particularmente em situações de contrato) que a abordagem evolucionária é controlável
  - (-) exige competência considerável na avaliação de riscos e depende daquela p/ obter sucesso



Procure componentes na biblioteca



Fonte: PRESSMAN (2002)

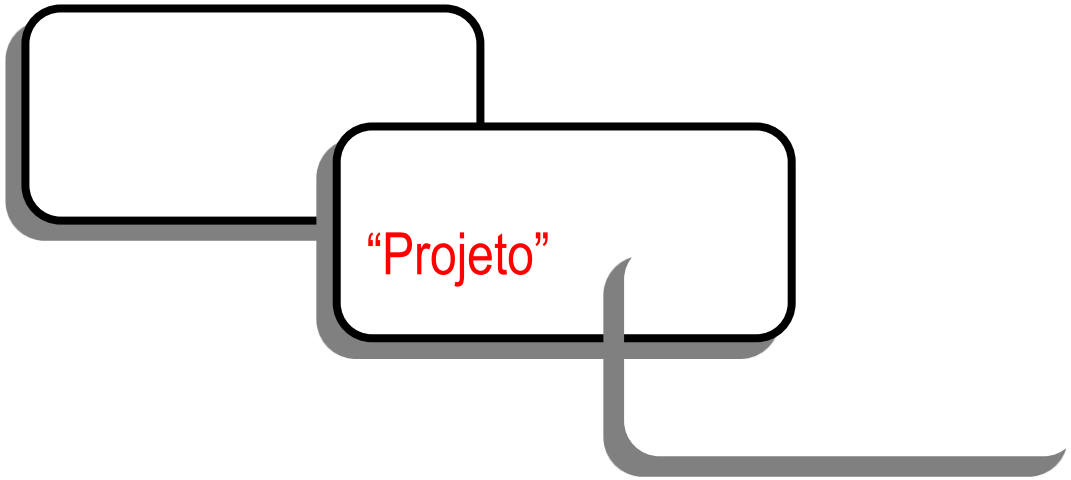


- Compõe aplicações a partir de componentes de software previamente preparados
- Leva ao reuso de sw e isso fornece aos engenheiros um certo nº de benefícios mensuráveis
- A montagem de componentes leva a uma redução de 70% no prazo do ciclo de desenvolvimento; uma redução de 84% no custo do projeto e um índice de produtividade de 26,2, comparado com o padrão de 16,9 para a indústria.

Concentra-se na capacidade de se especificar o software a uma máquina em um nível que esteja próximo à linguagem natural.

Engloba um conjunto de ferramentas de software que possibilitam que:

- o sistema seja especificado em uma linguagem de alto nível e
- o código fonte seja gerado automaticamente a partir dessas especificações

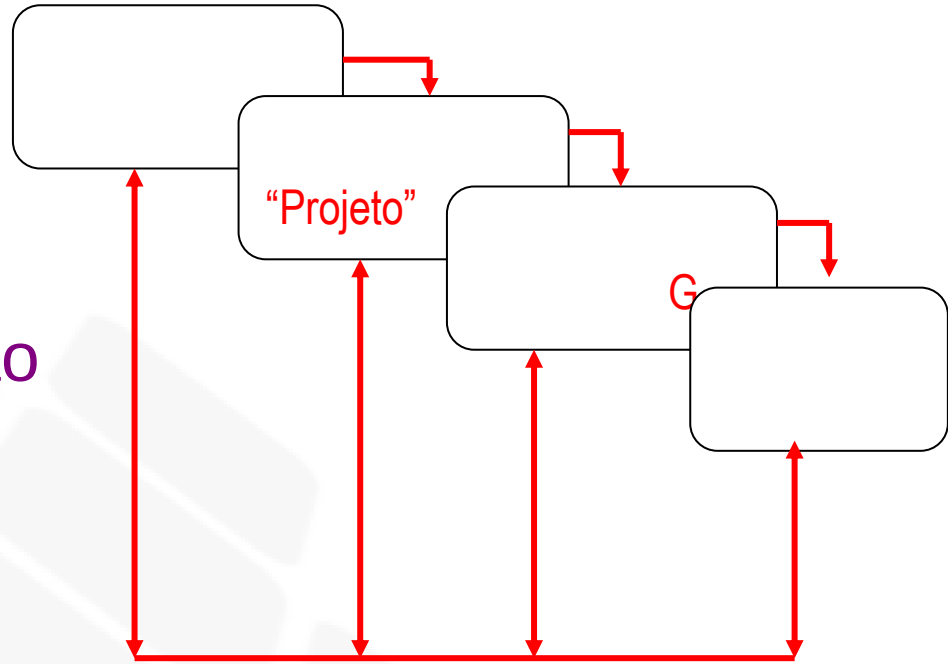

“Projeto”

O ambiente de desenvolvimento de software que sustenta o ciclo de vida de 4ª geração inclui as ferramentas:

- *linguagens não procedimentais para consulta de banco de dados*
- *geração de relatórios*
- *manipulação de dados*
- *interação e definição de telas*
- *geração de códigos*
- *capacidade gráfica de alto nível*
- *capacidade de planilhas eletrônicas*

1. **obtenção dos Requisitos:** o cliente descreve os requisitos os quais são traduzidos para um protótipo operacional

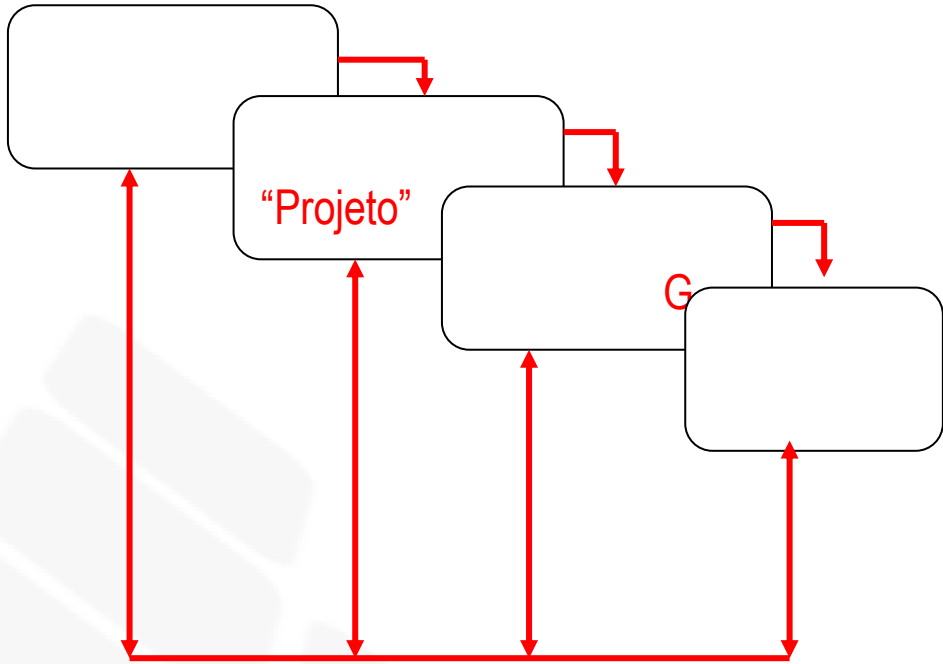
2. estratégia de "Projeto": para pequenas aplicações é possível mover-se do passo de Obtenção dos Requisitos para o passo de Implementação usando uma *Linguagem de 4G*



para grandes projetos é necessário desenvolver uma estratégia de projeto. De outro modo ocorrerão os mesmos problemas encontrados quando se usa abordagem convencional (baixa qualidade)

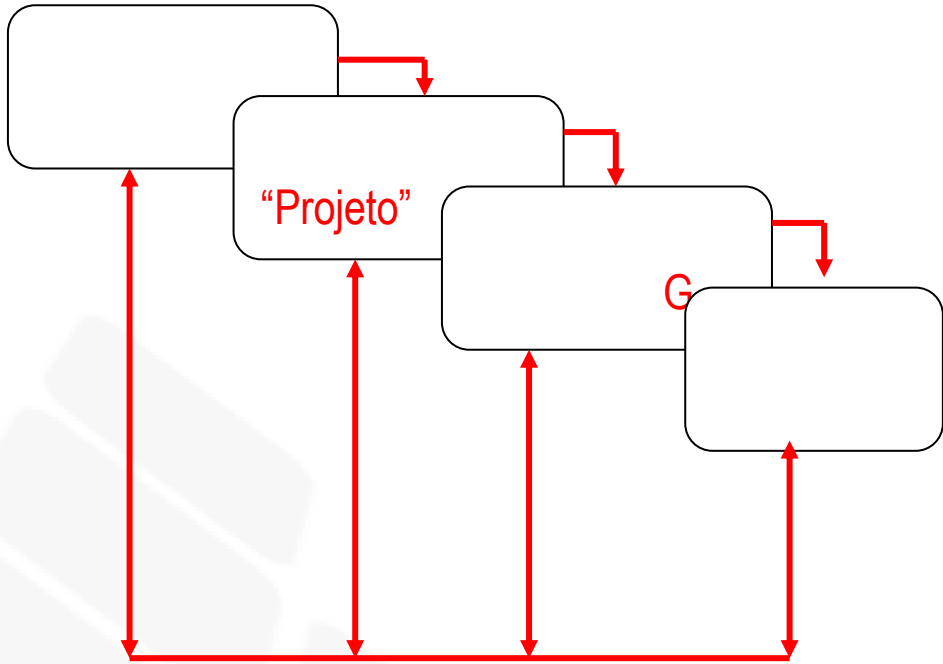
Atividades das Técnicas de 4ª Geração

3. implementação usando 4GL: os resultados desejados são representados de modo que haja geração automática de código . Deve existir uma estrutura de dados com informações relevantes e que seja acessível pela 4GL



Atividades das Técnicas de 4ª Geração

4. teste: o desenvolvedor deve efetuar testes e desenvolver uma documentação significativa. O software desenvolvido deve ser construído de maneira que a manutenção possa ser efetuada prontamente.



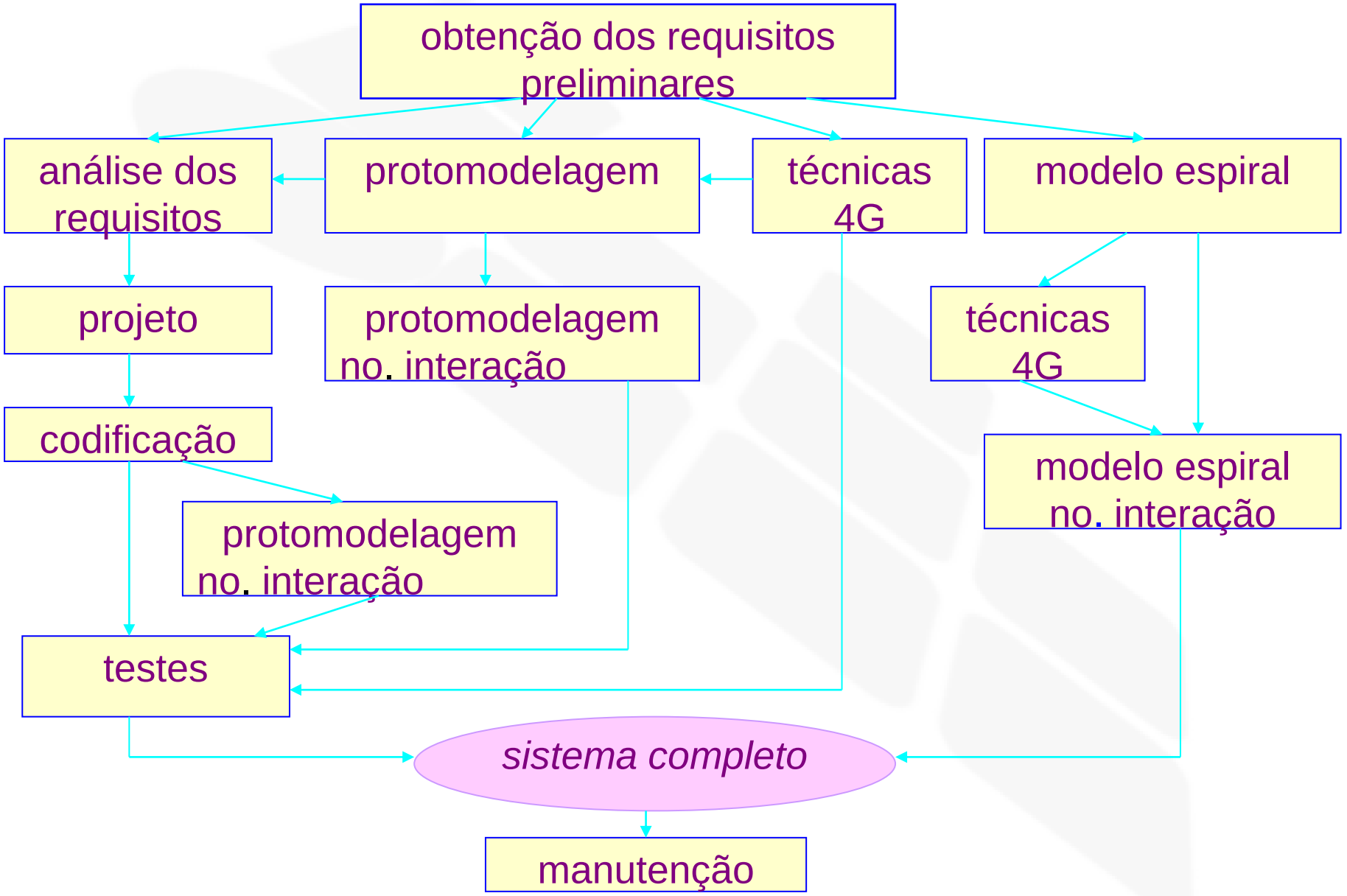
Técnicas de 4ª Geração (comentários)

PROPONENTES: redução dramática no tempo de desenvolvimento do software (aumento de produtividade)

OPONENTES: as 4GL atuais não são mais fáceis de usar do que as linguagens de programação

o código fonte produzido é ineficiente

a manutenibilidade de sistemas usando técnicas 4G ainda é questionável


obtenção dos requisitos preliminares
análise dos requisitos
protomodelagem
técnicas 4G
modelo espiral
projeto
protomodelagem no. interação
técnicas 4G
codificação
modelo espiral no. interação
protomodelagem no. interação
testes
sistema completo
manutenção

INF014 – Análise e Projeto de Sistemas

Ciclos de vida e Processos de Software

Maurício Pitangueira
antoniomauricio@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas