Análise de Sistemas – AULA 03
BCC Noturno - EMA908915A

Prof. Rafael Oliveira
rpaes@ic.unesp.br

Universidade Estadual Paulista
“Júlio de Mesquita Filho”
UNESP

Rio Claro 2014 (Sem 2)

Revisão Aula 02

- Engenharia de Software
- Processo de Software
  - Modelos prescritivos
  - Modelos ágeis

Rev: Processo de software

unesp

- Visão Genérica:
  - Um processo é uma coleção de atividades, ações e tarefas que são realizadas quando algun produto deve ser criado.
  - Uma atividade se esforça para alcançar um amplo objetivo (ex. Comunicação com stakeholders) e é aplicado independentemente do domínio da aplicação, tamanho do projeto, complexidade de esforços, ou grau de rigor com o qual a engenharia de software é aplicada.
  - Uma ação (ex: projeto arquitetural) engloba um conjunto de tarefas que produz um grande produto (ex: um modelo de projeto arquitetural).
  - Uma tarefa foca em um pequena, mas com um objetivo bem-definido (ex: conduzindo uma unidade de teste) que produz um resultado tangível.

Rev: Processo de software

- Um framework genérico de engenharia de software engloba 5 atividades:
  - Comunicação;
  - Planejamento;
  - Modelagem;
  - Construção;
  - Implantação.

Agenda – Aula 03

- Praticando a Engenharia de Software
- Como iniciar!?
- Processos de software
  - Modelos prescritivos de processo
  - Modelos ágeis (próxima aula)
  - Processo de Modelos específicos (próxima aula)
  - Processo uniicado (próxima aula)

Praticando Eng. Software

- Como funciona a Eng. de Software na prática?

  Um framework genérico para desenvolvimento foi sugerido (comunicação, planejamento, modelagem, construção e desenvolvimento):

  - Mas e aí!? Como isso se associaria à Eng. Soft?

Praticando Eng. Software

Resp: Partindo da essência da solução do problema!

- (1) Entendimento do problema (comunicação e análise)
- (2) Plano de solução (Modelagem e projeto de software)
- (3) Execução do plano (geração de código)
- (4) Examinação da pecisão dos resultados

Praticando Eng. Software

Tais itens levam a uma série de questões:

- **(1) Entendimento do problema (comunicação e análise)**

  Quem são os stakeholders?
- O que é desconhecido? Quais dados funções e características requeridas para resolver o problema adequadamente?
- O problema pode ser dividido em subproblemas? Tais problemas menores são mais fáceis de entender?
- O problema pode ser representado graficamente? Pode ser criado um modelo de análise?

Praticando Eng. Software

Tais itens levam a uma série de questões

- **(2) Plano de solução (Modelagem e projeto de software)**
- Você já viu algum problema similar anteriormente!?
- O problema similar foi solucionado? Se sim, os elementos são reutilizáveis?
- Os subproblemas puderam ser definidos? Se sim, há subproblemas associados?
- Você consegue representar a solução de modo ao direcionamento de uma implementação efetiva!? É possível que um modelo de projeto seja criado?

Praticando Eng. Software

Tais itens levam a uma série de questões:

- **(3) Execução do plano (geração de código)**
- A solução está de acordo com o plano? O código fonte é associado ao modelo do projeto?
- Cada componente que forma a solução está correto? O projeto e o código foram revisados adequadamente?

Tais itens levam a uma série de questões:

- **(4) Examinação da pecisão dos resultados**
- É possível testar cada componente que é parte da solução?
- A solução produz resultados em conformidade com o que é esperado: dados funções e características requeridas? O programa foi validado considerando os requisitos dos stakeholders?

Praticando Eng. Software

Consideração de princípios:

- 7 princípios podem ser considerados ao se praticar a Eng. de Software na indústria:
  - 1 A razão da existência do software
  - 2 KISS(Keep It Simple, Stupid)
  - 3 Mantenha a visão
  - 4 O que você produz, os outros consumem
  - 5 Esteja aberto ao futuro
  - 6 Planeje para futuros reusos
  - 7 Pense

Praticando Eng. Software

Princípios:

- **1 A razão da existência do software**

  “Um software existe para gerar/agregar algum valor para seu usuário ou organização/indústria”

  Antes de especificar requisitos, determinar tecnologias, definir hardware e plataformas,, é necessário se perguntar:

- **“Isto adiciona valor real ao sistema?”**
- **Se a resp for não, então, não faça!!**

Praticando Eng. Software

unesp

Princípios:

- **2 KISS(Keep It Simple, Stupid)**
- Projeto de software não é um processo casual.
- Todos os projetos e designs devem ser o mais simples possível, mas não subjetivos.
- Isso facilita a manutenção e entendimento
- Simples não significa rápido e bagunçado.

Praticando Eng. Software

Princípios:

- **3 Mantenha a visão**
- Uma visão clara é fundamental para o sucesso do software;
- Sem uma visão, um software torna-se uma mistura/miscelânia sem integrações consistentes e compatíveis;
- Mantendo-se uma visão e objetiva e o foco no produto final, o projeto será bem sucedido.

Praticando Eng. Software

unesp

Princípios:

- **4 O que você produz, os outros consumem**
- De algum modo, o que você está implementando será utilizado, mantido ou documentado por alguém;
- Então, sempre: especifique, projete e implemente sabendo que alguém vai ter que entender de algum modo o que você está fazendo;
- O público para os produtos de software modernos são geralmente diversos.

Praticando Eng. Software

Princípios:

- **5 Esteja aberto ao futuro**
- Um sistema com um tempo de vida maior tem mais valor;
- Nos dias atuais com o surgimento de novas tecnologias, hardware e especificações, o tempo de vida do software é medido em meses, não em anos.
- Entretanto, produtos de software de verdade devem durar mais do que a média;
- Nunca projete isoladamente e sem ouvir opiniões diversas de companheiros do time de desenvolvimento.

Praticando Eng. Software

Princípios:

- **6 Planeje para futuros reusos**
- Reúso economiza tempo e esforços;
- Atingir um alto nível de reúso em seus projetos de software é um dos objetivos mais complexos durante o desenvolvimento;
- Reúso é o maior benefício dos projeto orientados a objeto;
- Planejar futuros reúsos reduz custos dos novos projetos e aumenta o valor dos componentes reutilizáveis e do sistema no qual tais componentes serão incorporados

Praticando Eng. Software

Princípios

- **7 Pense**
- Pensar de modo claro, completo antes de quase todas a ações e tomadas de decisão prodozem resultados melhores
- Quando se pensa profundamente sobre efeitos e trade-offs de alguma técnica ou estratégia, se está ganhando conhecimento para projetos futuros;
- O efeito colateral de de pensar profundamente é o reconhecimento de você não tem domínio suficiente sobre algo e precisa assim pesquisar.
- A aplicação dos 6 primeiros princípios requer muito pensamento em diversas atividades.

Como iniciar

- Todo projeto de software é precido por alguma necessidade de negócio;
  - Corrigir um defeito de uma aplicação existente;
  - Adaptar um sistema legado;
  - Adicionar funcionalidades;
  - Ou então, criar um novo produto, serviço ou sistema.

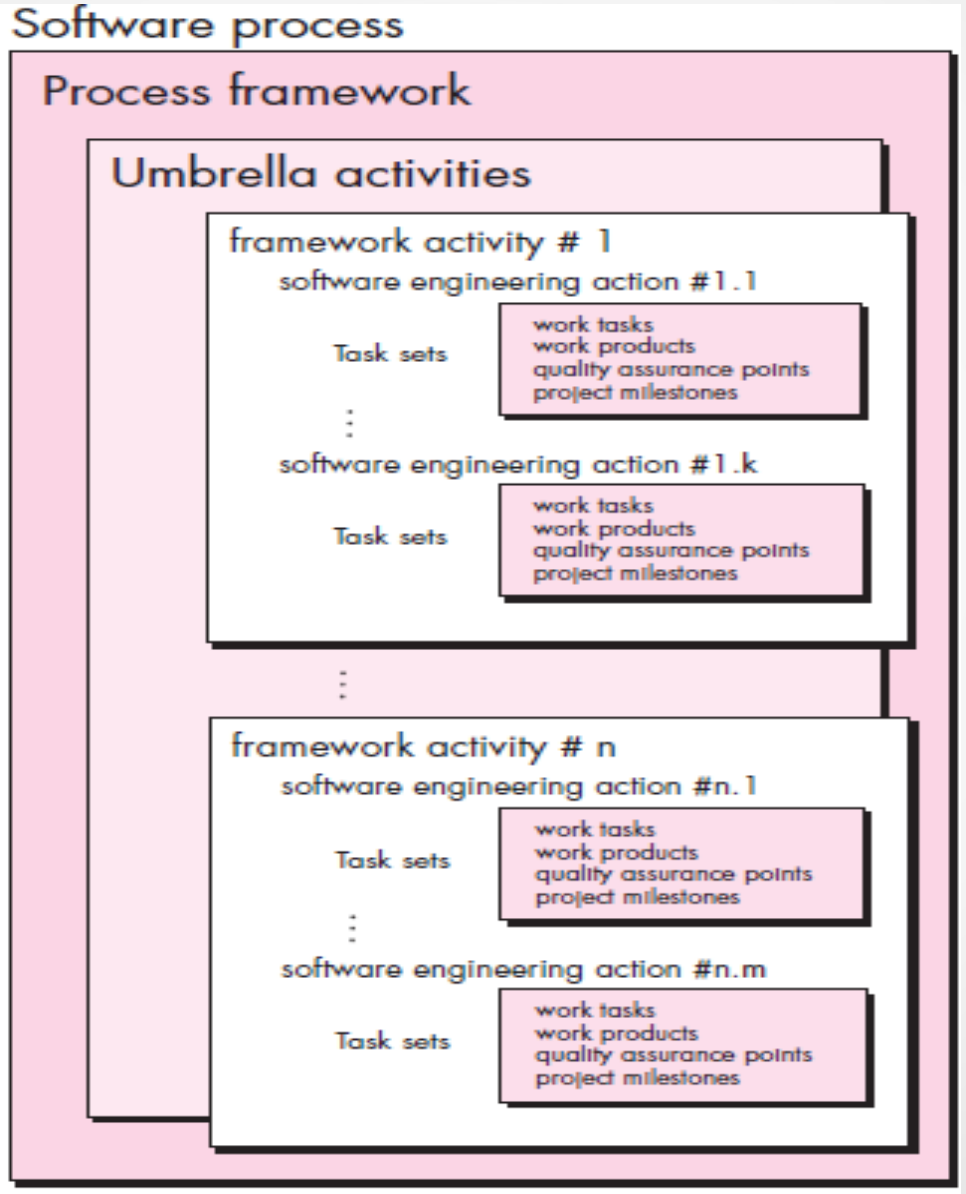
Como iniciar

- Todo projeto de software é precido por alguma necessidade de negócio;
  - No início, um projeto de software é uma necessidade de negócio expressada informalmente;
  - Informalmente como parte de uma simples conversa;
  - A partir disso, o software começa a ser entendido como uma solução que é endereçada para as necessidades do cliente.

Processo

- Aula passada: Processo foi definido como uma coleção de atividades, ações e tarefas;
- Cada uma dessas atividades, ações e tarefas residem em um framework ou modelo que define suas relações com outros frameworks ou modelos;
- Veja a figura do próximo slide;
- O processo de software é representado esquematicamente;
- Cada framework de atividade é povoado por um conjunto de ações de Eng. Software.
- Cada ação de Eng. Software é definida por um conjunto de tarefas

Processo


Software process
Process framework
Umbrella activities
framework activity # 1
software engineering action #1.1
Task sets
work tasks
work products
quality assurance points
project milestones
software engineering action #1.k
Task sets
work tasks
work products
quality assurance points
project milestones
framework activity # n
software engineering action #n.1
Task sets
work tasks
work products
quality assurance points
project milestones
software engineering action #n.m
Task sets
work tasks
work products
quality assurance points
project milestones

Processo

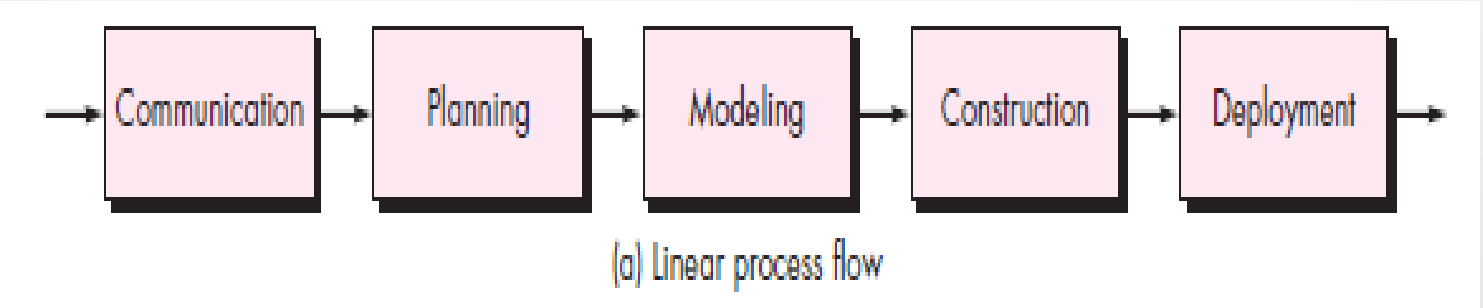
- Importantes aspectos dos processos não foram discutidos:
  - Fluxo do processo;
    - Descrevem como as atividades dos frameworks são organizadas em respeito à sequência e tempo

Processo

- Fluxo do processo:
  - Linear
  - Iterativo
  - Evolucionário
  - Paralelo

Processo

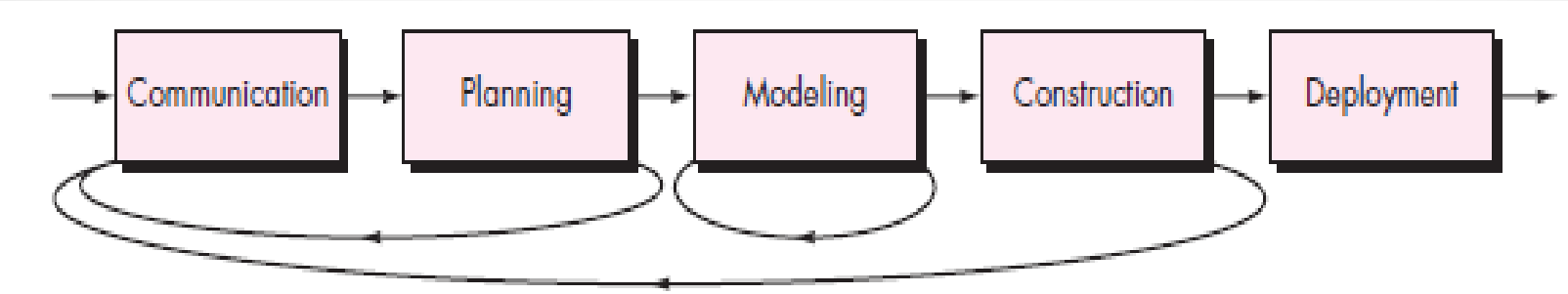
- Fluxo do processo:
  - **Linear:** Um processo de fluxo linear executa cada uma de suas atividades em sequência, iniciando com a comunicação e terminando com o desenvolvimento.



(a) Linear process flow

Processo

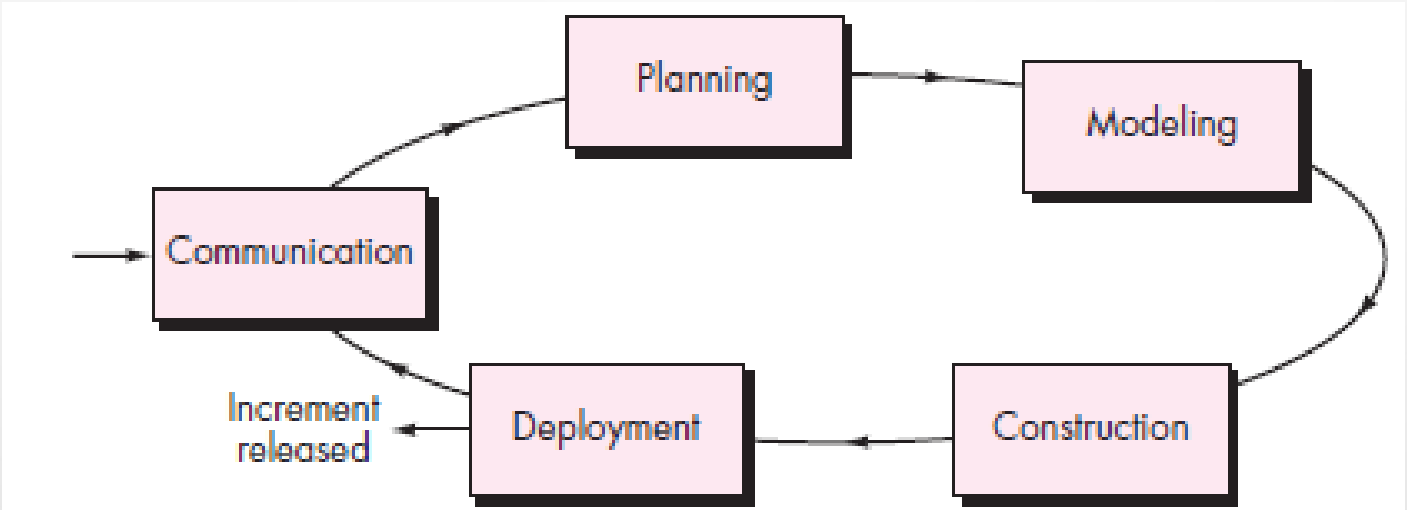
- Fluxo do processo:
  - **Iterativo:** Um processo de fluxo iterativo repete uma ou mais atividades antes de proceder para a próxima;



(b) Iterative process flow

Processo

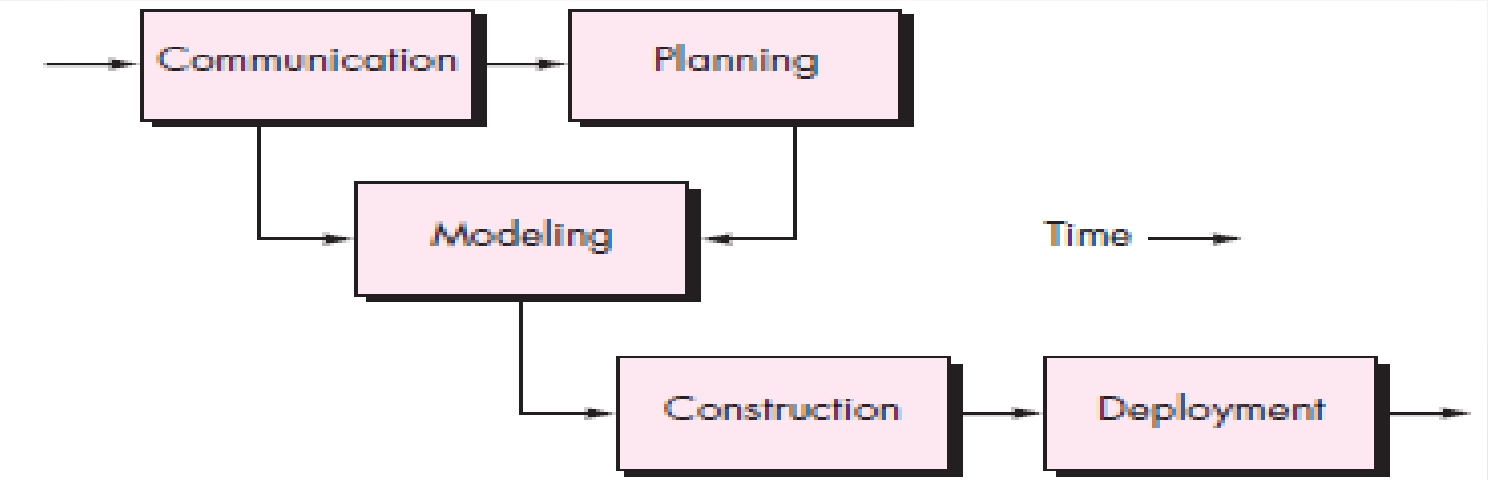
- Fluxo do processo:
  - **Evolucionário:** Processos de fluxos evolucionários executam atividades de modo circular. Cada circuito de atividades leva a uma versão mais completa do software;



(c) Evolutionary process flow

Processo

- Fluxo do processo:
  - **Paralelo:** Um processo de fluxo paralelo executa uma ou mais atividades de modo paralelo a outro aspecto do software;



(d) Parallel process flow

Processo

- Definindo um framework de atividades:
  - Apesar de termos elencados alguns frameworks genéricos, um time de desenvolvimento necessita de muito mais informações como parte do processo de software
  - Então, desenvolvedore chegam a uma questão:
    - *Quais ações são apropriadas para um framework de atividades, dado um problema do mundo real?*

Processo

- Identificação de um conjunto de tarefas:
  - A identificação de tarefas a serem executadas é uma tarefa particular para cada projeto;
  - Cada atividade de Eng. Software (ex: elicitação) pode ser representada por um conjunto diferente de atividades;
  - O Analista deve escolher um conjunto de tarefas que melhor acomode as necessidades do projeto de acordo com as características do seu time de desenvolvedores;
  - Isso implica dizer que ações de Eng. Software podem ser adaptadas para necessidades especiais do projeto de software e do time de desenvolvimento.

Processo

- Avaliação e melhora de processos:
  - A existência do processo de software não garante sua qualidade, pontualidade de entrega e que ele atende às necessidades do cliente por um longo tempo;
  - Os processos, por eles mesmos, podem ser avaliados para garantir que ele corresponde a um conjunto básico de critérios para a Engenharia de Software bem sucedida

Processo

- Avaliação e melhora de processos: Nas últimas décadas abordagens para avaliação e melhora no processo de software foram propostas:
  - Standard CMMI Assessment Method for Process Improvement (SCAMPI);
  - CMM-Based Appraisal for Internal Process Improvement (CBA IPI);
  - SPICE (ISO/IEC15504);
  - ISO 9001:2000 for Software

Processo

- Avaliação e melhora de processos: Nas últimas décadas abordagens para avaliação e melhora no processo de software foram propostas:
  - **Standard CMMI Assessment Method for Process Improvement (SCAMPI);**
  - *“provides a five-step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting, and learning. The SCAMPI method uses the SEI CMMI as the basis for assessment.”*

Processo

- Avaliação e melhora de processos: Nas últimas décadas abordagens para avaliação e melhora no processo de software foram propostas:
  - **CMM-Based Appraisal for Internal Process Improvement (CBA IPI);**
  - *“provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment”*.

Processo

- Avaliação e melhora de processos: Nas últimas décadas abordagens para avaliação e melhora no processo de software foram propostas:
  - **SPICE (ISO/IEC15504)**
  - *“ standard that defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process”*

Processo

- Avaliação e melhora de processos: Nas últimas décadas abordagens para avaliação e melhora no processo de software foram propostas:
  - **ISO 9001:2000 for Software**
  - *“ a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies ”*

Processo – Modelos prescritivos

- *Modelos prescritivos:*
  - Propostos originalmente para trazer ordem ao caos que era o desenvolvimento de software;
  - A literatura indica que estes modelos trouxeram estruturas muito úteis para a Engenharia de Software

Processo – Modelos prescritivos

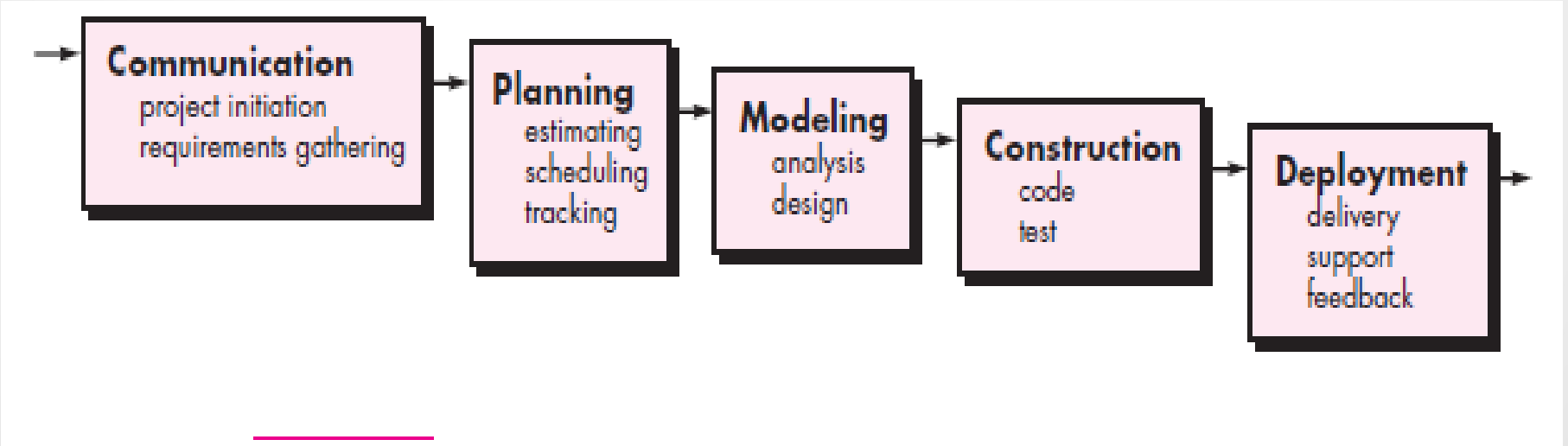
- *Modelos prescritivos;*
  - Modelo cascata
  - Modelos de processos incrementais
  - Modelos de processos evolucionários
  - Modelos concorrentes

Processo – Modelos prescritivos

- *Modelo cascata*
  - Ás vezes chamado de clássico ciclo de vida do software
  - Sugere um modo sistemático e sequencial de desenvolvimento
  - Quando os requisitos de um problema são bem entendidos
  - Poucas adaptações precisam ser feitas nas atividades

Processo – Modelos prescritivos

- *Modelo cascata*



Processo – Modelos prescritivos

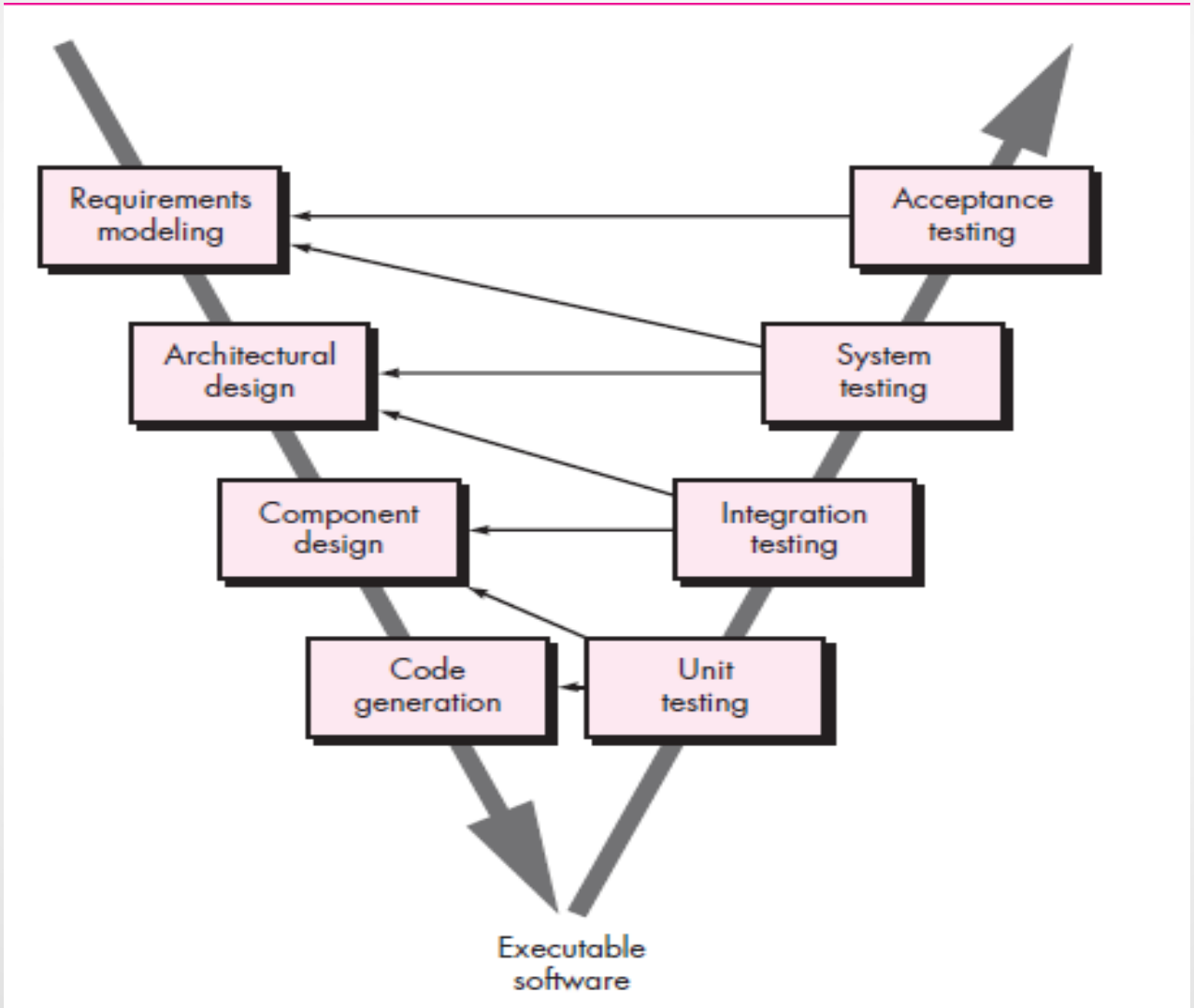
- *Modelo cascata*
  - Mais velho modelo da E.S
  - Críticas recentes
    - Uso limitado em projetos reais;
    - É de díficil entendimento por parte do cliente;
    - O cliente deve ter paciente (Não há versões parciais do software)
  - Pode ser considerado inapropriado nos dias atuais, mas serve de referência para outros modelos

Processo – Modelos prescritivos

- Modelo em V (variação do modelo cascata)
  - A equipe de desenvovimento deve mover-se para baixo, partindo da esquerda do V;
  - Requisitos básicos são refinados e progressivamente detalhados;
  - Posteriormente, o time move-se para cima no V;
  - O modelo V promove um modo de visualizar como ações de verificação e validação são aplicados em engenharias prévias.

Processo – Modelos prescritivos

- Modelo em V (variação do modelo cascata)



Processo – Modelos prescritivos

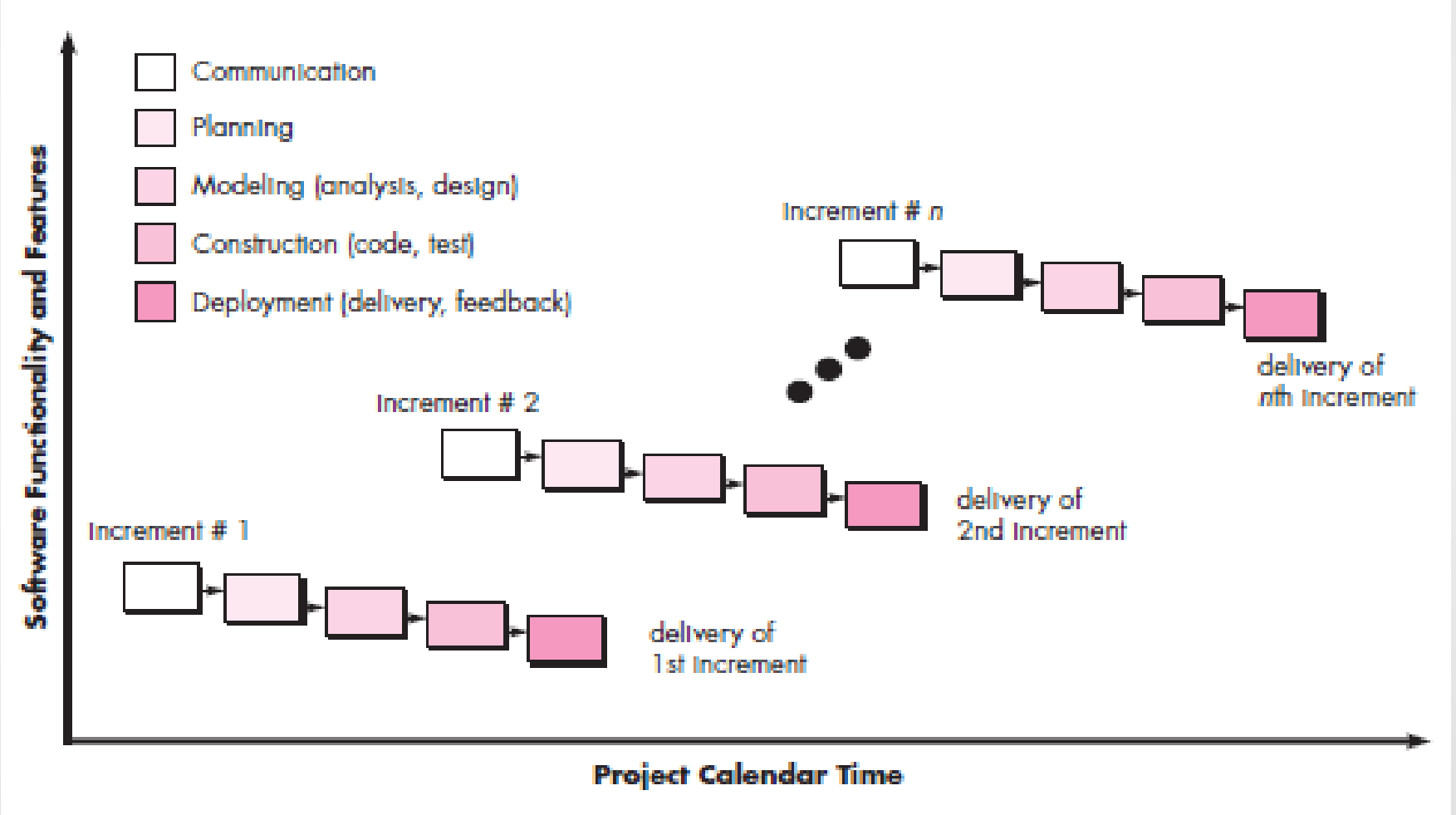
- Modelo incremental
  - *Combina elementos de processos de fluxos paralelos e lineares (apresentados anteriormente)*
  - *Cada sequência linear produz elementos "entregáveis" de software de uma maneira similar ao incremento produzido nos processos evolucionários*

Processo – Modelos prescritivos

- Modelo incremental
  - *“For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment can incorporate the prototyping paradigm”*

Processo – Modelos prescritivos

- Modelo incremental



Processo – Modelos prescritivos

- Processo evolucionário:
  - *Sistemas como, por exemplo, complexos envolvem um período de tempo;*
  - *Negócios e requisitos de produtos mudam conforme as necessidades;*
  - *Modelos envolucionários são iterativos*
  - *São caracterizados por permitir o desenvolvimento de software de um modo mais completo a cada iteração;*
  -

Processo – Modelos prescritivos

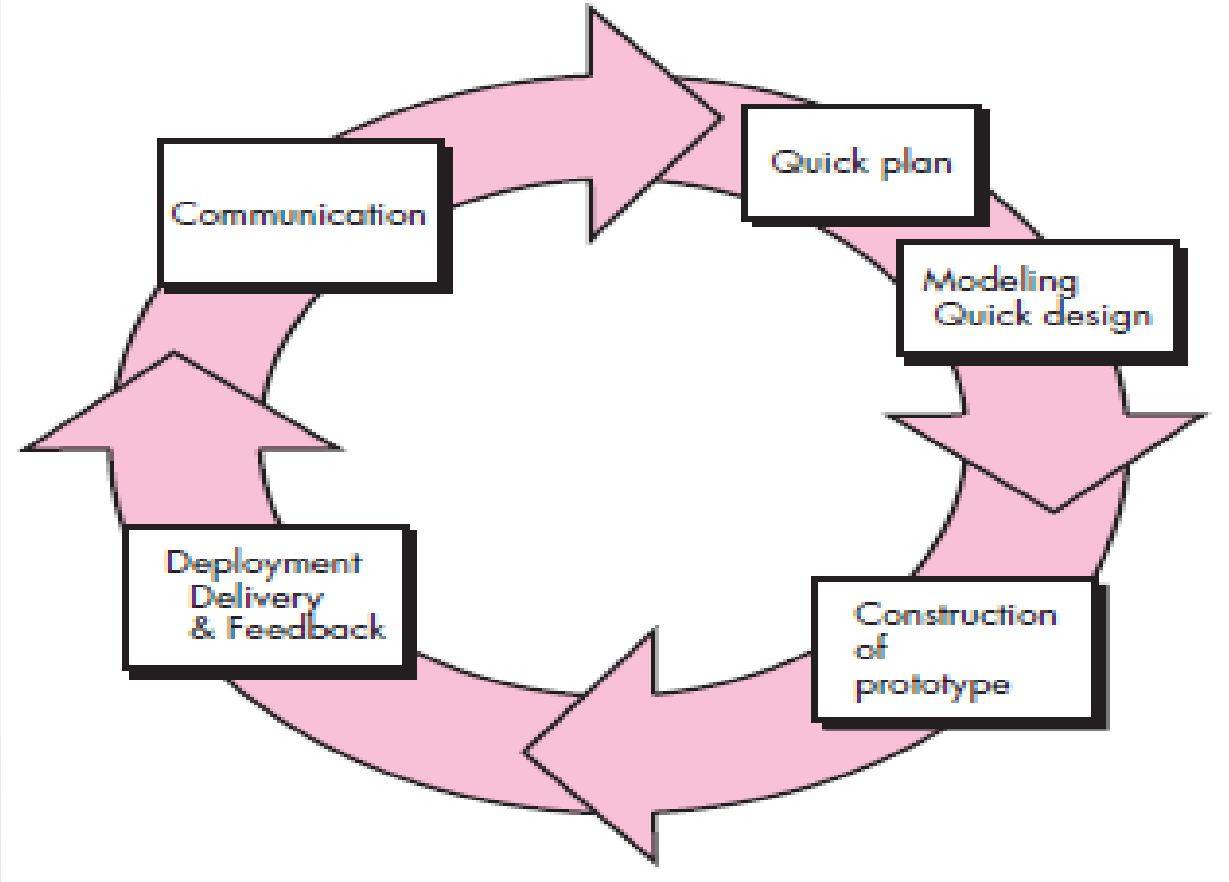
- Processo evolucionário:
  - Exemplos:
    - Prototipagem
    - Modelo Espiral

Processo – Modelos prescritivos

- Processo evolucionário:
  - **Prototipagem**
    - Útil quando o cliente especifica alguns objetivos gerais sem especificar requisitos
    - Nesses casos, a criação de protótipos é a melhor opção
    - Protótipos podem ser utilizados stand-alones
    - O protótipo serve como sendo o primeiro sistema a ser evoluído

Processo – Modelos prescritivos

- Processo evolucionário:
  - **Prototipagem**

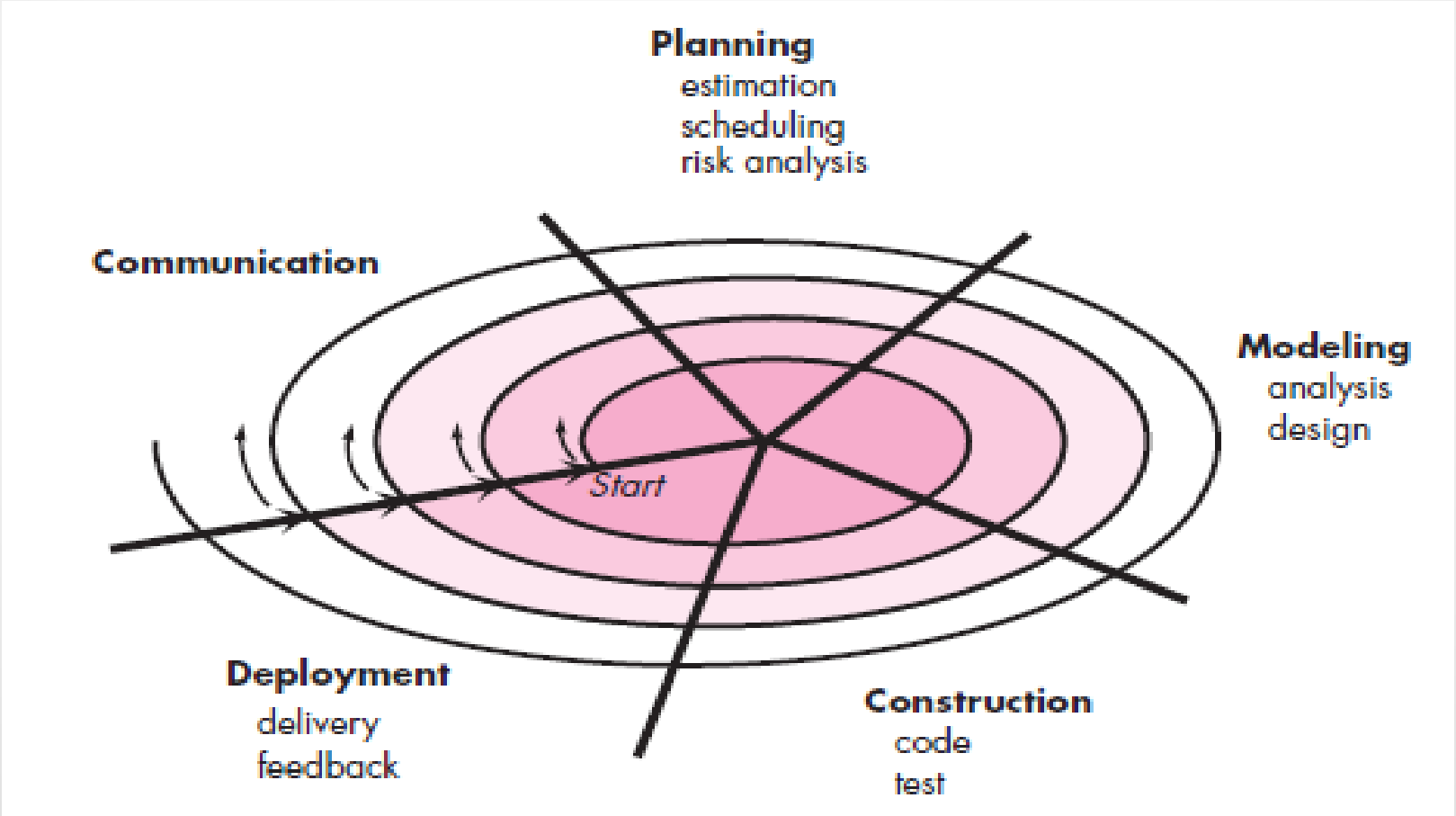


Processo – Modelos prescritivos

- Processo evolucionário:
  - **Modelo espiral**
    - Utilizar o modelo espiral implica em desenvolver uma série de releases evolucionário do software em desenvolvimento
    - As versões mais completas serão produzidas e liberadas nas últimas iterações;
    - O espiral é dividido em uma série de atividades desenvolvida pelo time de Eng. Soft.
    - É a melhor estratégia para software complexos
    - Novas versões formam a complexidade do sistemade.
    - Desvatagem: Difícil convencer clientes que a evolução do software é controlável

Processo – Modelos prescritivos

- Processo evolucionário:
  - **Modelo espiral**

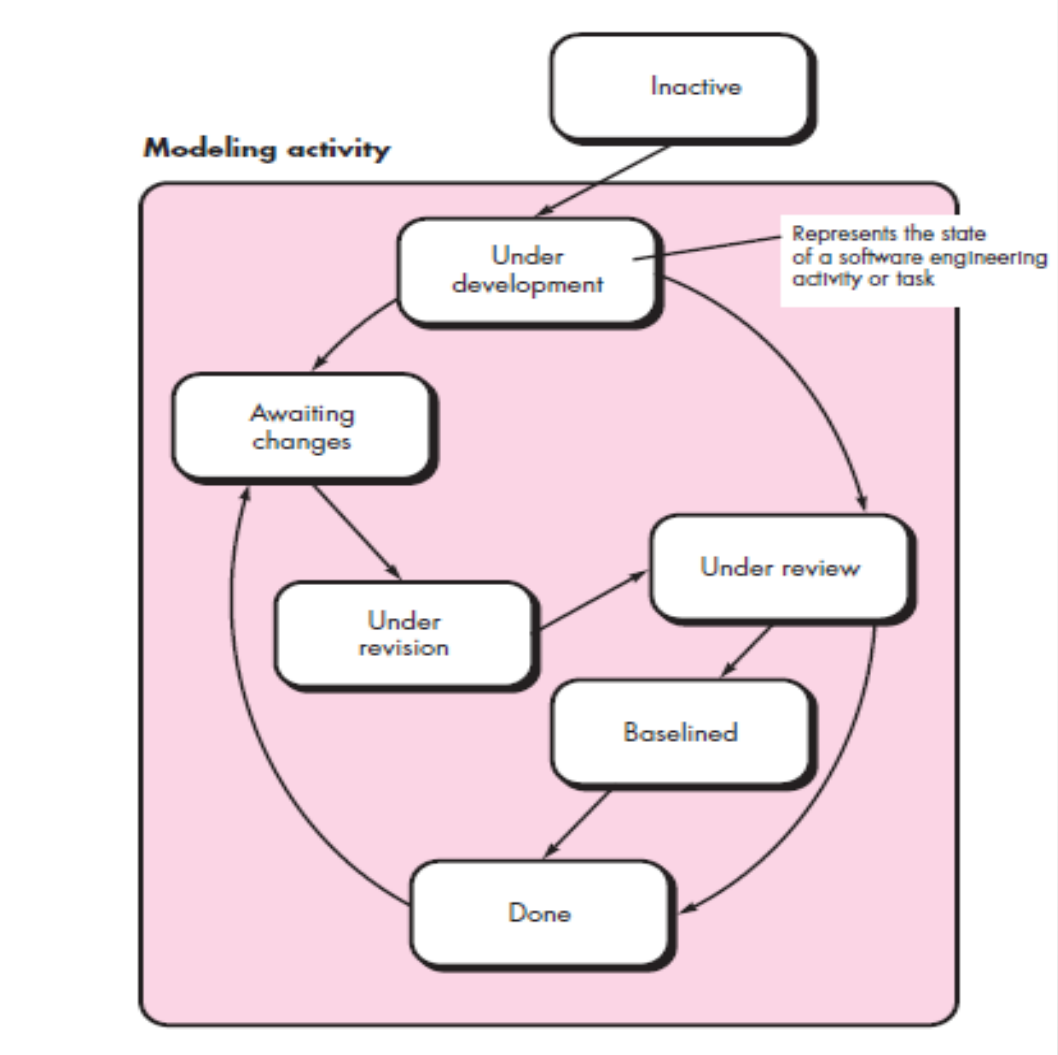


Processo – Modelos prescritivos

- Modelos concorrentes:
  - Permitem que o time de desenvolvimento a implementação de atividades de modo iterativo e concorrente;
  - O analista deve controlar a sincronizar as atividades paralelas.

Processo – Modelos prescritivos

- Modelos concorrentes:



Exercício 01

- 1 – Descreva um framework de processo de desenvolvimento de software com suas próprias palavras. Quando dizemos que um framework de atividades é aplicável a qualquer projeto, isso não significa dizer que as mesmas tarefas são aplicáveis para todos os projetos, independentemente de tamanho e complexidade? Explique!

Exercício 02

- 2 – Pode ser dito que o processo promove interação entre usuário e projetistas, usuários e ferramentas e, por fim, projetistas e ferramentas. Elenque 5 questões que:
  - Projetistas devem fazer aos usuários
  - Usuários devem fazer aos projetistas
  - Usuários devem fazer a eles mesmos sobre o produto a ser consumido
  - Projetistas devem fazer a eles mesmos sobre o produto a ser implementado

Exercício 03

- 3 – Imagine-se sendo um projetista de um projeto qualquer (defina o projeto em uma sentença). Tente desenvolver um conjunto de ações para a atividade de comunicação. Selecione uma dessas ações e definas as tarefas para que a ação seja completada (está com dúvidas: releia slides da aula passada!)

Exercício 04

- 4 – Pense e apresente 3 exemplos de projetos que seriam implementáveis por meio do modelo de cascata.

Exercício 05

- 5 – Pense e apresente 3 exemplos de projetos que seriam implementáveis por meio do modelo incremental.

Exercício 06

- 6 – Pense e apresente 3 exemplos de projetos que seriam implementáveis por meio de algum processo evolucionário.