Especialização em Gestão Estratégica de Tecnologia da Informação

Engenharia de Software

Visão Geral

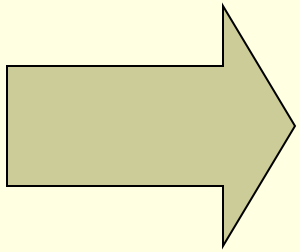
Introdução à Engenharia de Software

- Objetivo
  - Depois desta aula você terá uma noção geral do que é a engenharia de software e dos seus objetivos e conceitos básicos relacionados.

- “O Software ultrapassou o Hardware como chave para o sucesso de muitos sistemas baseados em computador” (Pressman, pg. 3, 1992)

O Software é o que faz a diferença!!!

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



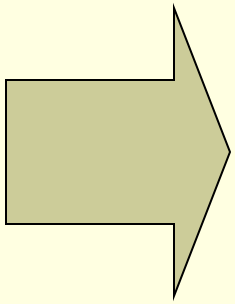
Tornam 1 produto melhor que outro

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**!

A evolução do Software

Computação

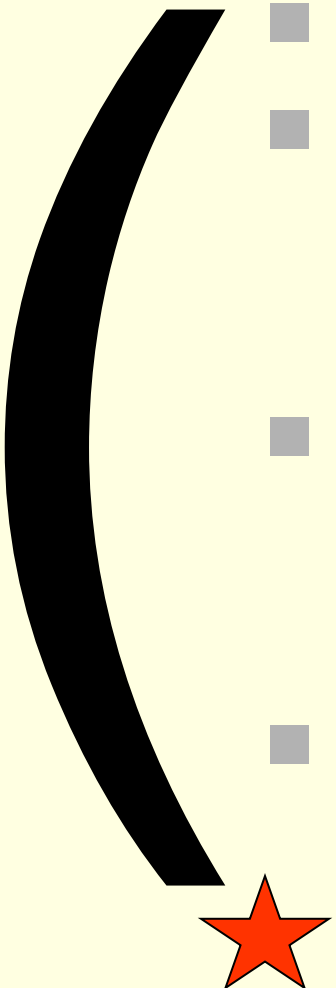


- Nova Revolução Industrial (*Toffler*)
- 3a. Onda

Parêntesis: Revolução Industrial Primeira Onda

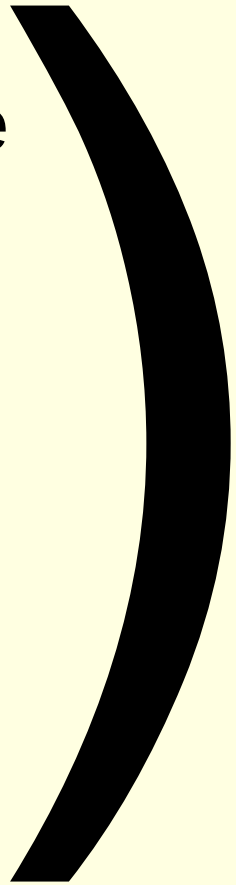
- Ferro (Darby, 1709)
- Máquina a vapor:
  - Inventada (Newcomen, 1712)
  - Aperfeiçoada (WATT, 1766 - ’69 -’82)
- Mecanização da indústria têxtil:
  - Tear Mecânico (Kay, 1722)
  - Maquina de fiar (Hargreaves, 1764)
- Aspectos sociais, políticos e econo

Têxteis, Carvão e Ferro



Parêntesis: Revolução Industrial Segunda Onda

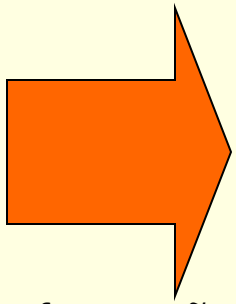
- Aço (Bessemel, 1856 e 1885 - Liga)
- Locomotiva a Vapor (Rede de Transporte - 1830)
- Máquina de Costura (SINGER,1851)
- Motor a combustão interna:
  - Primeiro eficiente (OTTO, 1876)
  - Produção automobilística em massa (Daimler e Benz, 1896)
- Desemprego e fim da escravidão



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...

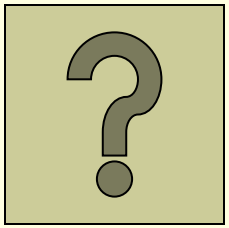
- A mudança de uma sociedade industrial para uma baseada na informação é uma *Radical Mudança Econômica*:
  - Material tem menos valor e Informação tem mais valor

- 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???

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.
- Diferente do Hardware.
- Situação esta mudando:
  - Orientação a objetos.
  - Reusabilidade é o “Santo Graal”(diminui custos e melhora projetos).

Uma Crise no horizonte (próximo)

- 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.

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

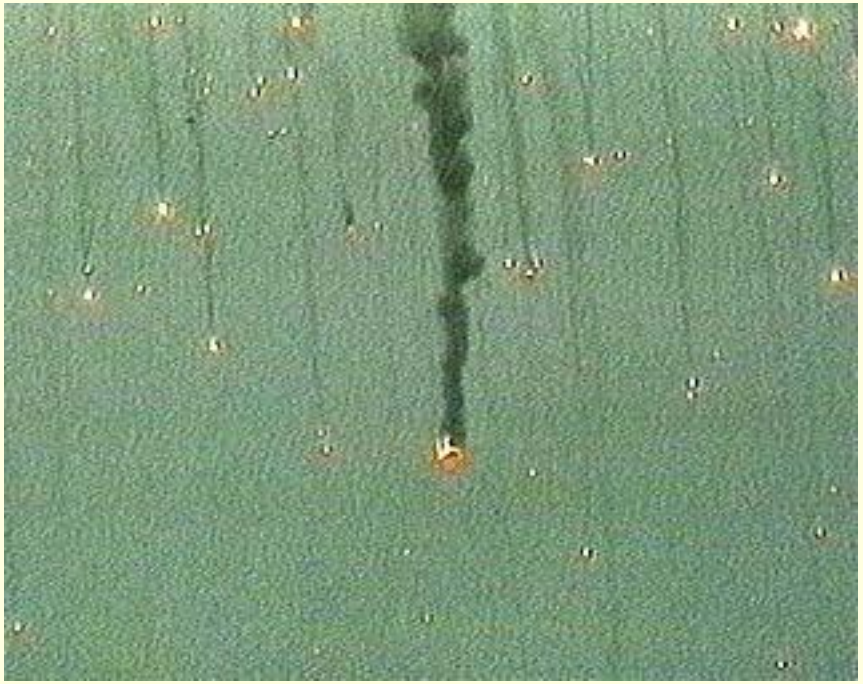
Ariane 5

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?
-

strict precondition 1:

{

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

and -32768 <= x <= +32767

}

Ironia...

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

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.

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.

O que é a Engenharia de Software?

Estudo ou aplicação de abordagens sistemáticas, econômicas e quantificáveis para o desenvolvimento, operação e manutenção de software de qualidade.

Objetivos da Engenharia de Software

- Qualidade de software
- Produtividade no desenvolvimento, operação e manutenção de software
- Qualidade versus Produtividade
- Permitir que profissionais tenham controle sobre o desenvolvimento de software dentro de custos, prazos e níveis de qualidade desejados

Pressman's Software Engineering (Third Edition)

- Software and Software Engineering
- Project Management: Software Metrics
- Project Management: Estimation
- Project Management: Planning
- Computer System Engineering
- Requirements Analysis Fundamentals
- Structured Analysis and its Extensions
- Object-oriented Analysis and Data Modeling
- Alternative Analysis Techniques and Formal Methods

Pressman's Software Engineering (Third Edition)

- Software Design Fundamentals
- Data Flow-oriented Design
- Object-oriented Design
- Data-oriented Design Methods
- User Interface Design
- Real-time Design
- Programming Languages and Coding
- Software Quality Assurance
- Software Testing Techniques
- Software Testing Strategies
- Software Maintenance
- Software Configuration Management

Pressman's Software Engineering (Third Edition)

- Computer-aided Software Engineering
- Integrated CASE Environments
- The Road Ahead

Qualidade de Software (para o Varejo)

- Correto
  - A loja não pode deixar de cobrar por produtos comprados pelo consumidor
- Robusto e altamente disponível
  - A loja não pode parar de vender
- Eficiente
  - O consumidor não pode esperar
  - A empresa quer investir pouco em recursos computacionais (CPU, memória, rede)

Qualidade de Software (para o Varejo)

- Amigável e fácil de usar
  - A empresa quer investir pouco em treinamento
- Altamente extensível e adaptável
  - A empresa tem sempre novos requisitos (para ontem!)
  - A empresa quer o software customizado do seu jeito (interface, teclado, idioma, moeda, etc.)
- Reusável
  - Várias empresas precisam usar partes de um mesmo sistema

Qualidade de Software (para o Varejo)

- Aberto, compatível, de fácil integração com outros sistemas
  - A empresa já tem controle de estoque, fidelização, etc.
- Portável e independente de plataforma (hw e sw)
  - A empresa opta por uma determinada plataforma
- Baixo custo de instalação e atualização
  - A empresa tem um grande número de PDVs

Produtividade

- Custo de desenvolvimento reduzido
  - A empresa consumidora quer investir pouco em software
  - A empresa produtora tem que oferecer “software barato”
- Tempo de desenvolvimento reduzido
  - Suporte rápido às necessidades do mercado

“Software Barato”

Nem tanto resultado de baixos custos de desenvolvimento, mas principalmente da distribuição dos custos entre vários clientes.

Reuso, extensibilidade e adaptabilidade são essenciais para viabilizar tal distribuição.

Relevância da Engenharia de Software

- Qualidade de software e produtividade
- garantem:
  - Disponibilidade de serviços essenciais
  - Segurança de pessoas
  - Competitividade das empresas
    - Produtores
    - Consumidores

Mas, na realidade, temos a Crise de Software...

- 25% dos projetos são cancelados
- o tempo de desenvolvimento é bem maior do que o estimado
- 75% dos sistemas não funcionam como planejado
- a manutenção e reutilização são difíceis e custosas
- os problemas são proporcionais a complexidade dos sistemas

Causas da Crise de Software

- Essências
  - Complexidade dos sistemas
  - Dificuldade de formalização
- Acidentes
  - Má qualidade dos métodos, linguagens, ferramentas, processos, e modelos de ciclo de vida
  - Falta de qualificação técnica

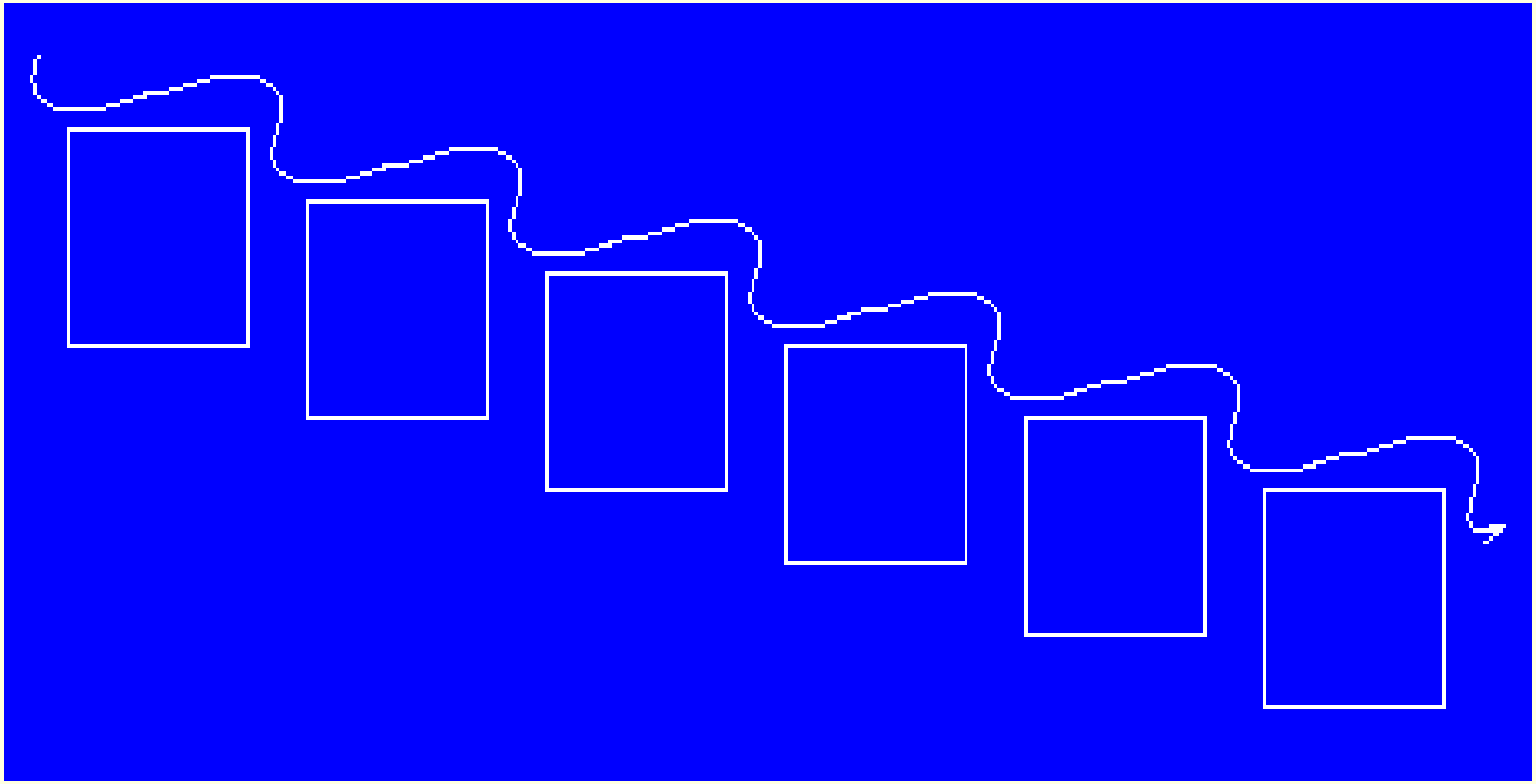
Elementos e Atividades da Engenharia de Software

- Elementos
  - Modelos do ciclo de vida do software
  - Linguagens
  - Métodos
  - Ferramentas
  - Processos
- Atividades
  - Modelagem do negócio
  - Elicitação de requisitos
  - Análise e Projeto
  - Implementação
  - Testes
  - Distribuição
  - Planejamento
  - Gerenciamento
  - Gerência de Configuração e Mudanças
  - Manutenção

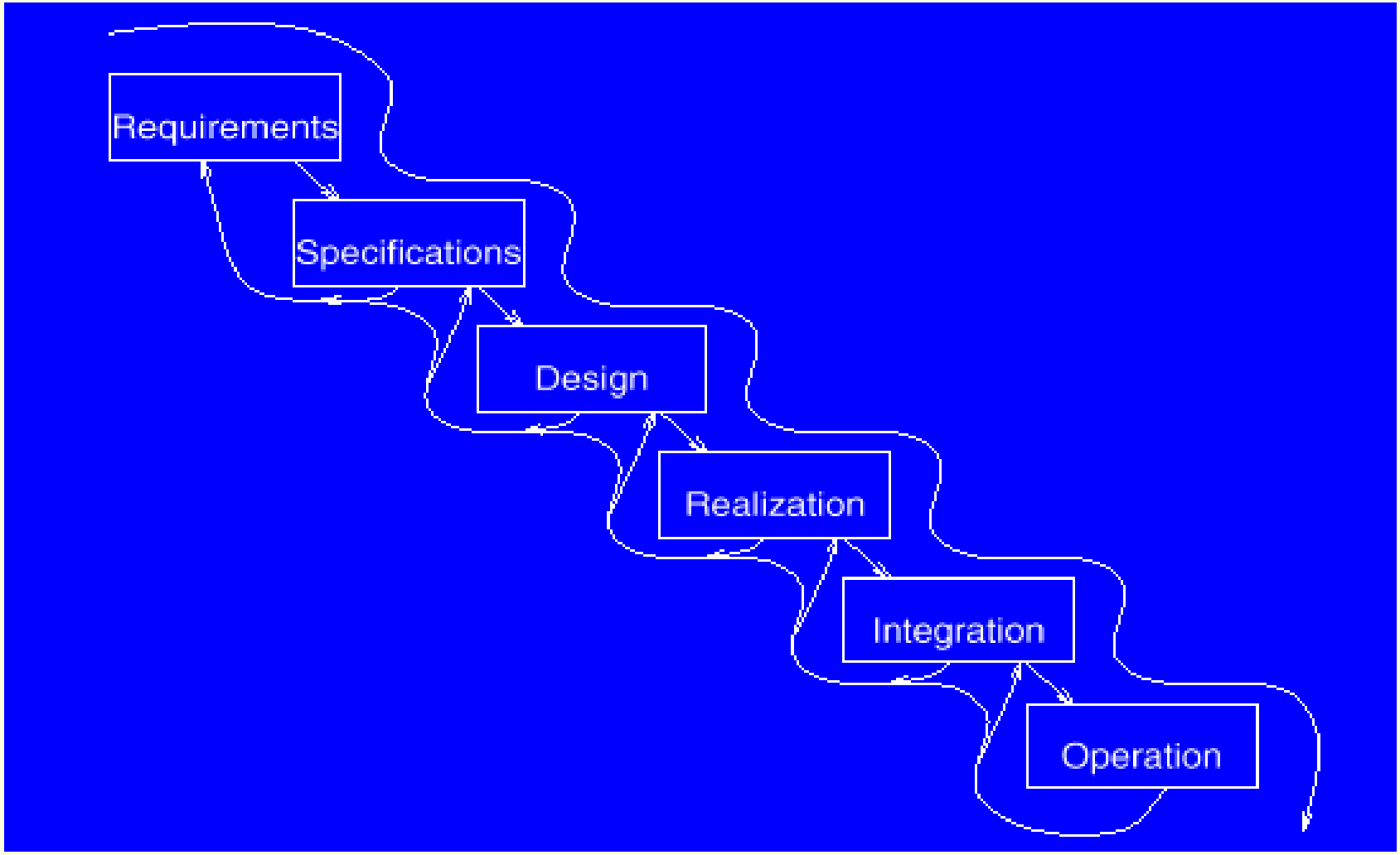
Modelos do Ciclo de Vida de Software

- Cascata
- Espiral
- Iterativo (do RUP)
- Prototipagem evolucionária
- Força bruta, gambiarra, hacking, ...

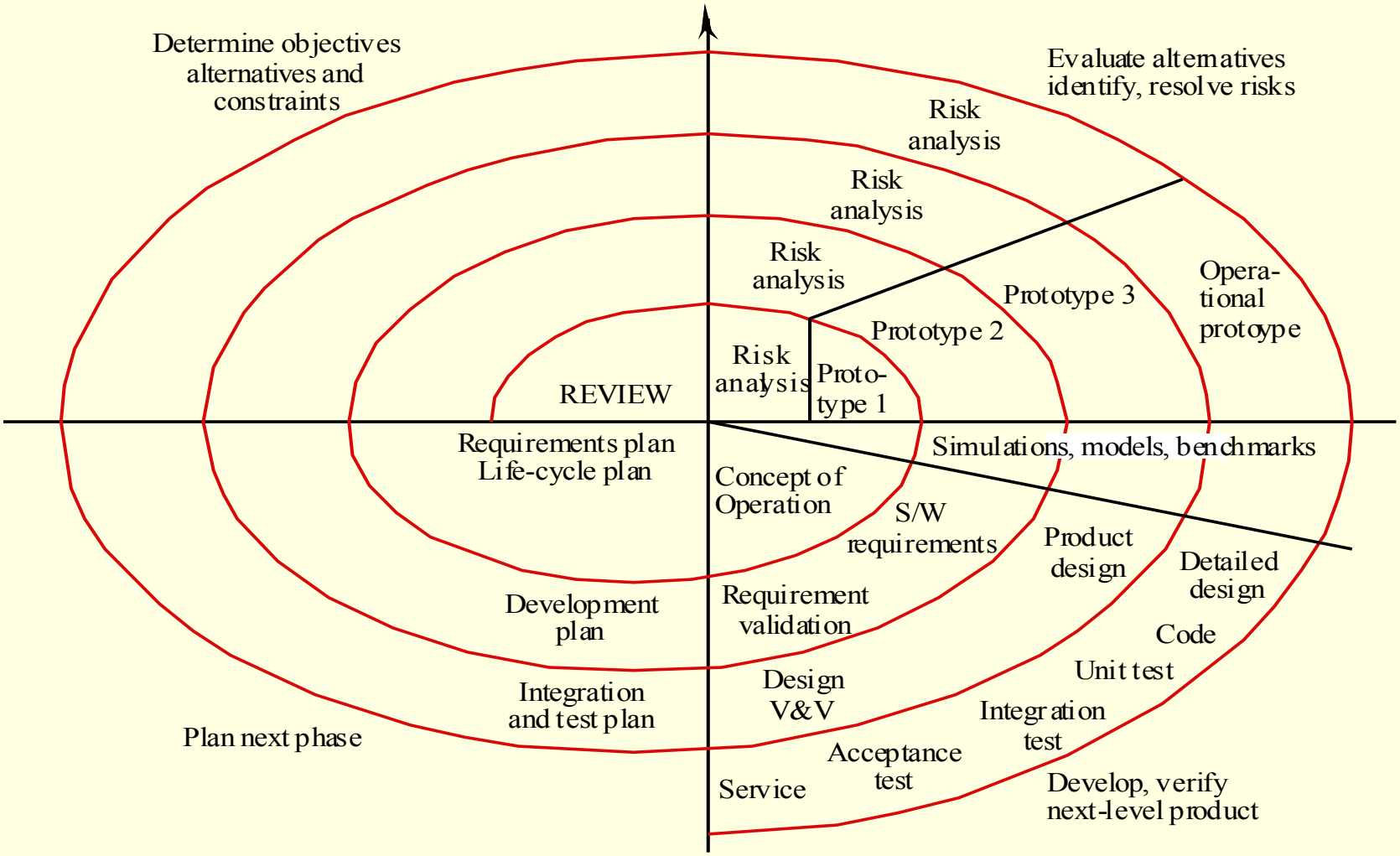
Modelo Cascata



Modelo Cascata na Prática



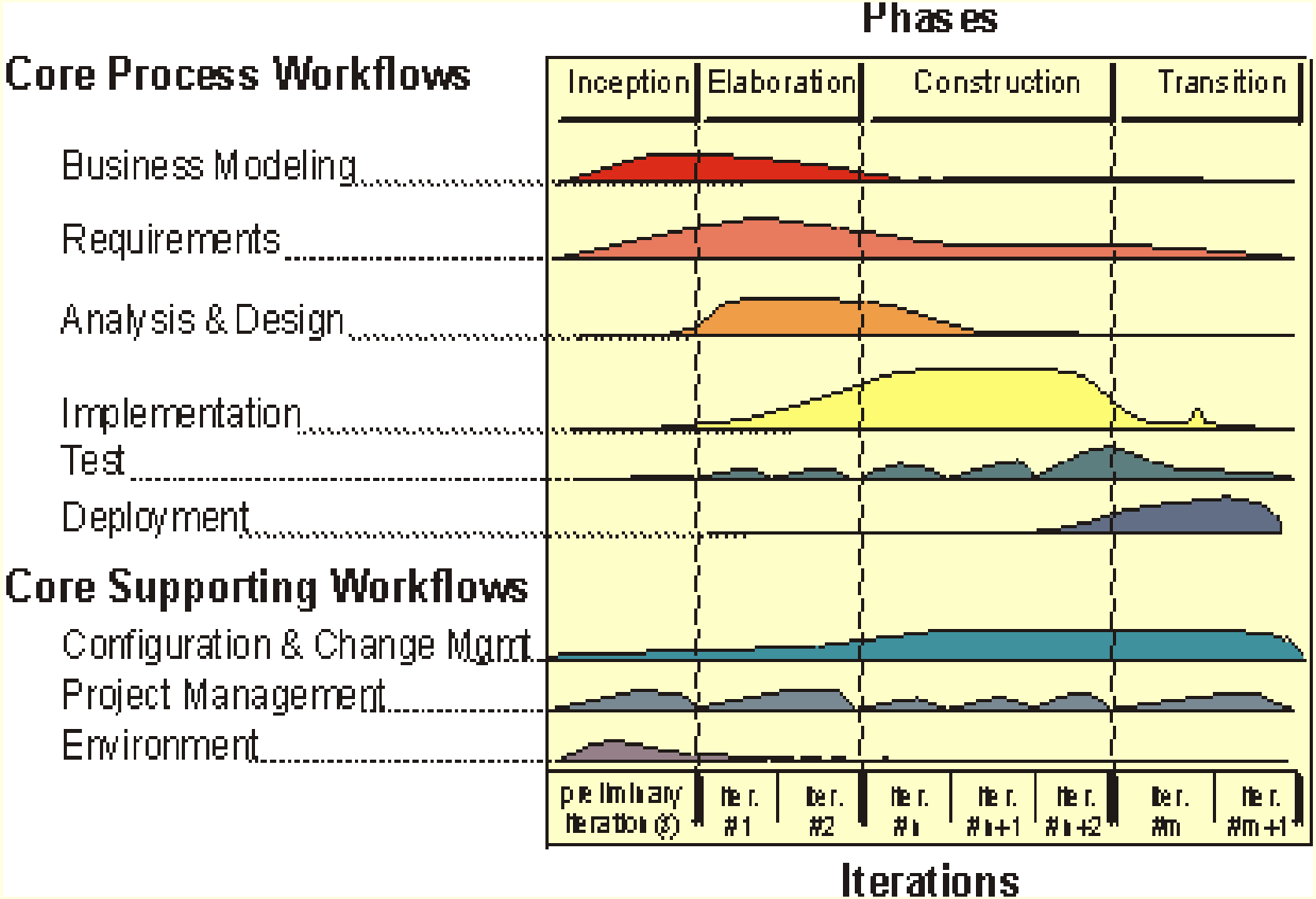
Modelo Espiral de Boehm (1988)



Fases do modelo Espiral

- Definição dos objetivos
  - Especificação dos objetivos específicos desta fase.
- Análise dos riscos
  - Identificação e solução dos principais riscos
- Desenvolvimento e validação
- Planejamento
  - O projeto é revisto e se define planos para a próxima “volta da espiral”

Modelo Iterativo (do RUP)



Linguagem

- Notação com sintaxe e semântica bem definidas
  - com representação gráfica ou textual
- Usada para descrever os artefatos gerados durante o desenvolvimento de software
- Exemplos: UML, Java

Método

- Descrição sistemática de como deve-se realizar uma determinada atividade ou tarefa
- A descrição é normalmente feita através de padrões e guias
- Metodologia
  - conjunto de métodos (+ processo)
- Exemplos: Booch, BON, Pim

Ferramenta

- Provê suporte computacional a um determinado método
- Ambiente de desenvolvimento: conjunto de ferramentas integradas (CASE)
- Exemplos: Rational Rose, Inprise JBuilder

Processo

- Conjunto de atividades
  - bem definidas
  - com responsáveis
  - com artefatos de entrada e saída
  - com dependências entre as mesmas e ordem de execução
  - com modelo de ciclo de vida

Em resumo...

- Resumo
  - Engenharia de software: objetivos e relevância
  - Produtividade e qualidade de software
  - Crise de software
  - Elementos e atividades da engenharia de software

Visão Geral do RUP

- Objetivo
  - Depois desta aula você terá uma visão geral do RUP, incluindo suas características e seus componentes principais.

O que é o RUP?

- O nome é uma abreviação de Rational Unified Process
  - mas na verdade é
    - Processo + Métodos + Linguagem (UML)
  - e os autores argumentam que é
    - *Framework para gerar processos*

O que é o RUP?

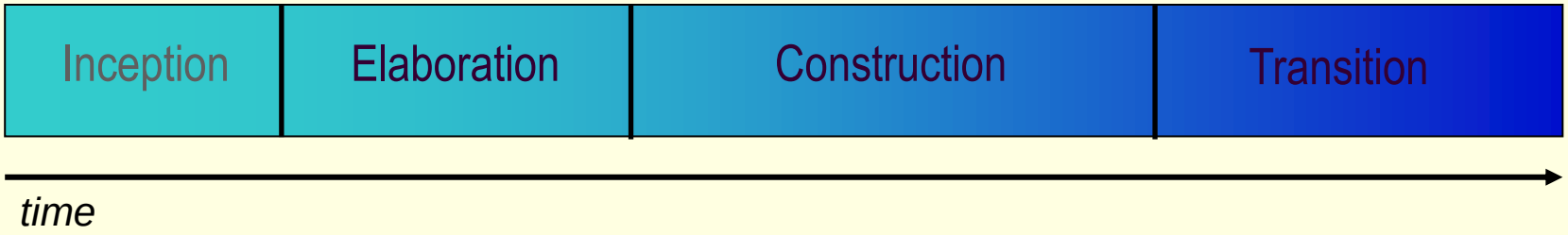
- Conjunto de atividades
  - bem definidas
  - com responsáveis
  - com artefatos de entrada e saída
  - com dependências entre as mesmas e ordem de execução
  - com modelo de ciclo de vida
  - descrição sistemática de como devem ser realizadas
  - UML

Características Principais do RUP

- O desenvolvimento de sistemas seguindo o RUP é
  - Iterativo e incremental
  - Guiado por casos de uso (use cases)
  - Baseado na arquitetura do sistema
- O RUP não é orientado a objetos?

O RUP é iterativo e incremental

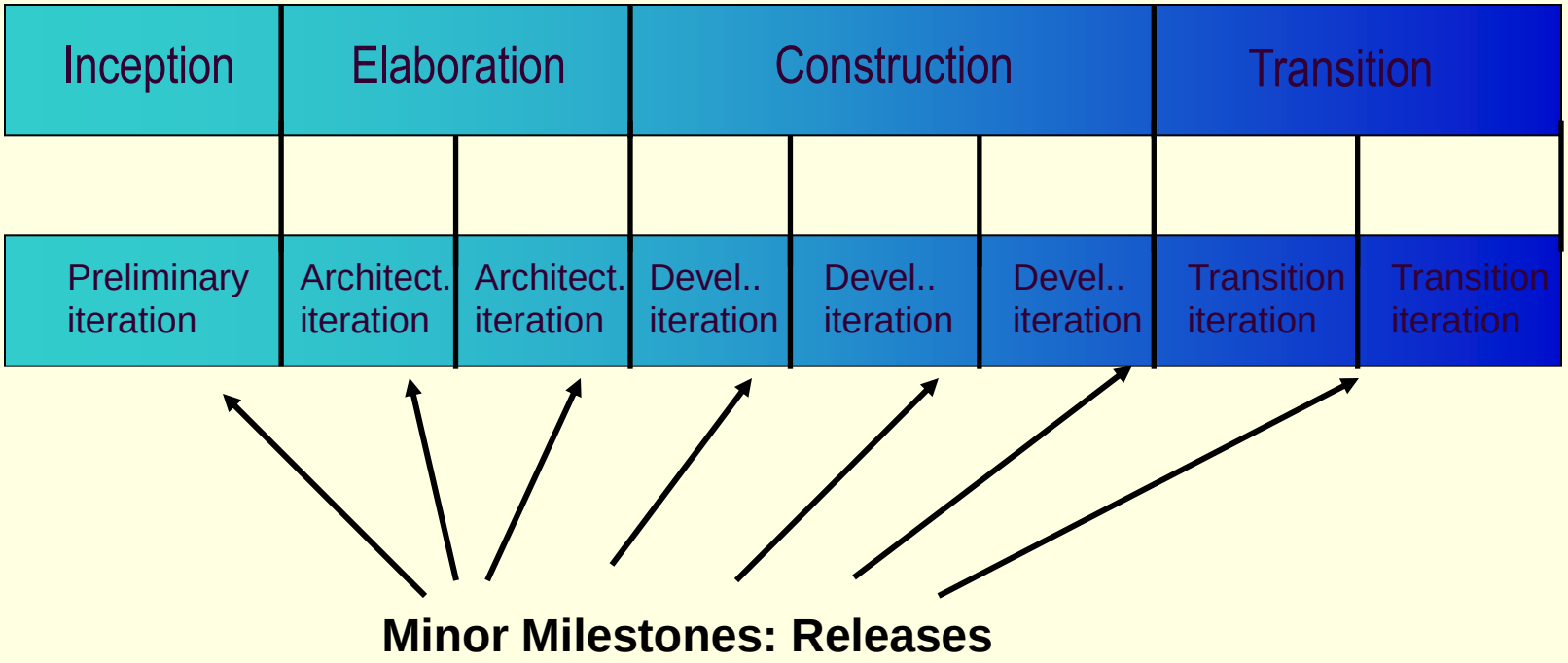
- O ciclo de vida de um sistema consiste de quatro fases:



  - Concepção (define o escopo do projeto)
  - Elaboração (define os requisitos e a arquitetura)
  - Construção (desenvolve o sistema)
  - Transição (implanta o sistema)

O RUP é iterativo e incremental

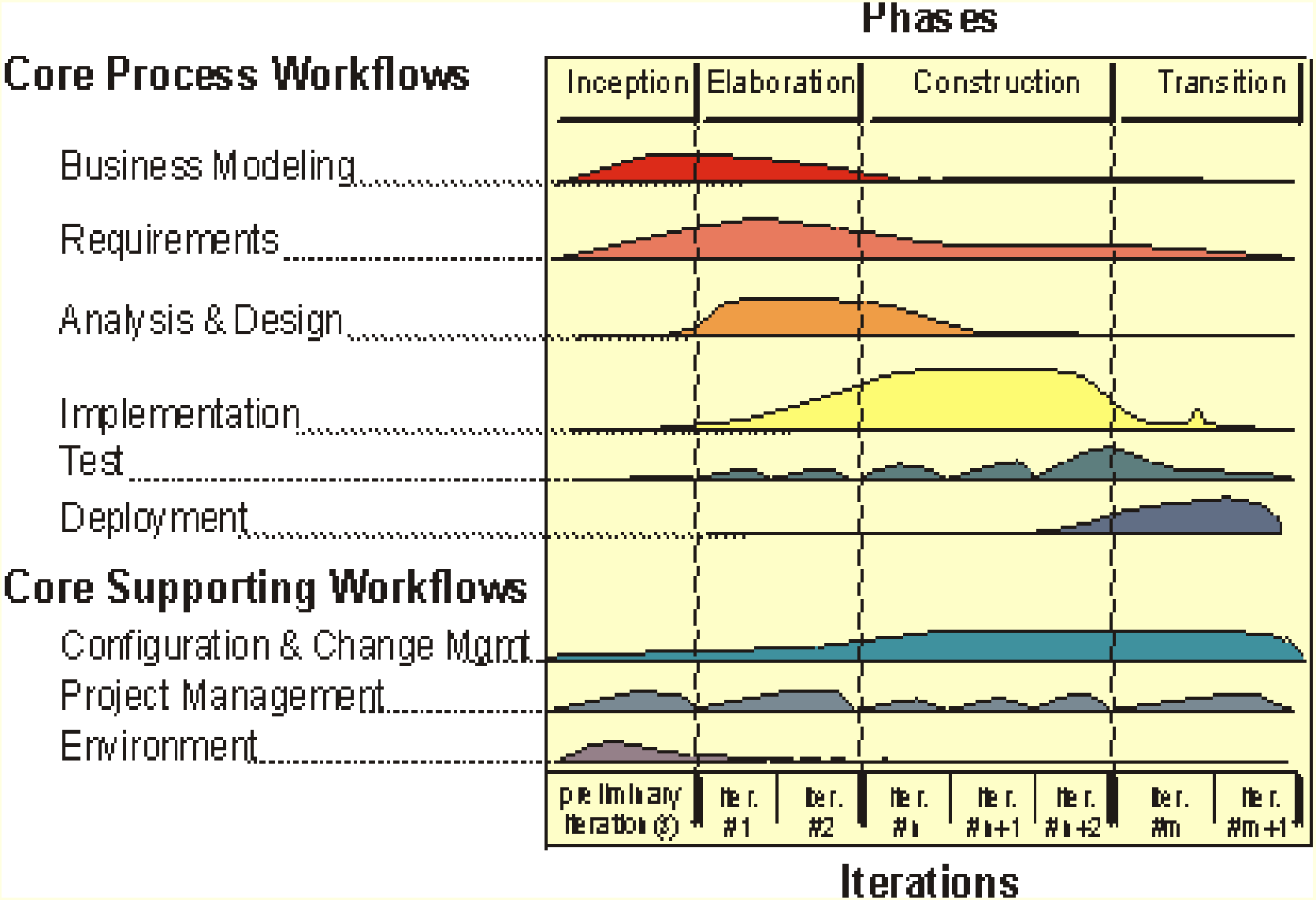
- Cada fase é dividida em iterações:



O RUP é iterativo e incremental

- Cada iteração
  - é planejada
  - realiza uma seqüência de atividades (de elicitação de requisitos, análise e projeto, implementação, etc.) distintas
  - resulta em uma versão executável do sistema
  - é avaliada segundo critérios de sucesso previamente definidos

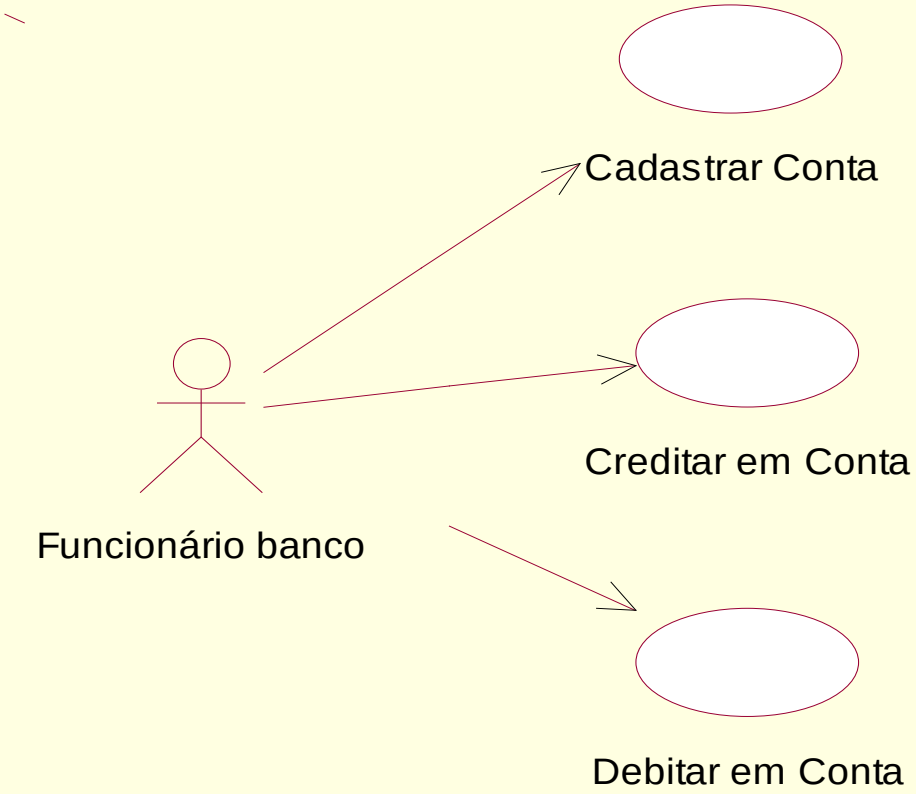
O RUP é iterativo e incremental



O RUP é guiado por casos de uso

- Atores e casos de uso
- Ator
  - algo que interage com o sistema
  - usuário, periférico, outro sistema, etc.
- Caso de uso
  - seqüência de eventos que o sistema realiza
  - gera resultados de interesse de algum **ator**

O RUP é guiado por casos de uso



O RUP é guiado por casos de uso

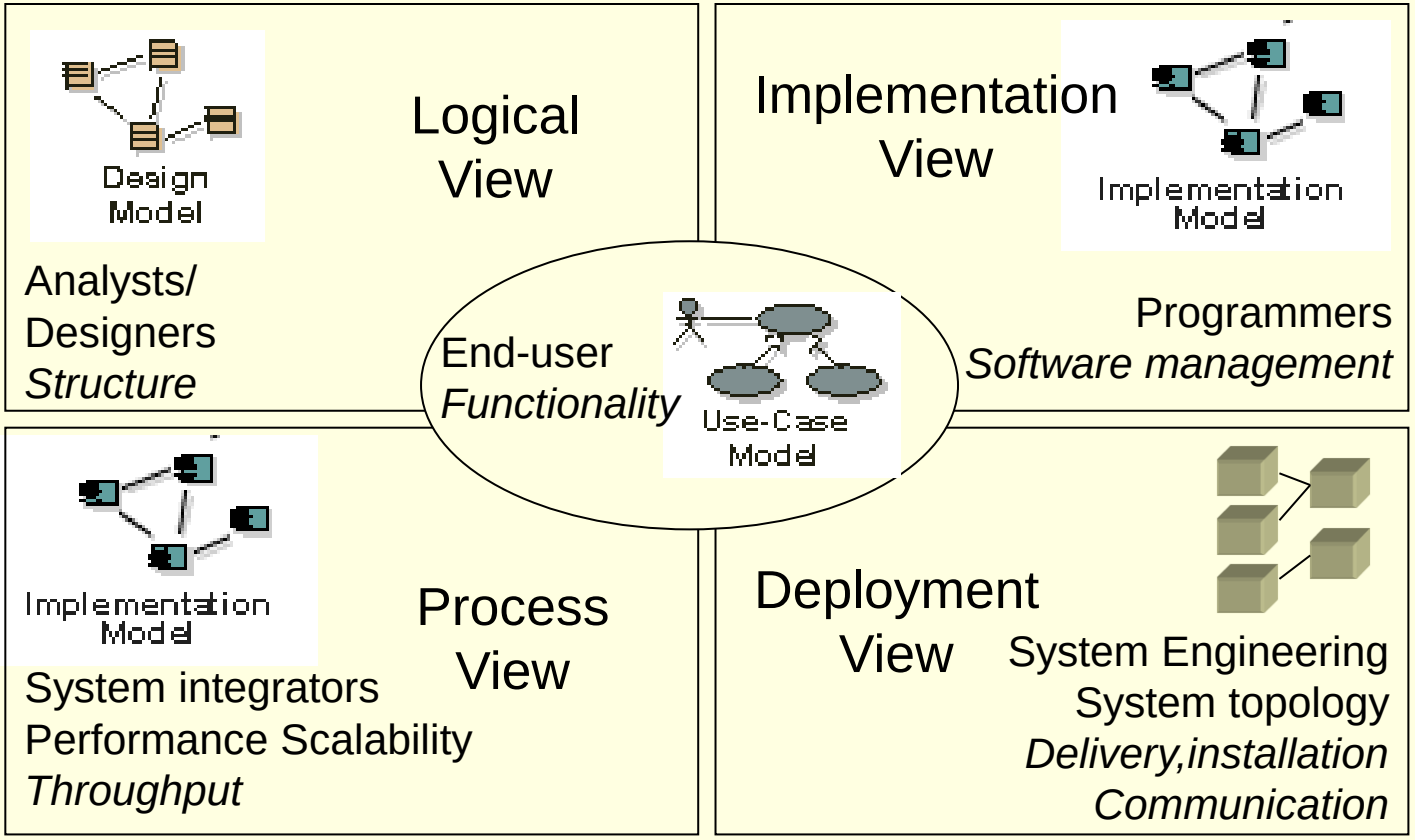
- Os casos de uso não servem apenas para definir os requisitos do sistema
- Várias atividades do RUP são guiadas pelos casos de uso:
  - planejamento das iterações
  - criação e validação do modelo de projeto
  - planejamento da integração do sistema
  - definição dos casos de teste

O RUP é baseado na arquitetura do sistema

- Arquitetura
  - visão geral do sistema em termos dos seus subsistemas e como estes se relacionam
- A arquitetura é prototipada e definida logo nas primeiras iterações
- O desenvolvimento consiste em complementar a arquitetura
- A arquitetura serve para definir a organização da equipe de desenvolvimento e identificar oportunidades de reuso

O RUP é baseado na arquitetura do sistema

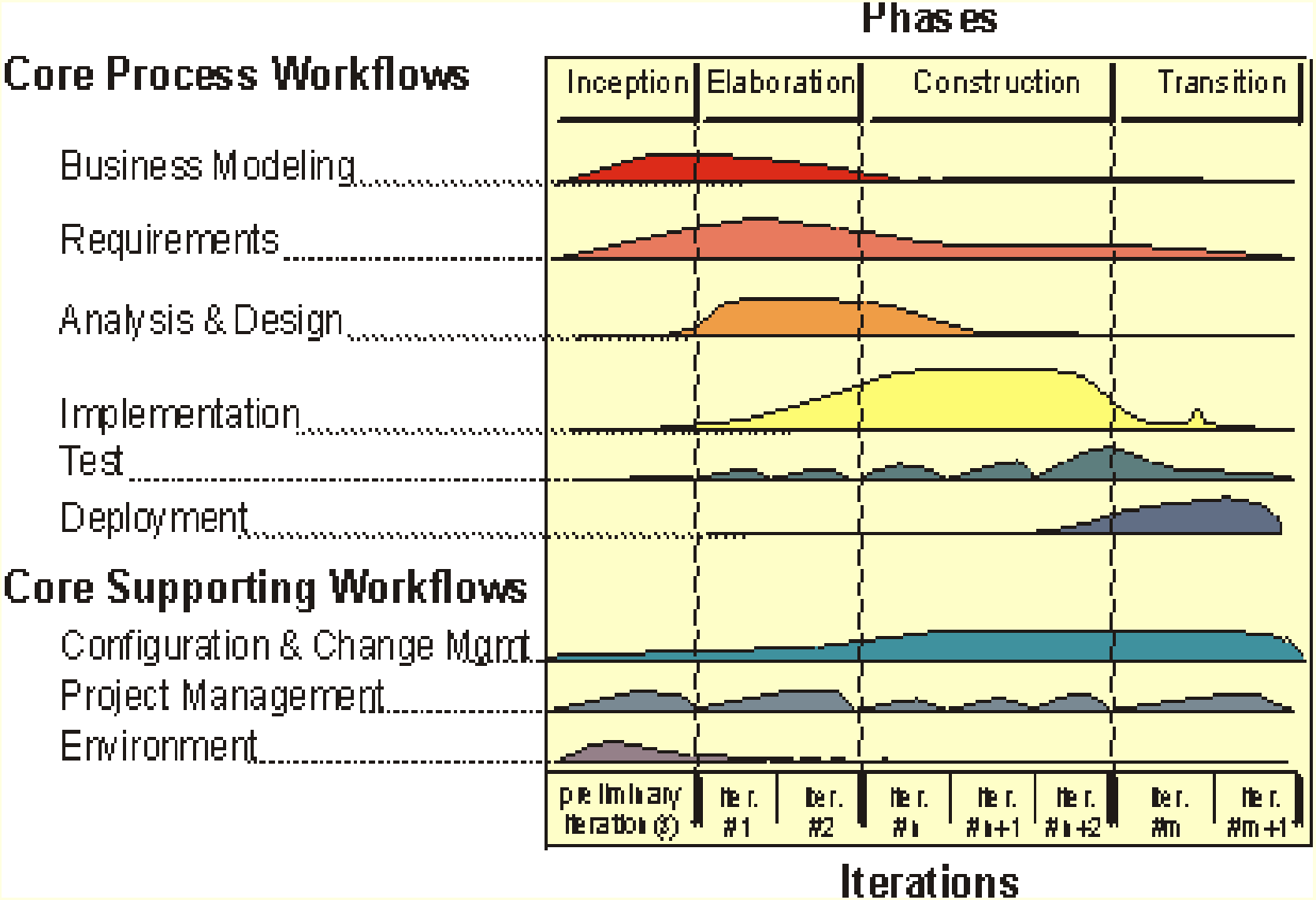
- Idealmente, tem-se 5 visões da arquitetura



Overview do Processo

- O processo pode ser descrito em duas dimensões, ou ao longo de dois eixos:
- O eixo horizontal representa o tempo e mostra os aspectos dinâmicos do processo - Ciclos, Fases, Iterações e Marcos.
- O eixo vertical representa o aspecto estático do processo - Atividades, Artefatos, Workers e Workflows.

O RUP é iterativo e incremental



Fases e Iterações

- O processo de desenvolvimento de um SW deve ser quebrado em partes da aplicação, que, através de um processo iterativo, são disponibilizadas progressivamente para a efetiva utilização do usuário final, promovendo inclusive o reuso de componentes já definidos.

Fases e Iterações

- O desenvolvimento de um SW é dividido em ciclos, cada um destes gerando um subconjunto do SW final.
- Cada ciclo é dividido em 4 (quatro) fases consecutivas, com objetivos específicos, sendo que cada uma delas pode apresentar várias iterações.

Fases e Iterações

- Cada fase é concluída com um Marco (Miles-tone) que é um momento em que decisões críticas devem ser tomadas e objetivos chave devem ter sido alcançados.
- As fases são:
  - Inicial ou Princípio (Inception);
  - Elaboração (Elaboration);
  - Construção (Construction);
  - Transição (Transiction).

Fase Inicial

- Identifica o “business case”
- Delimita o escopo do projeto

Principais Produtos:

- Um documento com a visão geral da essência do projeto, características chave e as principais restrições;
- A modelagem inicial de um "use case";
- Um glossário do projeto inicial;
- O "business case" inicial que inclui o contexto do negócio, critérios de sucesso (projeção de renda, reconhecimento de mercado, etc) e previsão financeira;

Principais Produtos:

- Uma avaliação inicial de riscos;
- Um planejamento do projeto, mostrando fases e iterações;
- O modelo de negócios;
- Um ou vários protótipos.

Objetivos do Ciclo de Vida

- Aceitação da instância decisória no que se refere a definição do escopo e estimativa e escalonamento de custos;
- Compreensão de requisitos evidenciados pelos casos primários de usuários;
- Credibilidade para as estimativas de custo e escalonamento, prioridades, riscos e processo de desenvolvimento;

Objetivos do Ciclo de Vida

- Profundidade e largura de qualquer protótipo que tenha sido desenvolvido;
- Despesas reais versus despesas planejadas.

Fase de Elaboração

- Analisa o domínio do problema;
- Estabelece as bases da arquitetura ser utilizada;
- Desenvolve o planejamento do projeto;
- Elimina os maiores risco;
- Delimita o escopo do projeto;

Principais Produtos:

- Um modelo de “use case” com a identificação de todos os atores e as descrições de suas interações;
- Identificação de requisitos suplementares do aspecto não funcional do projeto (P.Ex: desempenho) ou não associados aos “use case”;
- Uma descrição da arquitetura de “software”;
- Um protótipo executável da arquitetura;

Principais Produtos:

- Uma lista de riscos revisada e um “business case” revisado;
- Um plano de desenvolvimento para todo o projeto, incluindo um plano de projeto detalhado, mostrando as iterações e os seus critérios de avaliação;
- Um caso de desenvolvimento atualizado especificando o processo a ser utilizado;
- Um manual de usuário preliminar.

Marco: Arquitetura do Ciclo de Vida

- A visão do produto está estável?
- A arquitetura está estável?
- A demonstração executável mostra que os elementos de risco principais foram abordados e resolvidos com credibilidade?
- O planejamento para a fase de construção estão suficiente detalhados e precisos? Está suportado por uma estimativa precisa?

Marco: Arquitetura do Ciclo de Vida

- Todos os responsáveis pelas decisões concordam que a visão atual pode ser atingida se o planejamento atual for executado para o desenvolvimento do sistema completo, no contexto da arquitetura atual?
- As despesas reais versus as planejadas são aceitáveis?

Fase de Construção

- Sistema deve ser capaz de operar no ambiente do usuário (iterações e incremento que ao longo da fase torna evidente a viabilidade do SW)
- Desenvolvimento do SW

Principais Produtos:

- O “software” integrado às plataformas adequadas;
- Os manuais de usuário;
- Uma descrição da versão atual.

Marco: Capacidade Operacional Inicial

- A versão atual do produto está estável e madura o suficiente para ser disponibilizada para a comunidade usuária?
- Os responsáveis pela decisão estão prontos para a transição para a comunidade usuária?
- A comparação entre as despesas reais e as planejadas é aceitável?

Fase de Transição

- Atingir a capacidade final da operação

- Garantir que o produto está pronto para ser entregue ao usuário

Características

- “Beta” teste para validar o novo sistema em confronto com as expectativas dos usuários;
- Operação em paralelo com o sistema que está sendo substituído;
- Conversão das bases de dados;

Características

- Treinamento dos usuários e administradores;
- Apresentação do produto para as equipes de marketing, distribuição e vendas.

Principais Produtos:

- Autosuficiência dos usuários;
- Concordância da instância decisória em que o produto disponibilizado está completo e consistente com os critérios de avaliação da visão;
- Obter o produto final com a maior velocidade e o controle de custos tanto quanto possível.

Marco: Versão de Produto

- O usuário está satisfeito?
- As despesas realizadas são aceitáveis diante das planejadas?

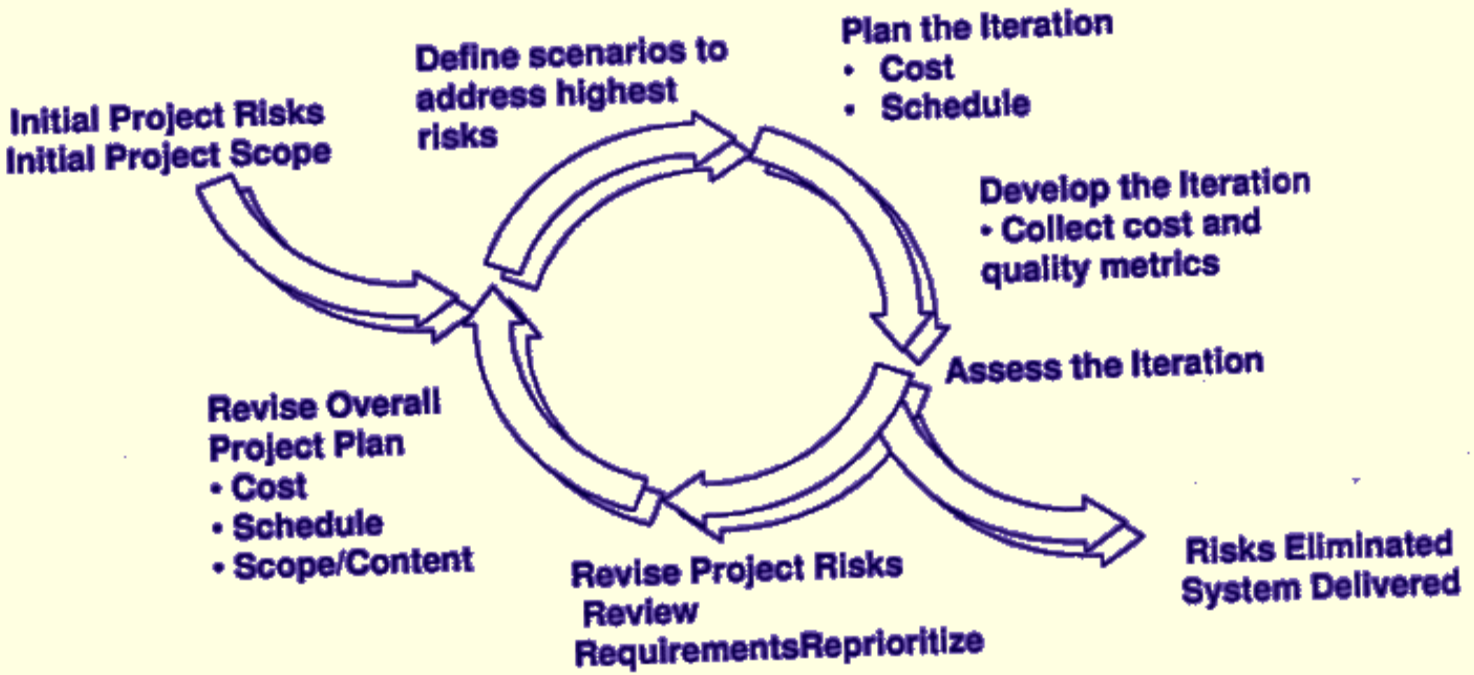
Cronograma X Esforço

	Inception	Elaboration	Construction	Transition
Effort	**~5 %**	**20 %**	**65 %**	**10%**
Schedule	**10 %**	**30 %**	**50 %**	**10%**

Iterações

- É um laço de desenvolvimento completo resultando em uma versão (interna ou externa) de um produto executável, um subconjunto do produto fina em desenvolvimento, que cresce incrementalmente de iteração para iteração para tornar-se o produto final

Iterações


Initial Project Risks
Initial Project Scope
Define scenarios to address highest risks
Plan the Iteration
• Cost
• Schedule
Develop the Iteration
• Collect cost and quality metrics
Assess the Iteration
Revise Overall Project Plan
• Cost
• Schedule
• Scope/Content
Revise Project Risks
Review RequirementsReprioritize
Risks Eliminated
System Delivered

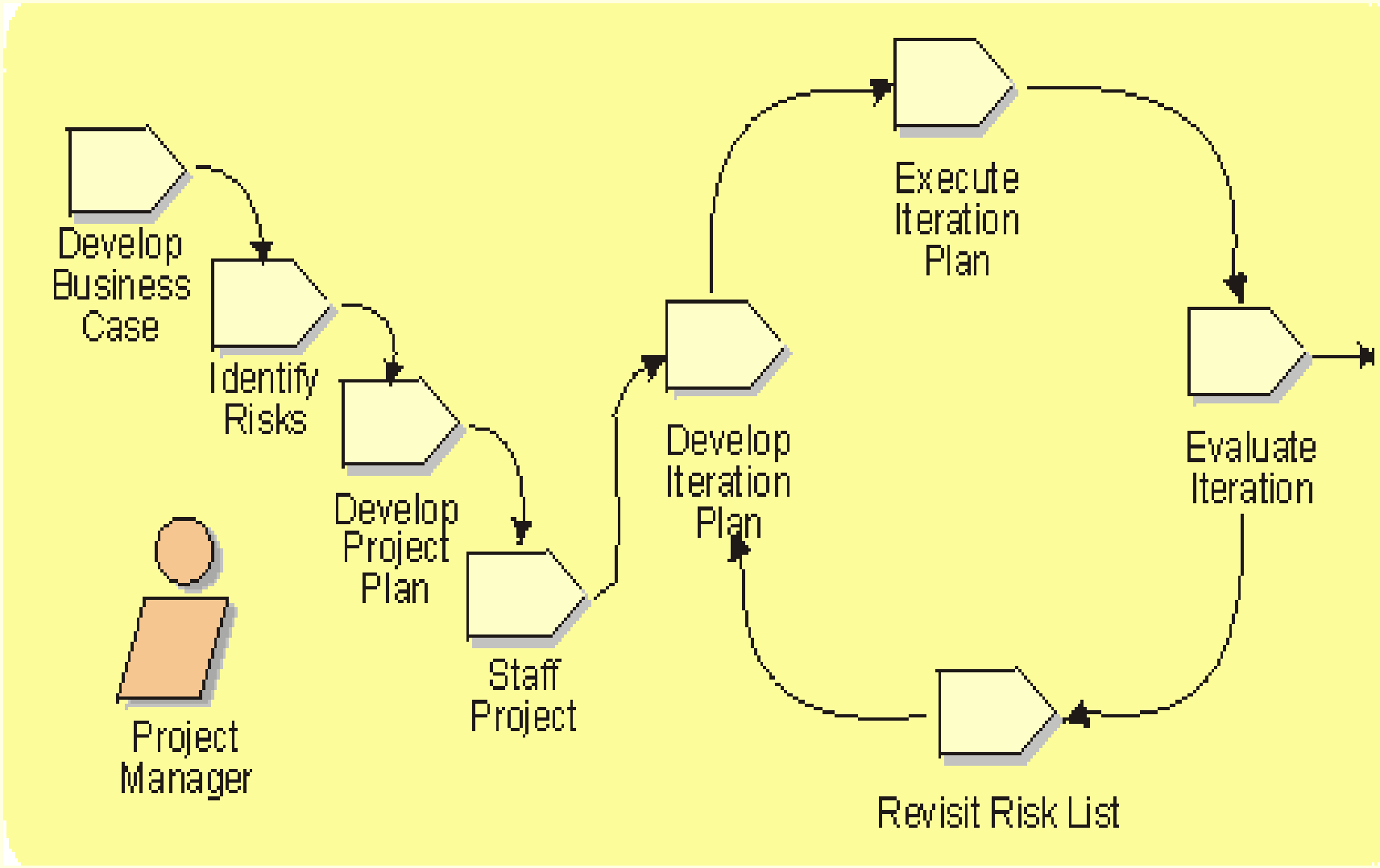
Benefícios da Abordagem Iterativa

- Os riscos são minimizados mais cedo;
- As mudança são mais gerenciáveis;
- Alto nível de reutilização;
- A equipe de projeto podem aprender ao longo do processo;
- Melhor qualidade geral.

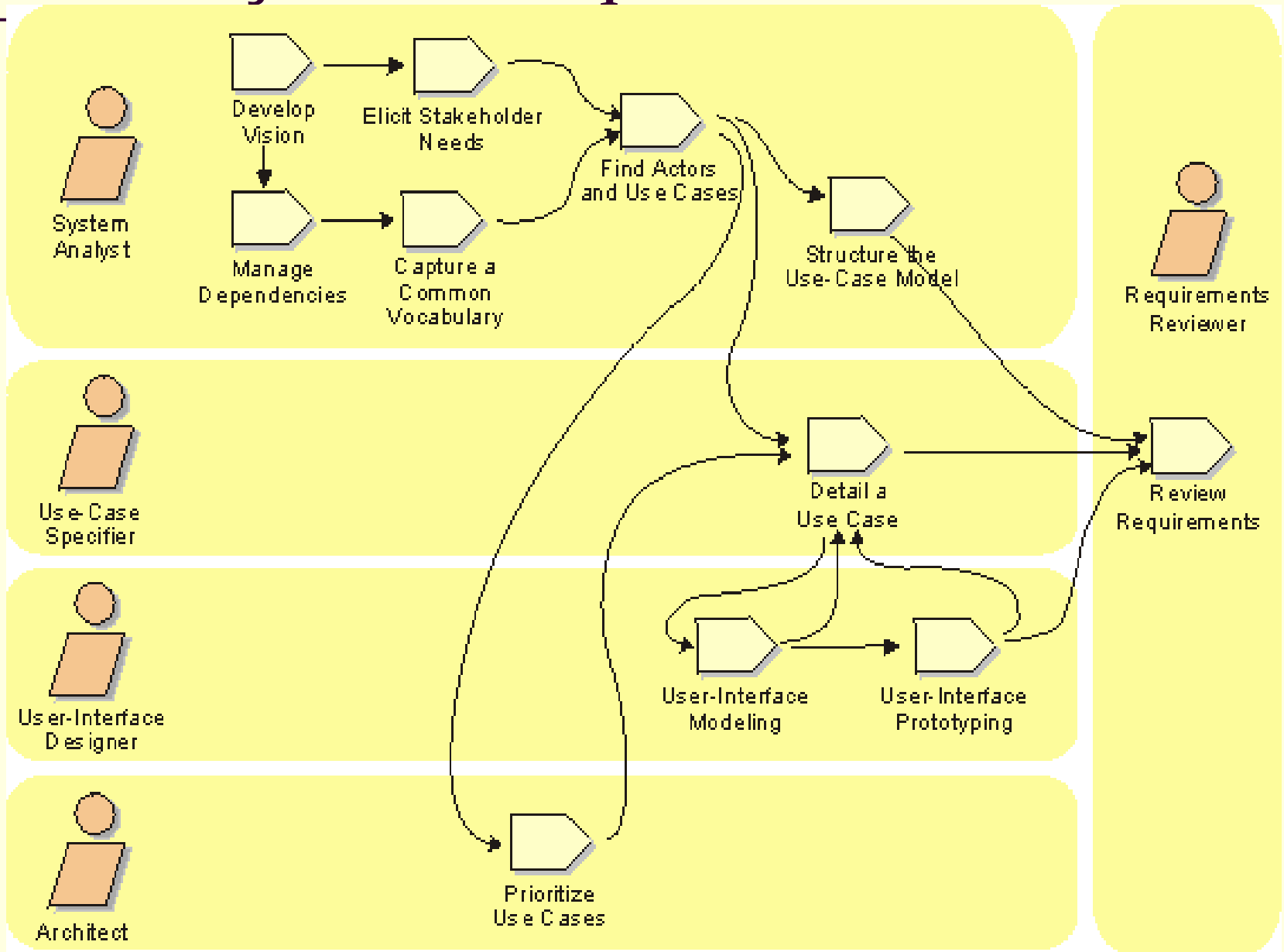
Organização do RUP

- Fluxos de atividades
- Atividades
  - passos
  - entradas e saídas
  - guias (de ferramentas ou não), *templates*
- Responsáveis (papel e perfil, não pessoa)
- Artefatos

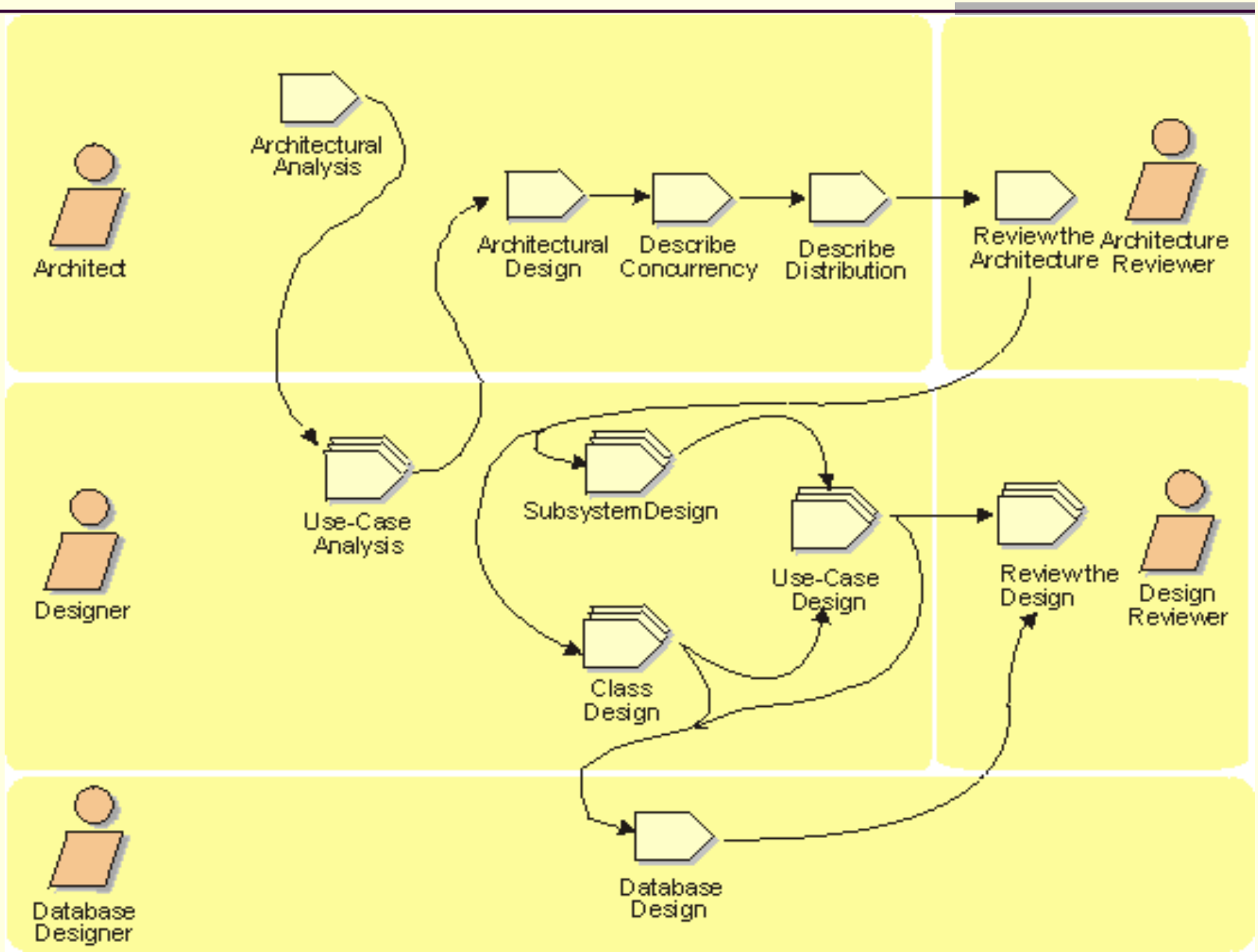
Planejamento e Gerenciamento



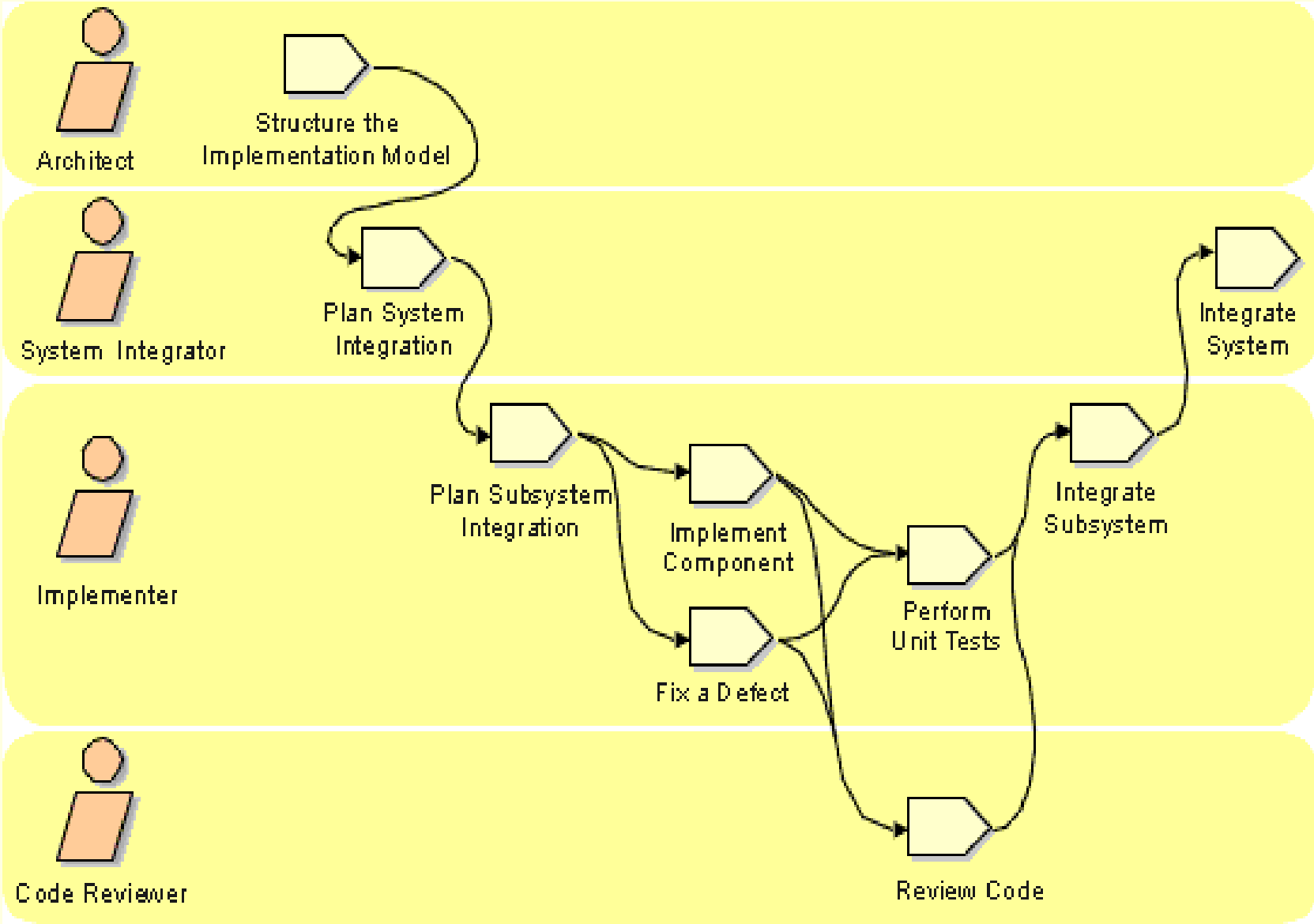
Elicitação de Requisitos



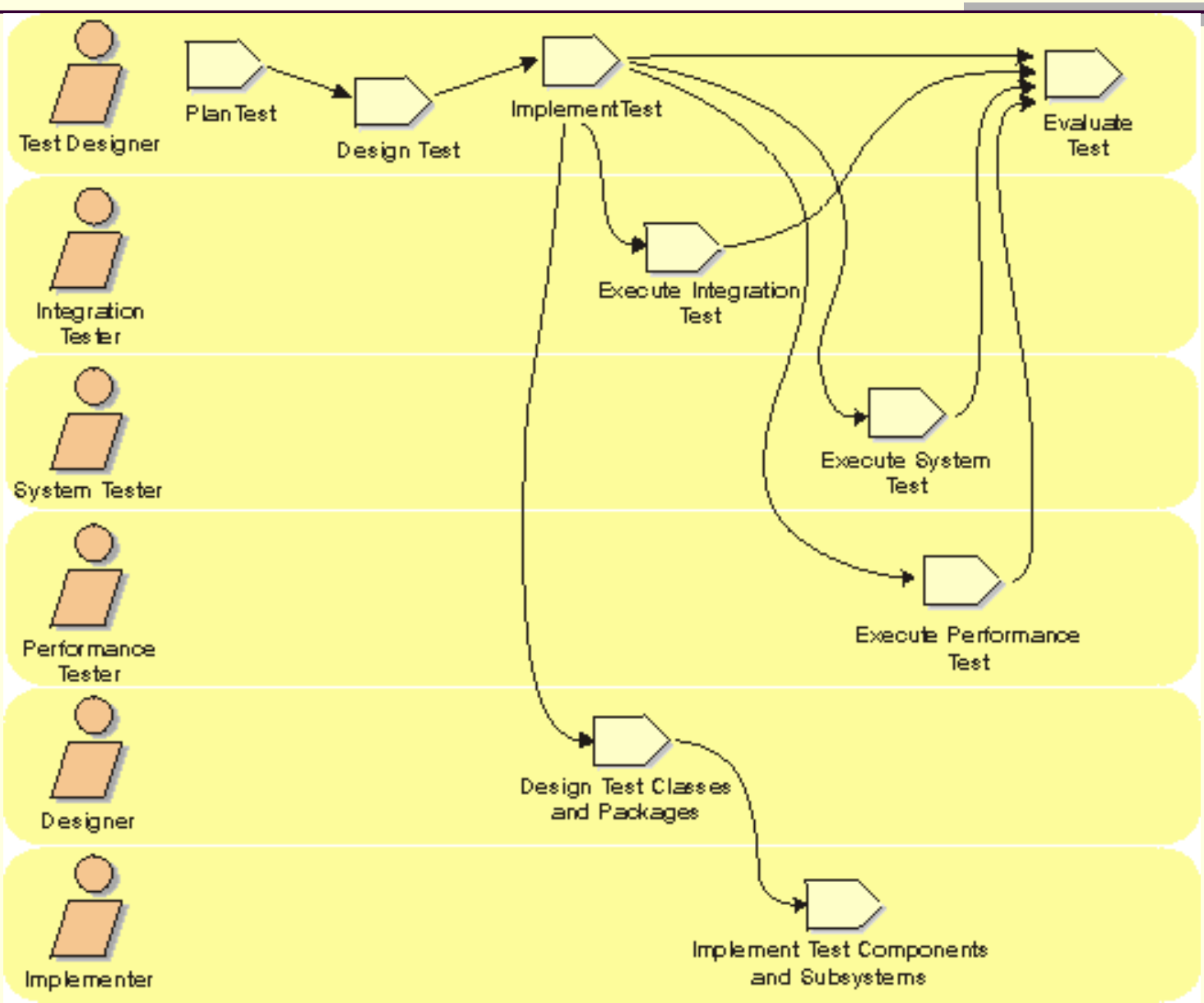
Análise e Projeto



Implementação



Testes


Test Designer
Plan Test
Design Test
Implement Test
Evaluate Test
Integration Tester
Execute Integration Test
System Tester
Execute System Test
Performance Tester
Execute Performance Test
Designer
Design Test Classes and Packages
Implementer
Implement Test Components and Subsystems

Visão Geral do RUP

- Resumo
  - O RUP é iterativo e incremental
  - O RUP é guiado por casos de uso
  - O RUP é baseado na arquitetura do sistema
  - Organização do RUP e componentes principais