

# Engenharia de Software

## Seção 2 – Processos

# Objetivos

O aluno deverá identificar a necessidade e características do processo de software, e os principais modelos de processo em uso.



# Engenharia de Software

- ✓ A Engenharia de software é uma tecnologia estruturada em camadas.



# Foco na Qualidade

- ✓ Qualquer abordagem de engenharia precisa se embasar em um direcionamento organizacional para a qualidade.
- ✓ Filosofias como TQM (*Total Quality Management*) e Six Sigma incentivam uma cultura de melhoria contínua no processo.
- ✓ É esta cultura que leva a uma busca por estratégias cada vez mais eficientes para o desenvolvimento de software.

# Processo

- ✓ O alicerce da Engenharia de Software é a camada de **Processo**.
- ✓ O processo de Engenharia de Software mantém as camadas de tecnologia juntas e habilita um desenvolvimento racional de software.
- ✓ Ele é quem forma as bases para que as outras atividades necessárias ao desenvolvimento possam ocorrer:
  - ✓ Controle gerencial do desenvolvimento,
  - ✓ Como os métodos são aplicados,
  - ✓ Quais produtos são gerados
  - ✓ Como é gerenciada a mudança

# Métodos

- ✓ **Métodos** proveem as orientações técnicas para a construção do software.
- ✓ Métodos englobam uma grande área de tarefas que incluem:
  - ✓ Comunicação
  - ✓ Análise de Requisitos
  - ✓ Modelagem de design
  - ✓ Construção do programa
  - ✓ Teste
  - ✓ Suporte

# Ferramentas

- ✓ **Ferramentas** proveem suporte automatizado ou semiautomatizado para o processo e os métodos.
- ✓ Quando ferramentas são integradas, de forma que o produto gerado por uma possa ser utilizado por outras, diz-se que existe um sistema de suporte ao desenvolvimento denominado de *Computer-aided software engineering* – **CASE** (Engenharia de software auxiliada por computador)

# *Framework* de Processo

- ✓ Um Framework (ou arcabouço) de processo é um conjunto de atividades que são realizadas em todo desenvolvimento de software.
- ✓ Estas atividades são de dois tipos:
  - ✓ *Atividades específicas*, que são compostas por ações de Engenharia de Software, e que possuem início e término bem definidos;
  - ✓ *Atividades guarda-chuva*, que são aplicáveis ao longo de todo o processo de desenvolvimento de software.



# Atividades Específicas

- ✓ As atividades específicas de Engenharia de Software são as mesmas já vistas em APSIS, com uma roupagem ligeiramente diferente:
  - ✓ **Comunicação**: obtenção dos requisitos do cliente
  - ✓ **Planejamento**: organização do projeto
  - ✓ **Modelagem**: análise e design
  - ✓ **Construção**: geração de código
  - ✓ **Distribuição**: entrega do software

# Atividades guarda-chuva

- ✓ Monitoramento e controle do projeto
- ✓ Gerenciamento de riscos
- ✓ Garantia de qualidade de software
- ✓ Revisões técnicas formais
- ✓ Medições
- ✓ Gerenciamento da configuração de software
- ✓ Gerenciamento de reuso
- ✓ Preparação e produção do produto

# Modelos de processo

- ✓ As atividades do Framework de processo podem ser organizadas de muitas formas.
- ✓ Modelos de processo são formas específicas segundo as quais as atividades foram organizadas e que apresentam características interessantes para sua aplicação como processo de desenvolvimento de software.

# Modelos de processo

(continuação)


- ✓ Nem todos os modelos são aplicáveis ou adequados a qualquer tipo de projeto.
- ✓ Algumas características diferenciam um modelo de processo de outro.



# Diferenças entre Modelos de processo

- ✓ O fluxo geral de atividades e tarefas, bem como suas interdependências;
- ✓ O grau com que cada tarefa é definida dentro de cada atividade do framework;
- ✓ O grau com que produtos de trabalho são requeridos e identificados;

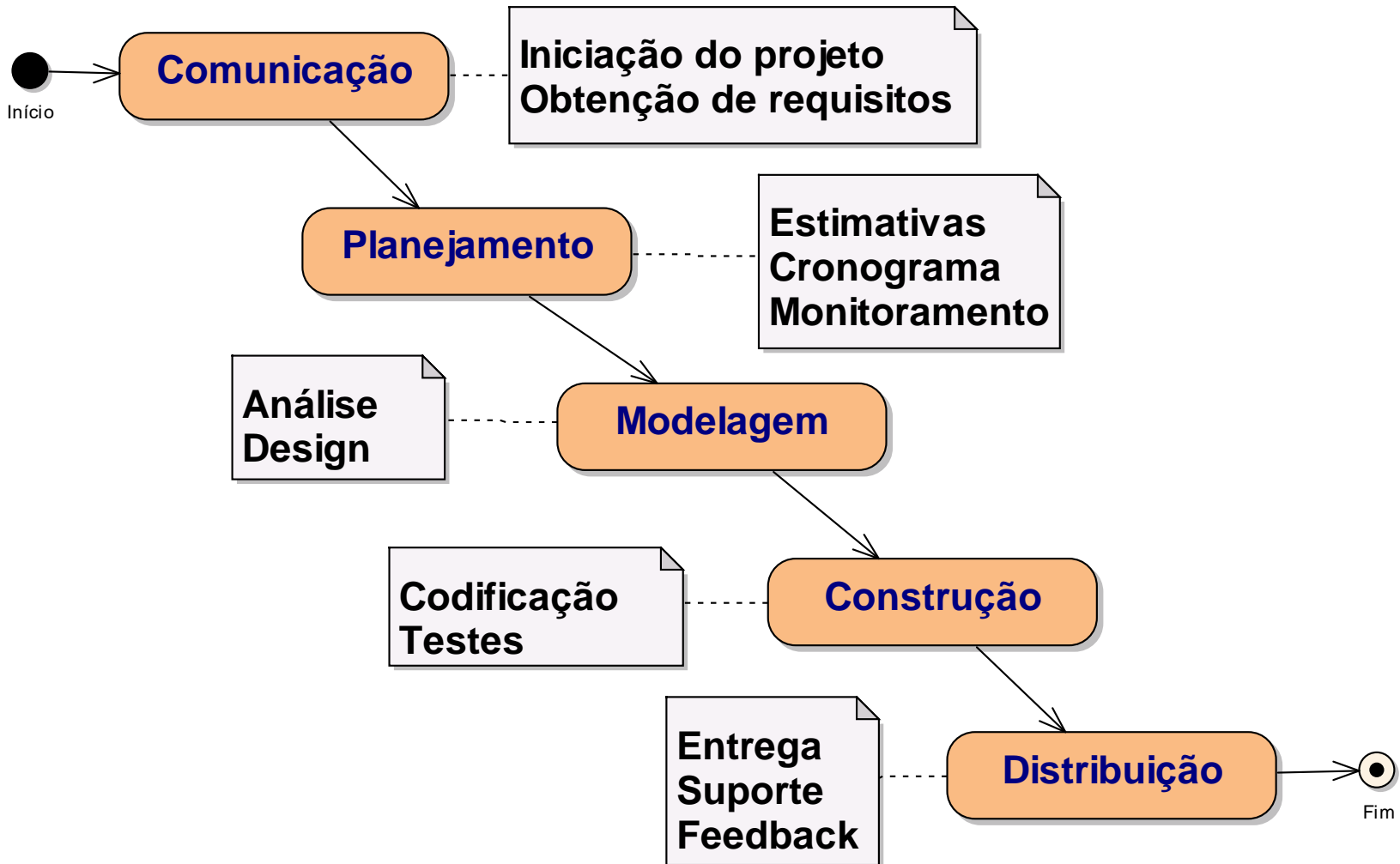
# Diferenças entre Modelos de processo (continuação)

- ✓ A maneira como as atividades de garantia de qualidade de produto são aplicadas;
  - ✓ A maneira como as atividades de controle e monitoramento de projeto são aplicadas.
- 

# Diferenças entre Modelos de processo (continuação)

- ✓ O grau geral de detalhe e rigor com que o processo é descrito;
- ✓ O grau com que o cliente e outros stakeholders estão envolvidos com o projeto;
- ✓ O nível de autonomia dado à equipe de desenvolvimento de software;
- ✓ O grau com que a organização e os papéis da equipe de desenvolvimento são prescritos.

# Waterfall





# Waterfall (continuação)

- ✓ Também conhecido como ciclo de vida clássico.
- ✓ Estratégia sequencial e sistemática para o desenvolvimento de software.
- ✓ Modelo de desenvolvimento mais antigo.
- ✓ Útil em situações onde requisitos são fixos e o trabalho para completar o desenvolvimento deve proceder de forma linear.

# Problemas do Waterfall

- ✓ Projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Embora sejam possíveis iterações, tal é feito indiretamente e as mudanças podem causar confusão.
- ✓ É frequentemente difícil para o cliente afirmar todos os requisitos explicitamente. O modelo não consegue acomodar a incerteza comum no início do projeto.
- ✓ O cliente deve ser paciente. Uma versão funcional do software não estará disponível logo.

# Modelo Incremental (continuação)

- ✓ Combina elementos do modelo waterfall aplicados de uma maneira iterativa.
- ✓ O modelo implementa sequências lineares, cada uma produzindo incrementos do software.

# Modelo Incremental (continuação)

- ✓ Este modelo é particularmente útil quando a equipe não está completamente disponível para a implementação completa.
- ✓ Equipes menores podem realizar os primeiros incrementos, e mais pessoas podem ser agregadas ao desenvolvimento para os demais incrementos.

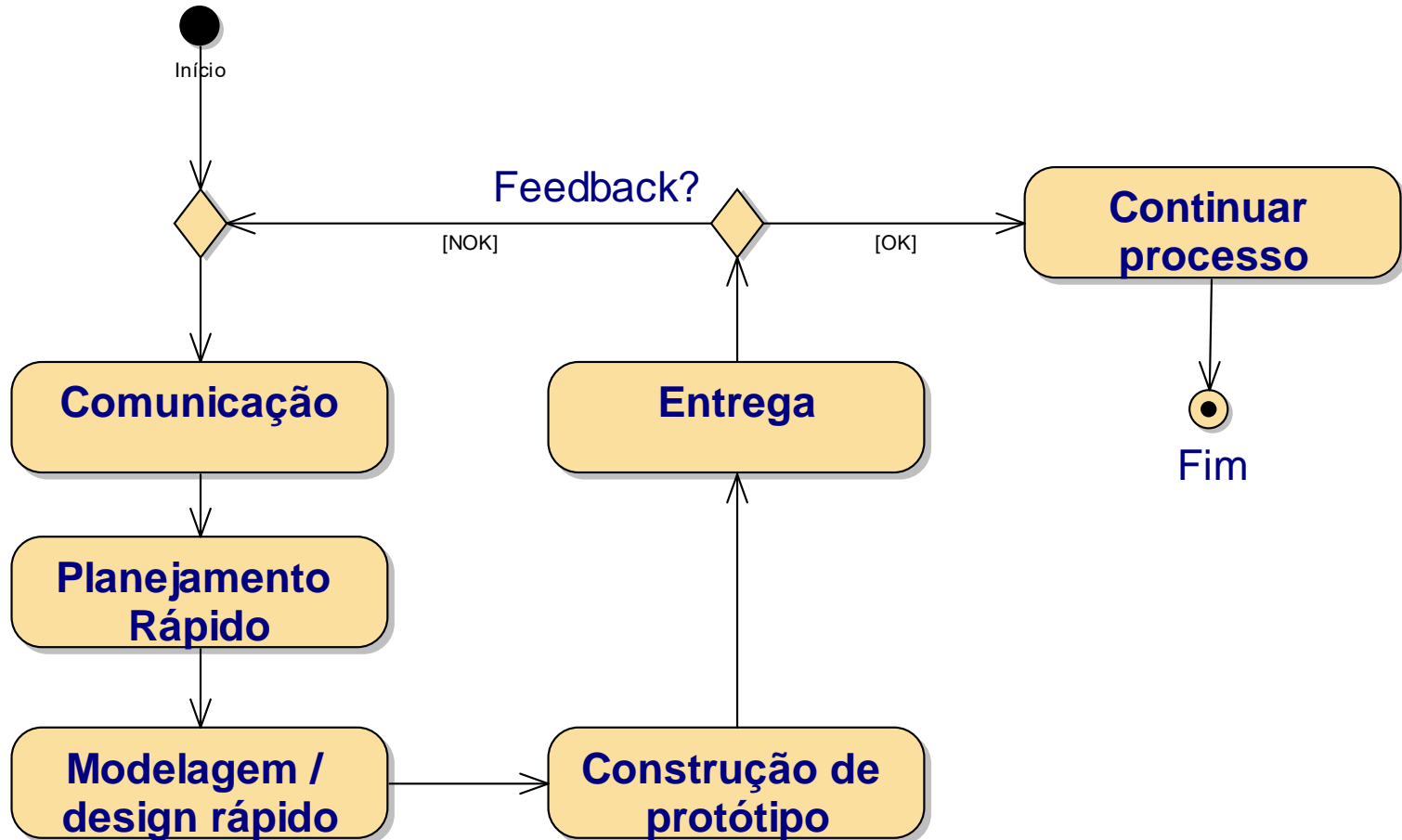
# Modelo de Desenvolvimento Rápido (continuação)

- ✓ Conhecido como RAD (*Rapid Application Development*), é um processo incremental de desenvolvimento de software que enfatiza um ciclo de desenvolvimento curto.
- ✓ É uma adaptação de alta velocidade do modelo waterfall, em que o desenvolvimento rápido é obtido pela estratégia de construção baseada em componentes.
- ✓ Se os requisitos são bem conhecidos e o escopo do projeto está bem determinado, é possível criar uma versão funcional em um prazo muito curto.


# Problemas do RAD

- ✓ Muito custoso para grandes projetos.
- ✓ Projeto precisa ser escalável.
- ✓ Se desenvolvedores e clientes não estão comprometidos com as atividades rápidas necessárias, RAD falhará.
- ✓ Se o software não pode ser modularizado, RAD é problemático.
- ✓ Pode não ser apropriado quando os riscos técnicos são altos.

# Prototipação



# Prototipação (continuação)

- ✓ Pode servir como o processo de desenvolvimento em si, ou ser utilizado como uma parte deste, para:
    - ✓ Identificar a forma como a interação homem-máquina deve ocorrer;
    - ✓ Validar a eficiência de um algoritmo;
    - ✓ Validar o uso de uma tecnologia específica.
- 



# Problemas da Prototipação

- ✓ É necessário planejar adequadamente o que será feito com o protótipo depois de ele ser utilizado para apresentação ao cliente.
- ✓ Este planejamento precisa ficar claro ao cliente, para que lhe seja aceitável reconstruir todo o sistema depois que as características que motivaram o protótipo tiverem sido validadas.



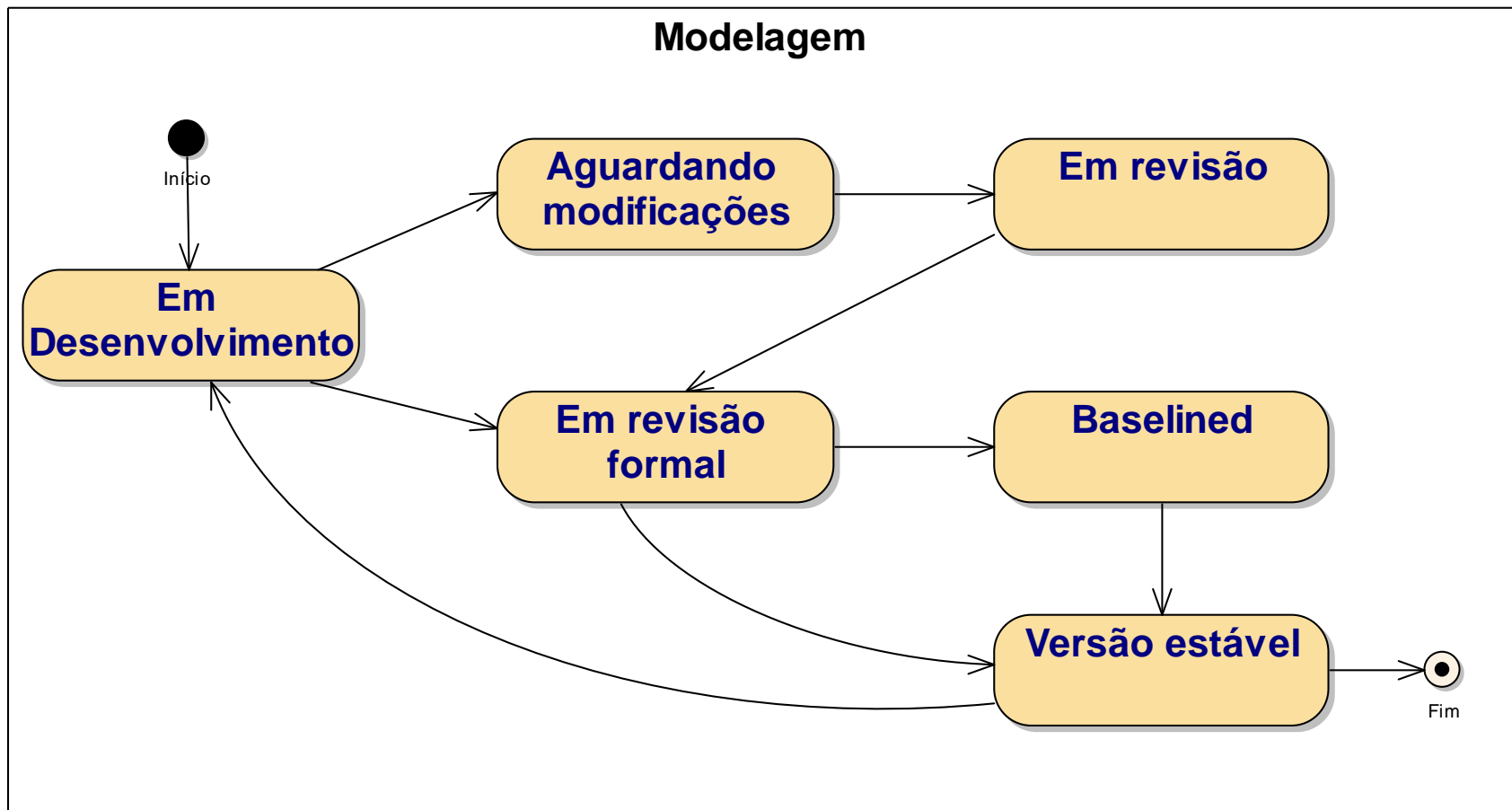
# Modelo Espiral (continuação)

- ✓ Combina elementos do modelo waterfall com a natureza iterativa da prototipação.
- ✓ Estratégia cíclica de compromisso entre:
  - ✓ Redução de risco
  - ✓ Grau de definição do sistema
  - ✓ Interação com cliente para identificação do grau de satisfação com o sistema no ponto de verificação (marcos de ponto de âncora).

# Problemas do Modelo Espiral

- ✓ Pode ser difícil convencer o cliente de que o processo é controlável.
- ✓ Exige considerável especialização em avaliação de risco e se baseia nessa especialização para o sucesso.
- ✓ Se um risco maior não é descoberto e gerenciado, pode se tornar um grande problema.

# Modelo de Desenvolvimento Concorrente



# Modelo de Desenvolvimento Concorrente

(continuação)

- ✓ Todas as atividades do desenvolvimento acontecem concorrentemente.
- ✓ Cada uma delas está em um estado.
- ✓ São definidos os eventos que gatilham as transições de um estado para outro.
- ✓ Aplicável a projetos de qualquer tamanho.
- ✓ Principal problema do modelo é o gerenciamento do estado em que cada atividade se encontra.