Processo Unificado para Desenvolvimento de Software

Engenharia de Software - Processo Unificado – Prof. Antonio Guardado

Assuntos discutidos

- 1 - Breve histórico
- 2 - Descrição sucinta
- 3 - Fases do ciclo de vida
- 4 - Artefatos do processo
- 5 - Fluxos de Trabalho

1 - Breve Histórico

- Origem nas experiências de projeto de sistema da Ericsson, comandado por Jacobson (1967)
- Objectory Process
  - Objectory: originado de Object Factory
  - Objectory AB: empresa de Jacobson
  - Objectory Process: versões 1.0 a 3.8 (1987 a 1995)

1 - Breve Histórico (2)

- Rational Objectory Process (ROP)
  - compra de Objectory AB pela Rational (1995)
  - ROP versão 4.1: resultado da integração de Objectory 3.8 e experiência da Rational
- Rational Unified Process (RUP)
  - RUP versão 5.0 em junho de 1998
  - baseado em UML
  - incorporação de experiências de vários métodos
- Rational vendida para a IBM - 2003

2 - Processo Unificado

- O Processo Unificado é uma estrutura (*framework*) de processo de desenvolvimento de software que pode ser especializado para:
  - diferentes classes de sistemas de software
  - diferentes áreas de aplicação
  - diferentes tipos de organização
  - diferentes níveis de competências
  - diferentes portes de sitemas

2.1 - Características do Processo Unificado

- É baseado em casos de uso: os casos de uso direcionam as atividades de desenvolvimento.
- É centrado em arquitetura: a arquitetura é estabelecida no início do desenvolvimento e refinada ao longo das atividades.
- É iterativo e incremental: o produto é desenvolvido por partes, de modo a acrescentar requisitos de forma controlada.

2.1.1 – Orientado (dirigido) a Casos de Uso

Cada *use-case* representa uma sequência de **ações** que produzem um **resultado** de utilidade para um **ator**

Um **ator** é uma pessoa ou outro sistema

Cada ***use-case*** é um requisito funcional do sistema

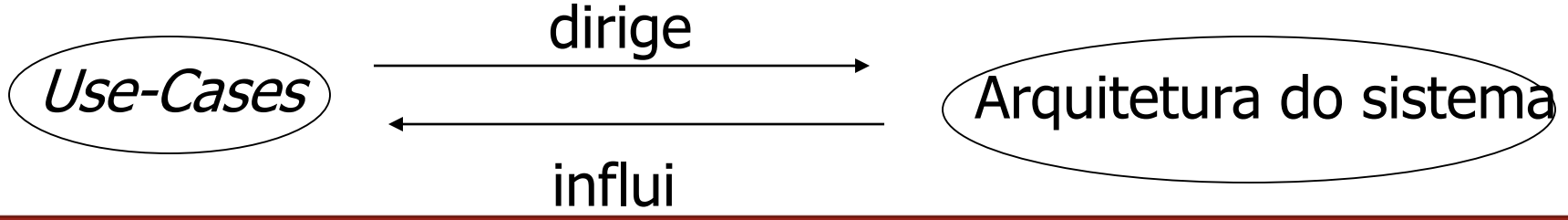
Modelo *Use-Case* = Σ *use-cases* = funcionalidade do sistema

Os *Use-Cases* acompanham todo o processo de desenvolvimento: Especificação de Requisitos, Análise, Projeto, Implementação e Testes

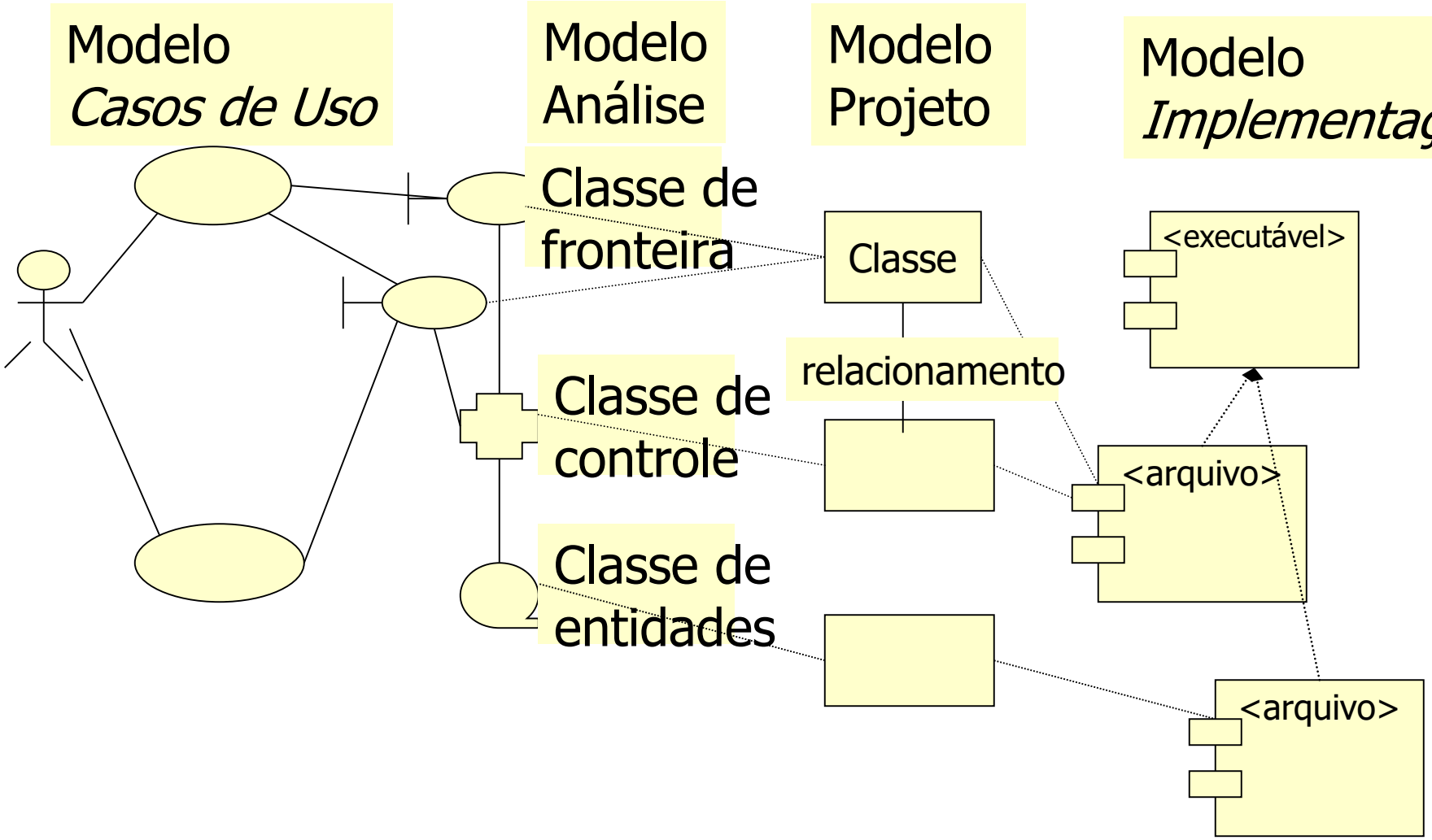
2.1.1 – Orientado a Casos de Uso

Porque USE-CASES??

- **Capturar os requisitos**: o *Diagrama Use-Case* mostra quais **atores** usam quais ***use-cases***
- **Dirigir o processo**: para realizar os *use-cases* são definidos *classificadores* (classes, subsistemas, interfaces) e relacionamentos (colaborações) entre estes
- **Elaborar a arquitetura, determinar iterações, determinação dos manuais de usuário**



2.1.1 – Orientado a Casos de Uso



2.1.1 – Orientado a Casos de Uso

↓Dirigir o processo:

ANÁLISE: definir as CLASSES DE ANÁLISE para realizar um *use-case* (classes limite, classes de controle e classes de entidades)

PROJETO: ENTRADA: modelo de análise, pacote de construção de interfaces, SGBD, sistemas existentes.
SAIDA: **classificadores** (classes, subsistemas, interfaces), **relacionamentos** e **colaborações**

IMPLEMENTAÇÃO:criação de **programas executáveis** e **arquivos** que realizam os *use-cases*

TESTE: **casos de teste** e **procedimentos de teste.** Testar entradas, resultados e condições

2.1.2 – Centrado em Arquitetura

DECISÕES SOBRE:

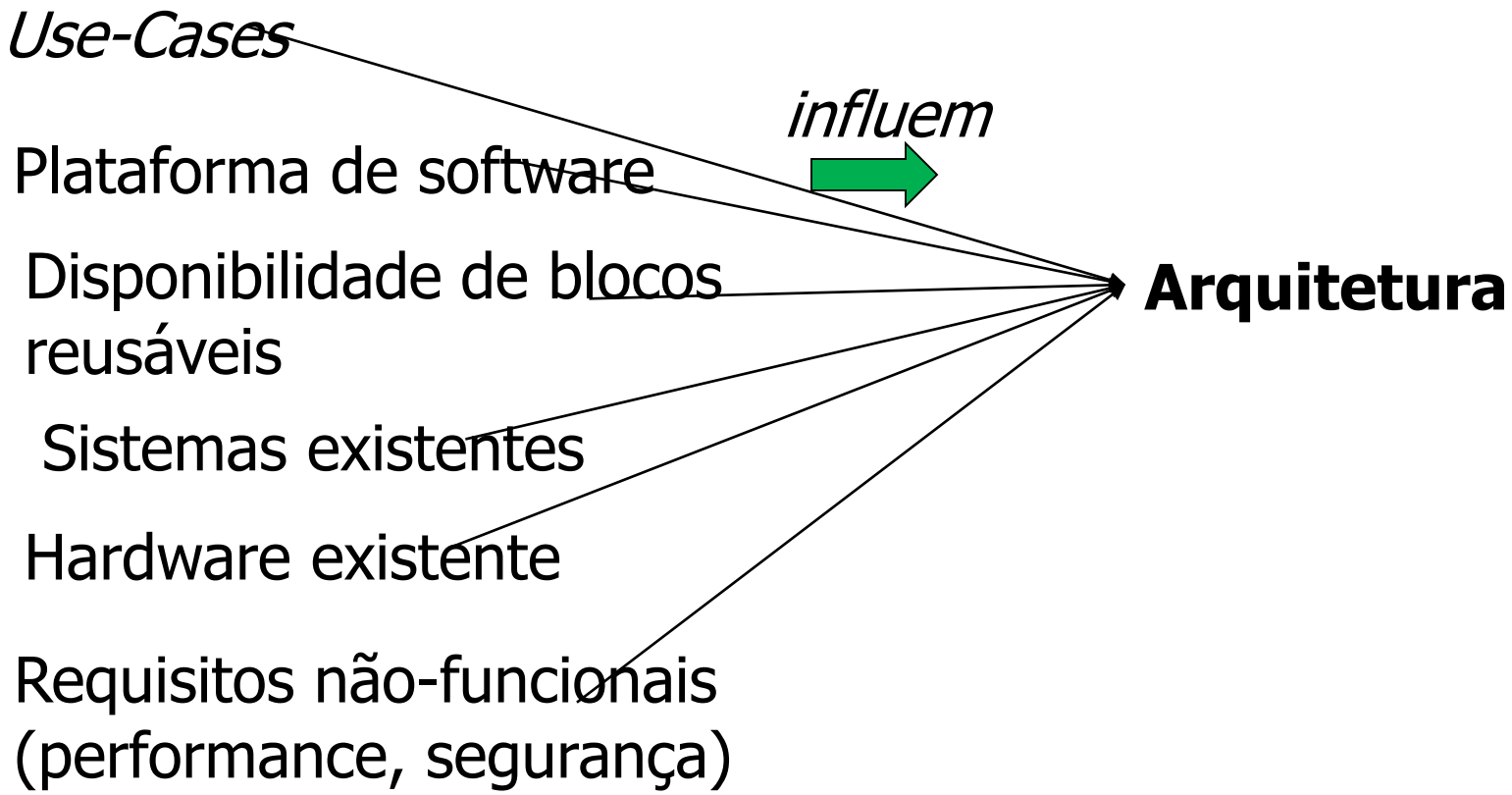
- a organização do sistema como um todo
- os elementos estruturais, interfaces e seu comportamento
- composição de elementos estruturais e comportamentais em subsistemas

A ARQUITETURA descreve as **partes essenciais** do sistema, importantes para todos desenvolvedores

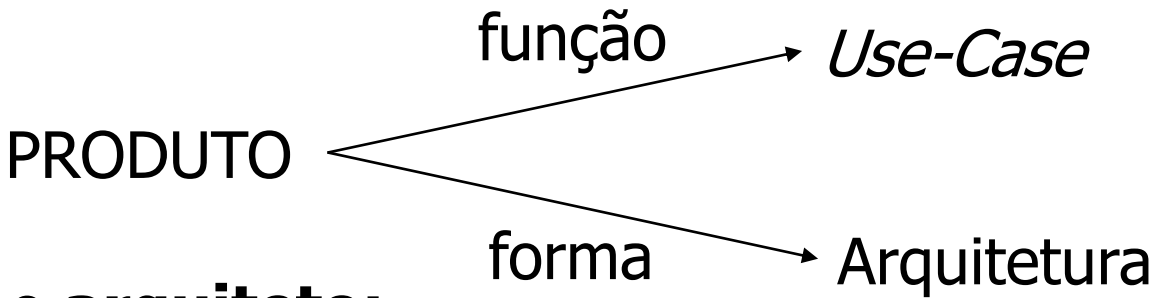
Menos de 10% das classes são relevantes para a arquitetura

Descrição de REQUISITOS ADICIONAIS: segurança, distribuição, concorrência, plataformas, etc.

2.1.2 – Centrado em Arquitetura



2.1.2 – Centrado em Arquitetura



Sequência para o **arquiteto:**

- Criar uma visão preliminar da arquitetura
- Analisar os ***use-cases* chave** (5-10%) e especificar um **subsistema** para cada um
- Pela especificação dos subsistemas aparecem mais detalhes da arquitetura e novos *use-cases*
- Repetir o passo acima, até terminar o sistema

2.1.3 – Iterativo e Incremental

Projetos grandes são divididos em **mini-projetos**

Cada mini-projeto é uma **iteração**

Cada iteração gera um **incremento**

PORQUE ITERATIVO E INCREMENTAL
- atenuar riscos
- obter arquitetura robusta
- facilitar alterações táticas ou contínuas
- conseguir aprendizado mais cedo

- Um grupo de *use-cases* que estendem usabilidade

Fatores de risco mais importantes

→ MINI-PROJETO

2.1.3 – Iterativo e Incremental

- Cada iteração contém as atividades de análise de requisitos, projeto, codificação, teste de unidade, teste de integração e teste de sistema, em variadas proporções.
- Utiliza UML para a representação do sistema.

2.2 - Elementos do Processo Unificado

- **Pessoas**: participantes do projeto de software
- **Projeto**: elemento que gerencia um desenvolvimento de software
- **Produto**: artefatos criados durante o projeto
- **Processo**: conjunto completo de atividades para transformar os requisitos em produto

P4 = Pessoa, Projeto, Produto, Processo

PESSOAS financiam, escolhem, desenvolvem, gerenciam, testam, usam e são beneficiadas por produtos

PROJETOS sofrem alterações. Determinam os tipos de *pessoas* que irão trabalhar no projeto e os *artefatos* que serão usados

P4 = Pessoa, Projeto, Produto, Processo

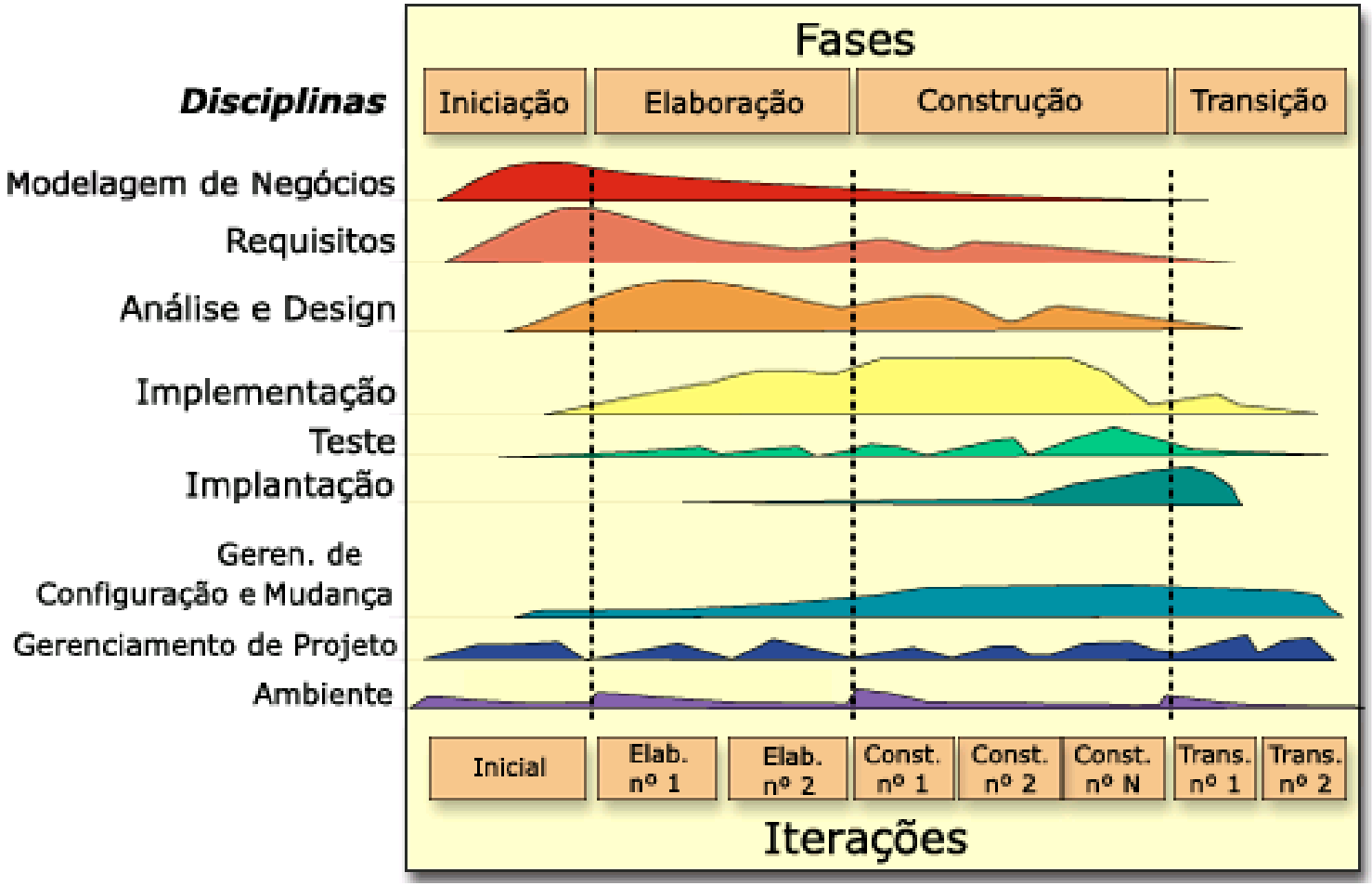
PRODUTO código fonte, código de máquina, subsistemas, classes, diagramas: interação, de estados e outros **artefatos**

ARTEFATO é qualquer tipo de informação criada por uma pessoa (diagramas UML, textos, modelos de interfaces)

PROCESSO define **quem** faz **o que, quando** e **como**

PU é um processo. Considera fatores *organizacionais*, do *domínio*, *ciclo de vida* e *técnicos*

2.3-Modelo do Processo Unificado



2.3.1 - Conceitos principais

- Componentes estáticos
  - Disciplinas
  - Fluxos
  - Papéis
  - Atividades
  - Artefatos
- Componentes dinâmicos
  - Ciclos
  - Fases
  - Iterações
  - Marcos

2.3.1 - Conceitos-chave: Disciplina

- Disciplina
  - **Uma disciplina é um conjunto de atividades relacionadas a uma "área de interesse" importante em todo o projeto.**
  - O fluxo de trabalho de uma disciplina é uma seqüência semi-ordenada das atividades que são realizadas para alcançar um determinado resultado.
- Tipos de Disciplina
  - Relacionadas ao desenvolvimento
    - Requisitos, análise e Design, implementação, implantação,...
  - Relacionadas ao suporte
    - Ger. de projeto, ger. de configuração e mudança, ambiente,...

2.3.1 - Conceitos-Chave: Fluxo de trabalho

- Fluxo de Trabalho
  - Uma simples enumeração de todos os papéis, atividades e artefatos não constitui um processo;
  - É necessária uma forma para descrever as seqüências significativas das atividades que produzem algum resultado importante e para mostrar as interações entre os papéis.
  - **O *fluxo de trabalho* é uma seqüência das atividades que produzem um resultado de valor observável.**

2.3.1 - Conceitos-Chave: Papel

- Define
  - Comportamento e as responsabilidades de um indivíduo ou de um conjunto de indivíduos que trabalham juntos como uma equipe
- Papéis não são indivíduos
  - Descrição do comportamento e das responsabilidades que eles devem ter.
  - Mapeamento Indivíduo/Papel é feito pelo Gerente do Projeto
- Exemplos
  - Analista de Sistemas, Designer de Negócio, Revisor de Requisitos, Implementador, Arquiteto de Software,...

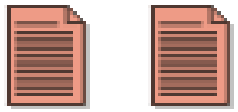
2.3.1 - Conceitos-Chave: Atividade

- **Uma atividade é uma unidade de trabalho que um indivíduo, desempenhando o papel descrito, pode ser chamado a realizar.**
- Os papéis possuem atividades que definem o trabalho que executam. Uma atividade é algo que um papel faz e produz um resultado significativo no contexto do projeto
- Exemplos
  - Avaliar viabilidade do conceito arquitetural
  - Estruturar modelo de implementação

2.3.1 - Conceitos-Chave: Artefato

- Artefato é um conjunto de informação que:
  1. É criado, alterado e usado pelos participantes nas suas atividades;
  2. Representa uma área de responsabilidade;
  3. Pode ser colocado sob controle de versão.
- Um artefato é um produto de trabalho do processo: os papéis usam os artefatos para executar atividades e produzem artefatos ao executarem as atividades
- As atividades possuem artefatos de entrada e saída

Analista de Sistema/ Especificador de Requisitos

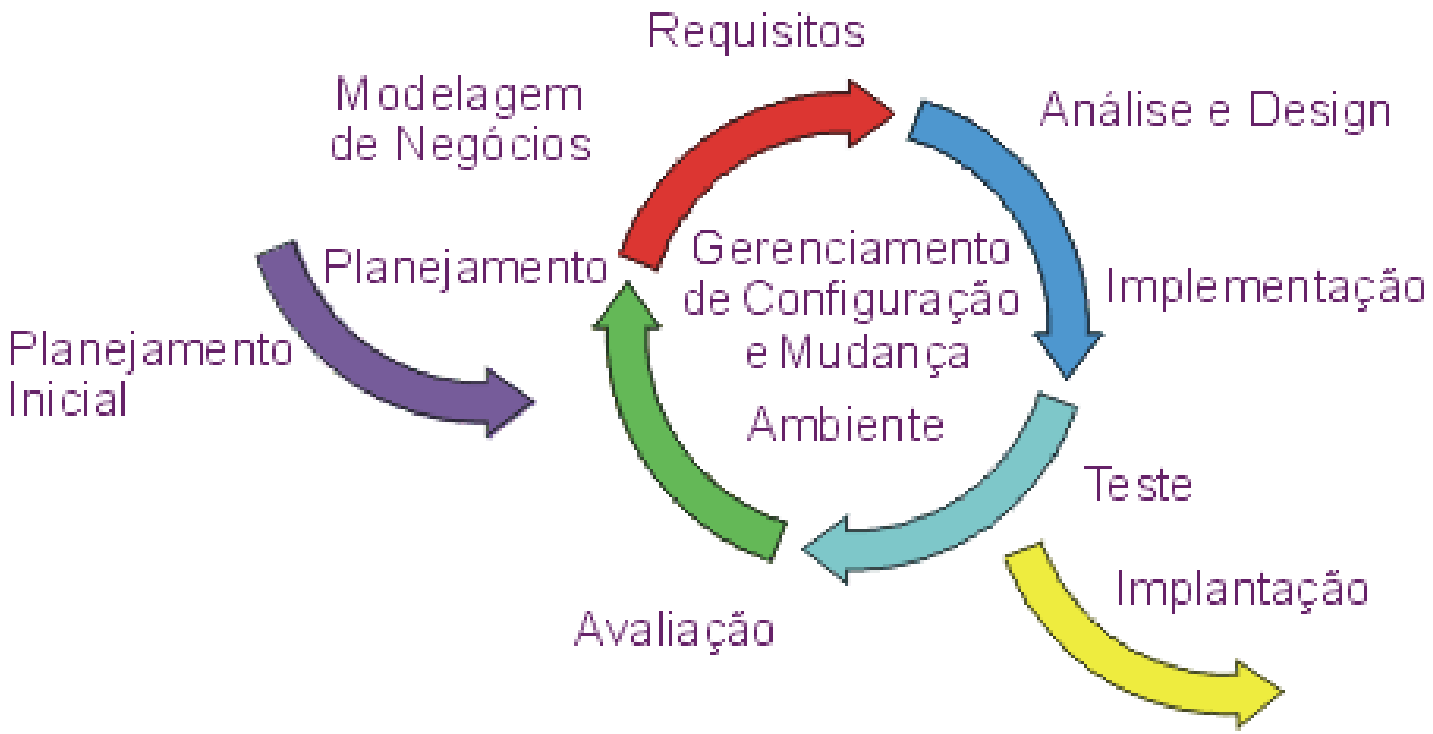


Domínio de Negócio e Especificação de Casos de uso

2.4 - Componentes dinâmicos

- UP é um processo baseado no modelo espiral
  - Projeto dividido em ciclos
  - Cada ciclo representa uma nova versão do produto
  - Cada ciclo possui 4 fases consecutivas
- Fases
  - **Iniciação – definição do escopo**
  - **Elaboração – análise e projeto**
  - **Construção – desenvolvimento e integração**
  - **Transição – passagem ao "domínio público"**
- O final das fases define os principais marcos do projeto
- Cada fase pode ser quebrada em iterações
  - Cada iteração resulta em uma nova release

2.4 – Componentes Dinâmicos - Espiral



2.4 – Componentes Dinâmicos

Cada **fase** termina com um **MARCO (milestone):**

Um **CICLO** é uma sequência das 4 fases.
Um projeto pode necessitar de vários ciclos.

INICIO: o que o sistema fará? Como poderia ser a arquitetura?
Prazos e custos?

- identificar os **riscos**
- fixar subconjunto chave -> arquitetura candidata
- estimativas iniciais (custos, esforços, alocações e qualidade do produto)
- iniciar caso gerencial (*business case*)

2.4.1 - Fases do Processo Unificado(1)

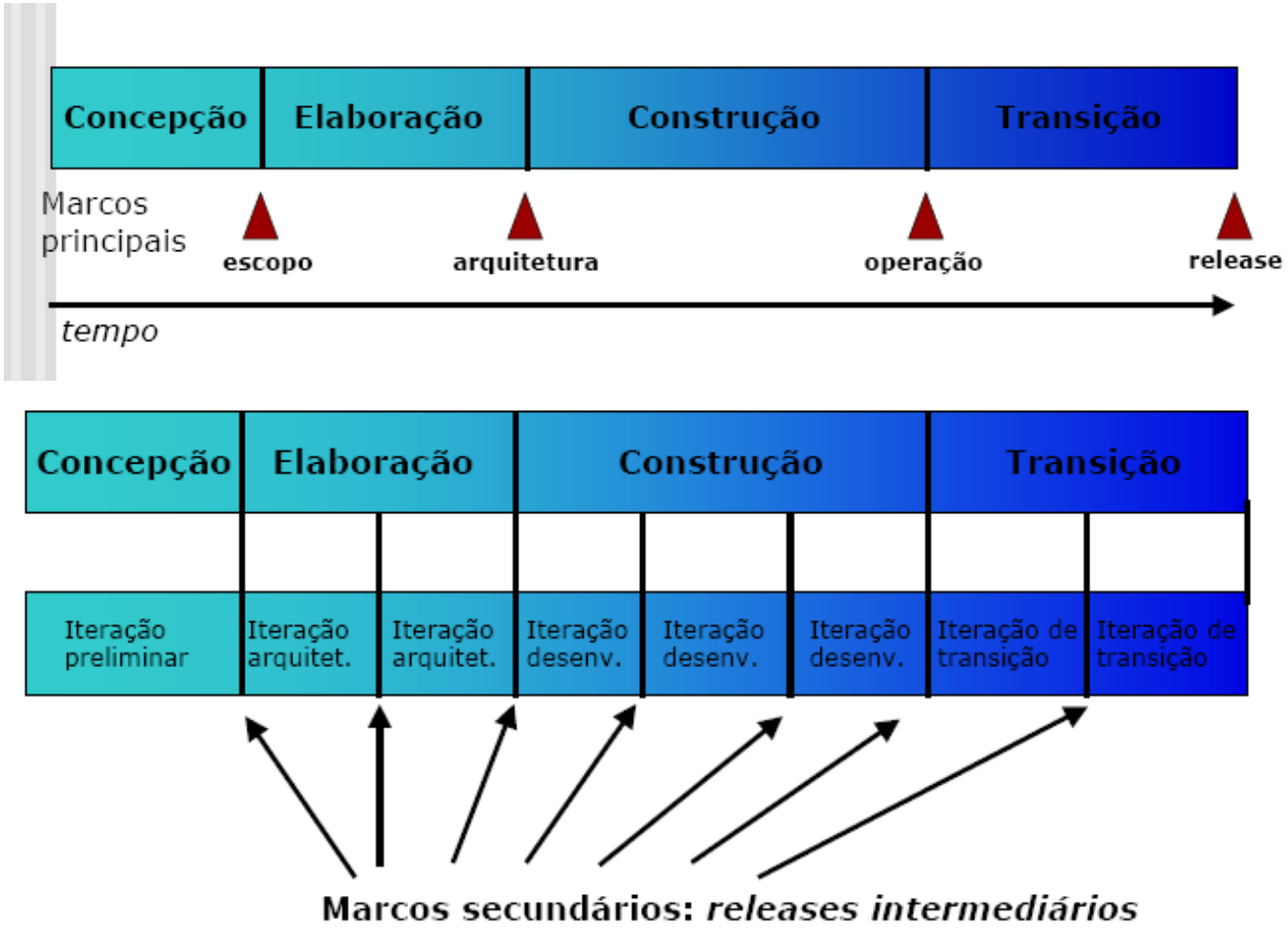
2.4 - Fases e Fluxos de Trabalho

- Fases:
  - Concepção(iniciação)
  - Elaboração
  - Construção
  - Transição
- Fluxos de Trabalho
  - Técnico
    - Requisitos
    - Análise
    - Projeto
    - Implementação
    - Teste
  - de Gerência
  - de Ambiente
  - de Avaliação
  - de Implantação

2.4 - Fases do Processo Unificado(2)

- Fase de Concepção: estabelece a viabilidade do sistema, através dos requisitos mais relevantes.
- Fase de Elaboração: define uma arquitetura estável e estima o custo da fase de construção, com base no refinamento de requisitos.
- Fase de Construção: produz um produto de software pronto como versão inicial operacional.
- Fase de Transição: transfere o sistema para o ambiente do usuário.

2.4.1 – Fases e Iterações



2.5 - Fluxos de Trabalho (FT)

- FT Técnicos: compreendem as atividades normalmente consideradas no processo de software: requisitos, análise, projeto, implementação e testes.
- FT de Gerência: compreende as atividades de planejamento e acompanhamento de projeto.

2.5 - Fluxos de Trabalho (FT) 2

- FT de Ambiente: compreende as atividades de automação do processo e desenvolvimento do ambiente de manutenção.
- FT de Avaliação: compreende as atividades de avaliação das tendências do processo e da qualidade do produto.
- FT de Implantação: compreende as atividades para transferir o produto final para o usuário.

2.6 - Modelos Gerados

- Requisitos: modelo de casos de uso
- Análise: modelo de análise
- Projeto:
  - modelo de projeto
  - modelo de implantação
- Implementação: modelo de implementação
- Teste: modelo de teste

3 - Fases do Ciclo de Vida

- Mudança em relação ao ciclo de vida tradicional: necessidade de diferenciar as atividades de pesquisa e desenvolvimento das atividades de produção:
  - pesquisa e desenvolvimento: atividades relacionadas com a funcionalidade
  - produção: atividades relacionadas com o produto a ser entregue ao cliente (robustez, desempenho, adequação e finalização)

3 - Fases do Ciclo de Vida (2)

- Um processo moderno de desenvolvimento de software deve ser definido para prever:
  - evolução de planos, requisitos e arquitetura
  - gerência de riscos e medidas objetivas do progresso e da qualidade
  - evolução da capacidade do sistema, através da demonstração do acréscimo de funcionalidade

3.1 - Estágios do Ciclo de Vida

- estágio de engenharia:
  - atividades menos previsíveis
  - equipe menor com atividades de projeto e síntese
- estágio de produção:
  - atividades mais previsíveis
  - equipe maior com atividades de construção, teste e implantação

3.1 - Estágios do Ciclo de Vida (2)

- estágio de engenharia:
  - fase de concepção
  - fase de elaboração
- estágio de produção:
  - fase de construção
  - fase de transição

3.1.1 - Fase de Concepção

- Definir o escopo de software e as condições de contorno (conceito operacional, critérios de aceitação, entendimento claro do produto).
- Identificar os casos de uso críticos.
- Sintetizar pelo menos uma arquitetura candidata.
- Estimar custo e cronograma do projeto inteiro.
- Estimar riscos potenciais.

3.1.2 - Fase de Elaboração

- Obter arquitetura, requisitos e planos confiáveis, de forma mais estável possível:
  - riscos minimizados
  - previsão precisa de custo e cronograma
- construir um protótipo executável da arquitetura em uma ou mais interações:
  - protótipo evolucionário
  - protótipo descartável

3.1.3 - Fase de Construção

- Completar o sistema com as características da aplicação.
- Fazer testes sistemáticos, segundo critérios estabelecidos.
- Gerenciar recursos e controlar operações para otimizar custos, cronogramas e qualidade.
- Avaliar os produtos contra os critérios de aceitação.

3.1.4 - Fase de Transição

- Validar o sitema contra as expectativas do cliente.
- Validar o sistema em relação ao sistema legado (operação paralela).
- Converter as bases de dados operacionais.
- Treinar os usuários e o pessoal de suporte.

4.1 - Conjuntos de Artefatos Relevantes

- Gerência
- Engenharia
  - Requisitos
  - Projeto
  - Implementação
  - Implantação
- Avaliação

4.1.1 - Artefatos de Gerência

- Artefatos de Planejamento: plano de desenvolvimento de software e outros documentos que dão suporte para a sua elaboração
- Artefatos Operacionais: documentos relacionados com a execução do plano

4.1.2- Artefatos de Engenharia

- Requisitos:
  - documento da visão (visão geral do sistema a ser desenvolvido)
  - modelos de requisito
- Projeto:
  - modelos de projeto
  - modelo de teste
  - descrição da arquitetura de software

4.1.2 - Artefatos de Engenharia (2)

- Implementação
  - *baseline* de códigos fontes
  - arquivos compilados correspondentes
  - códigos executáveis dos componentes
- Implantação
  - *baseline* do produto integrado
  - código executável correspondente
  - manual do usuário

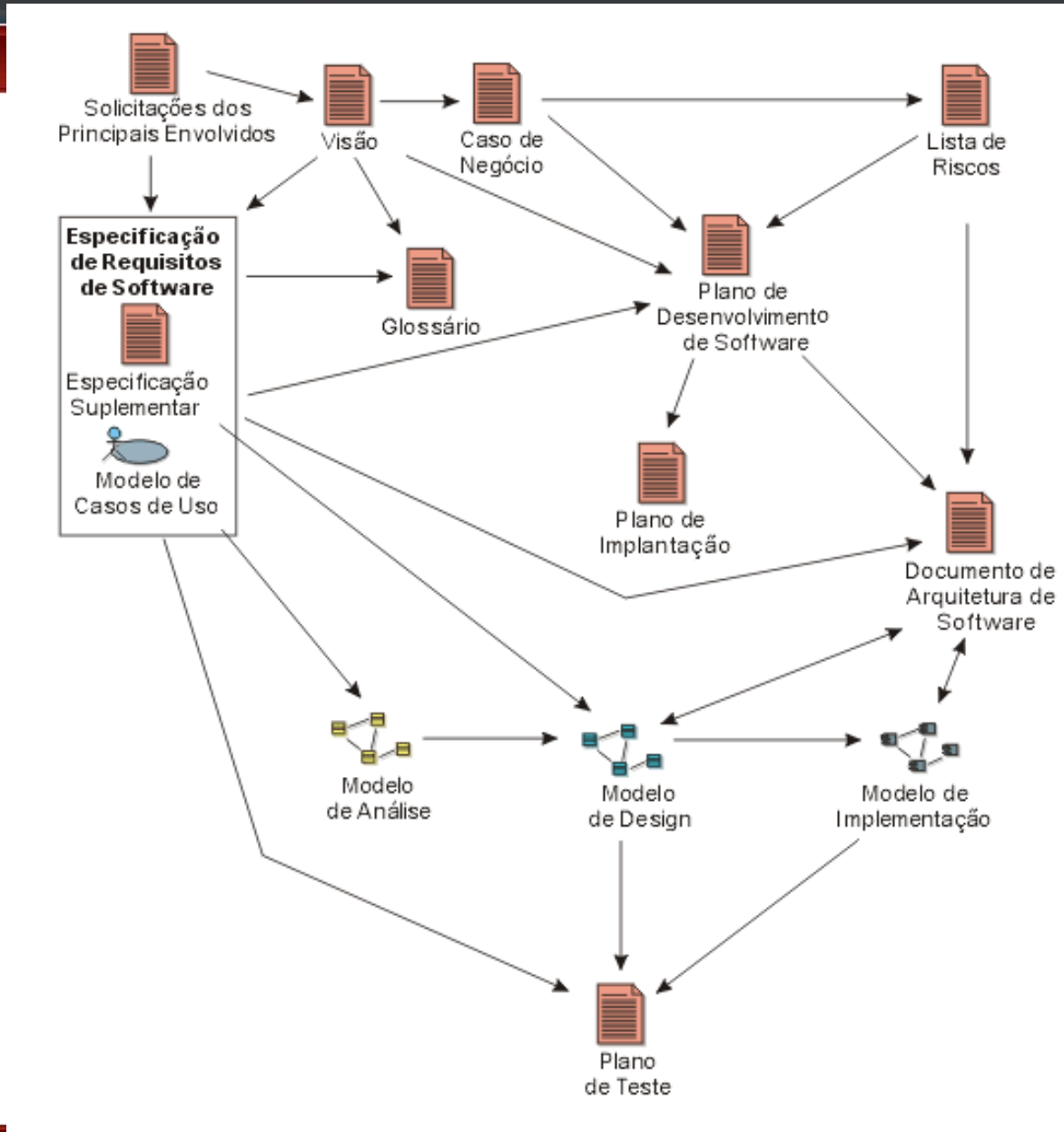
4.1.3 - Artefatos de Avaliação

- O FT de avaliação está relacionado com as atividades de inspeção, análise, demonstração e teste.
- Os artefatos de teste dependem do tipo de sistema.
- Exemplos: plano de teste, descrição de testes de aceitação, projeto de componentes auxiliares para a execução de testes, etc.

- Visão

Geral

Dos Artefatos



5 - Descrição de Fluxos de Trabalho

- Introdução (visão geral do FT)
- Papel do FT no ciclo de vida de software
- Artefatos produzidos
- Participantes
- Atividades

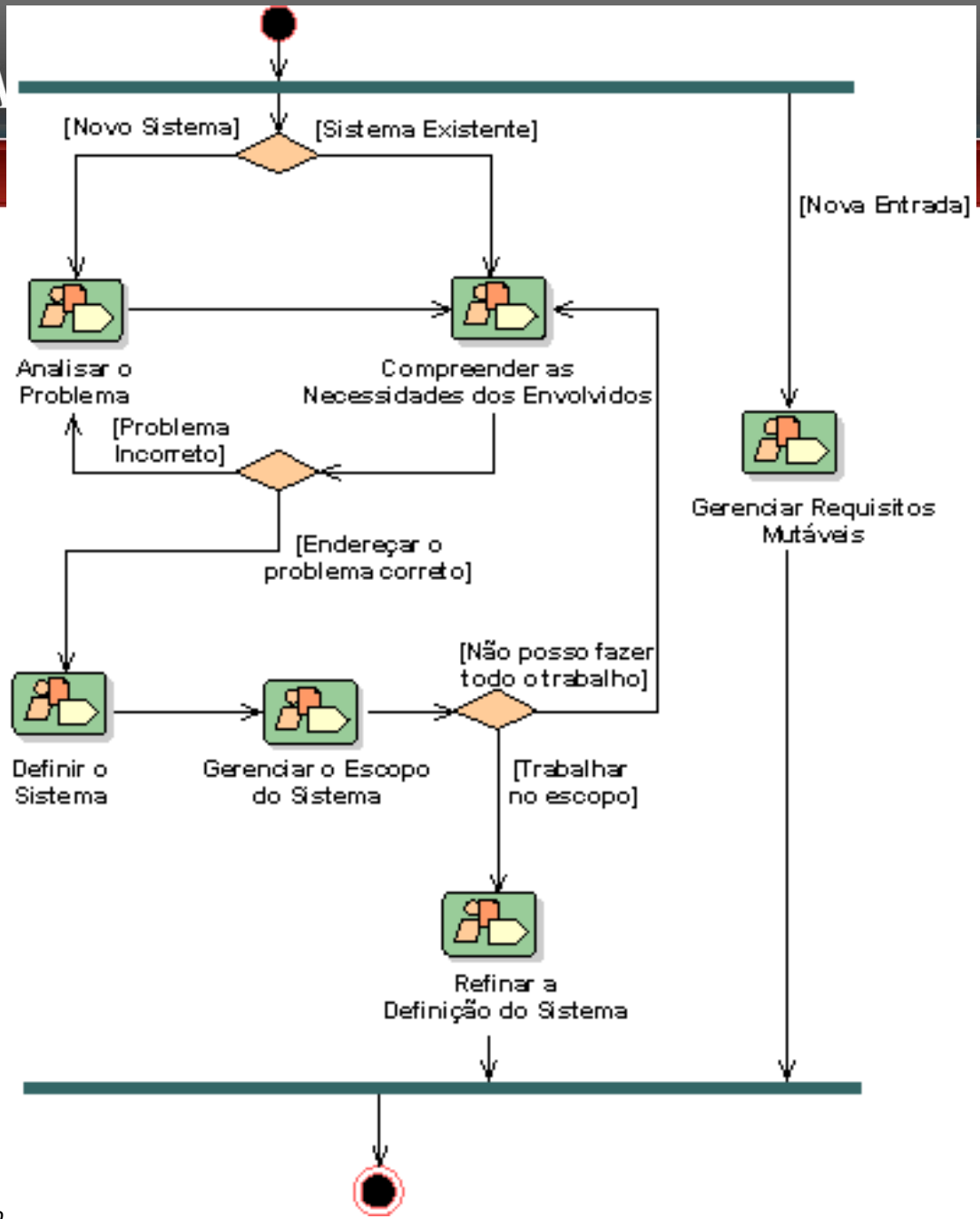
5.1 - FT - Requisitos

- Objetivo principal:direcionar o desenvolvimento para o sistema correto.
- Atividade principal: captura de requisitos
- O cliente deve ser capaz de entender os requisitos capturados.
- O resultado é usado para planejar as iterações e as versões a serem entregues ao cliente.

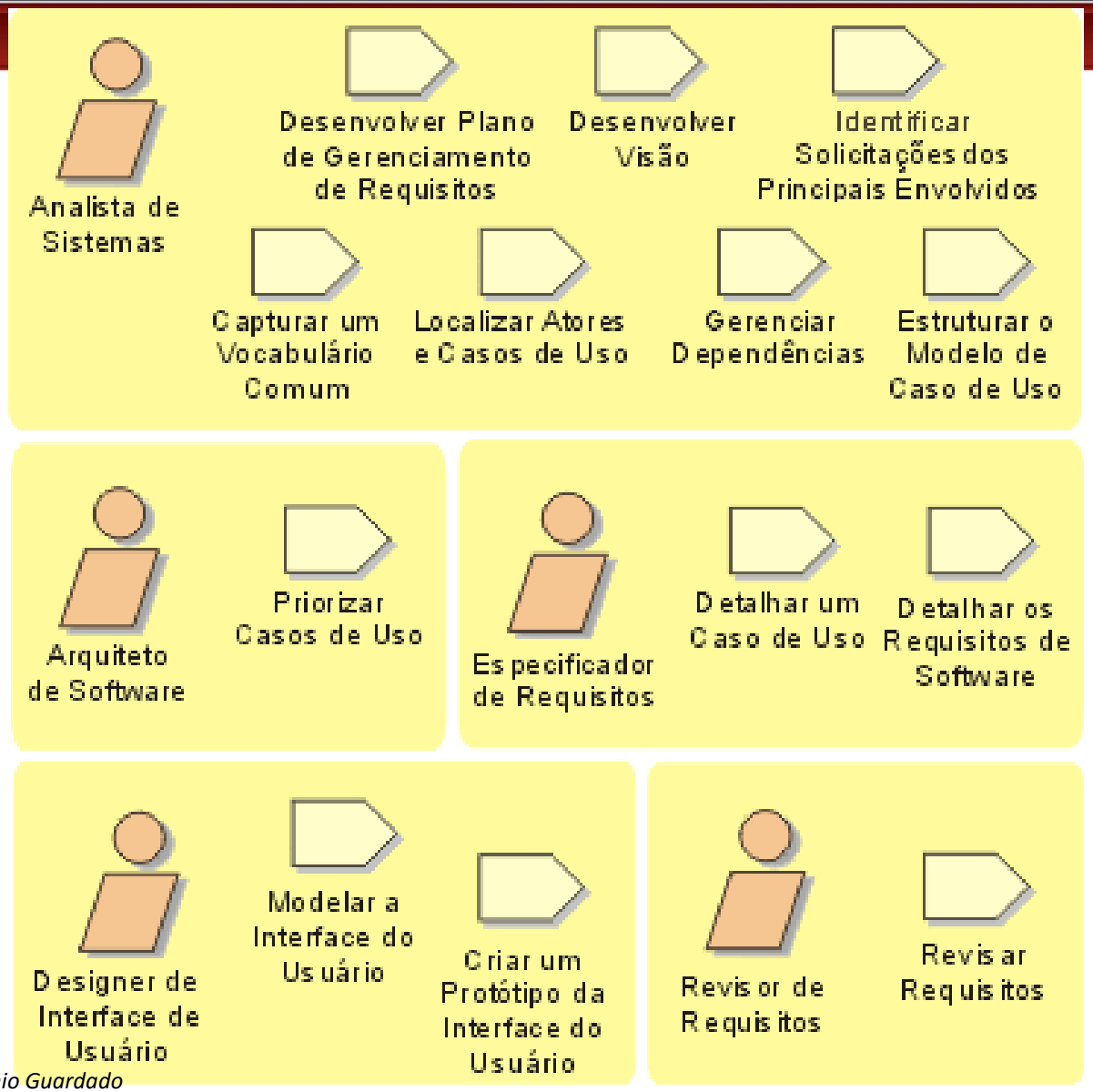
5.1 - Requisitos - Trabalhadores

- Analista de sistema
  - Definição do conjunto de requisitos (funcionais e não funcionais)
- Especificador de casos de uso
  - Especificação de um ou mais casos de uso
- Projetista de Interface do usuário
  - Especificação da interface do usuário
- Arquiteto de Software
  - Responsável pela arquitetura do modelo de casos de uso

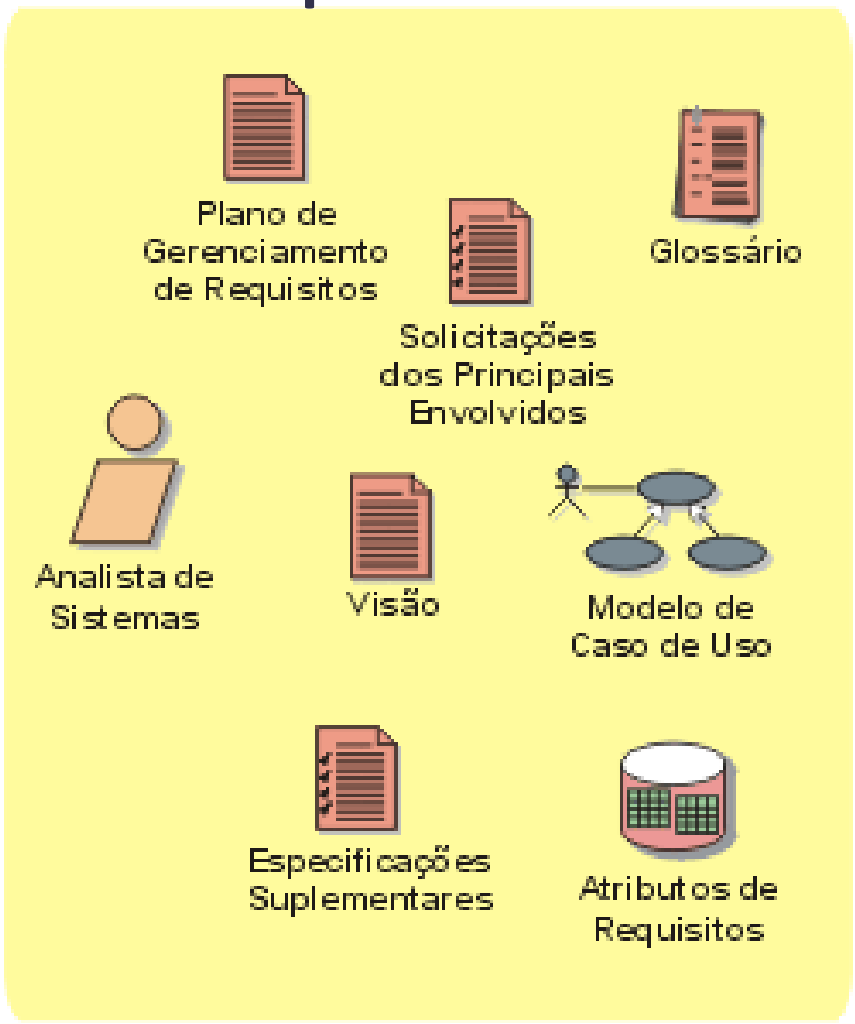
5.1.1- Requisitos:Fluxo de Trabalho

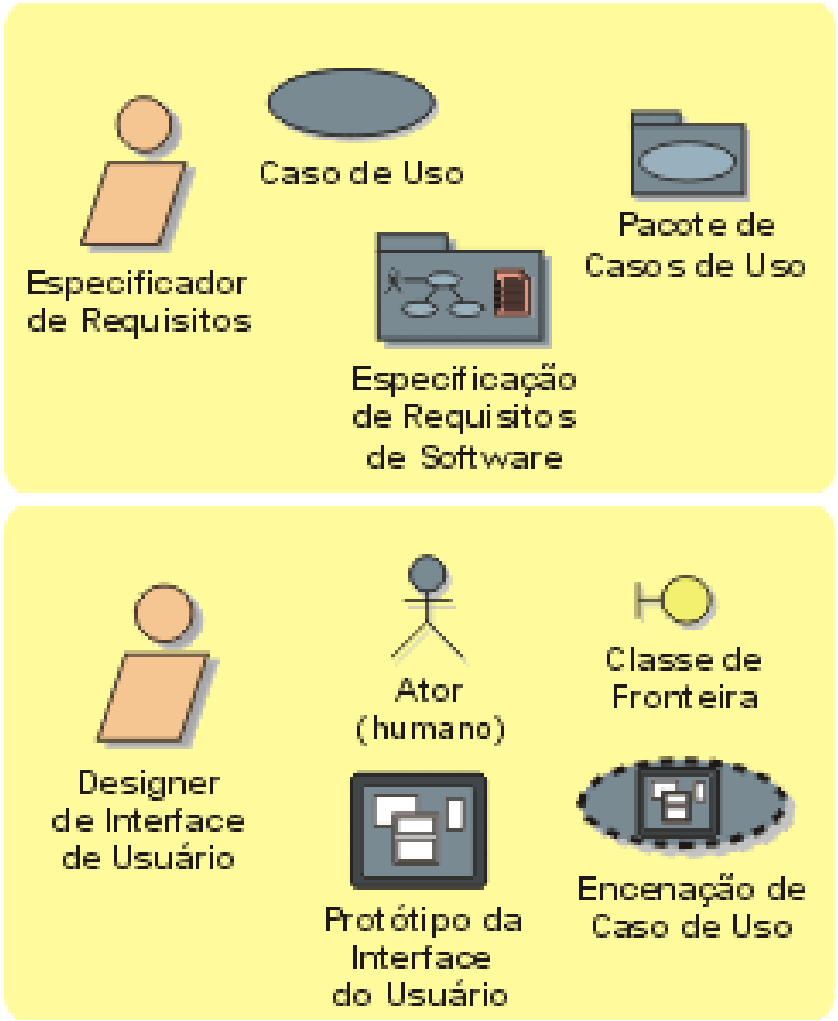


5.1.2 – Requisitos : Atividades x Trabalhadores



5.1.3-Requisitos:Artefatos





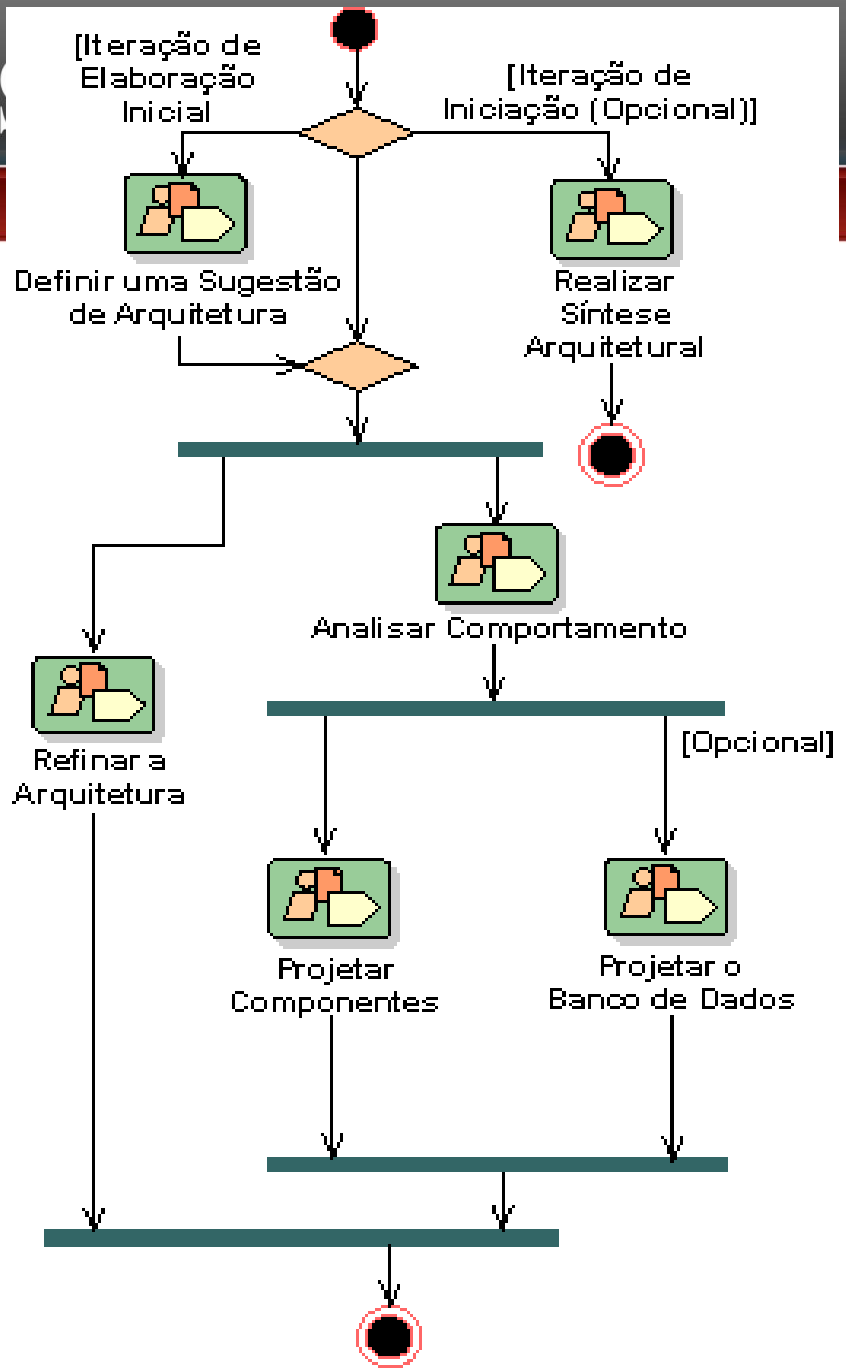
5.2 - FT - Análise

- Objetivo: estruturar e refinar os requisitos capturados, para obter um entendimento mais preciso dos requisitos.
- A estruturação dos requisitos tem a finalidade de estruturar o sistema como um todo e obter uma descrição de requisitos que facilite a sua manutenção.

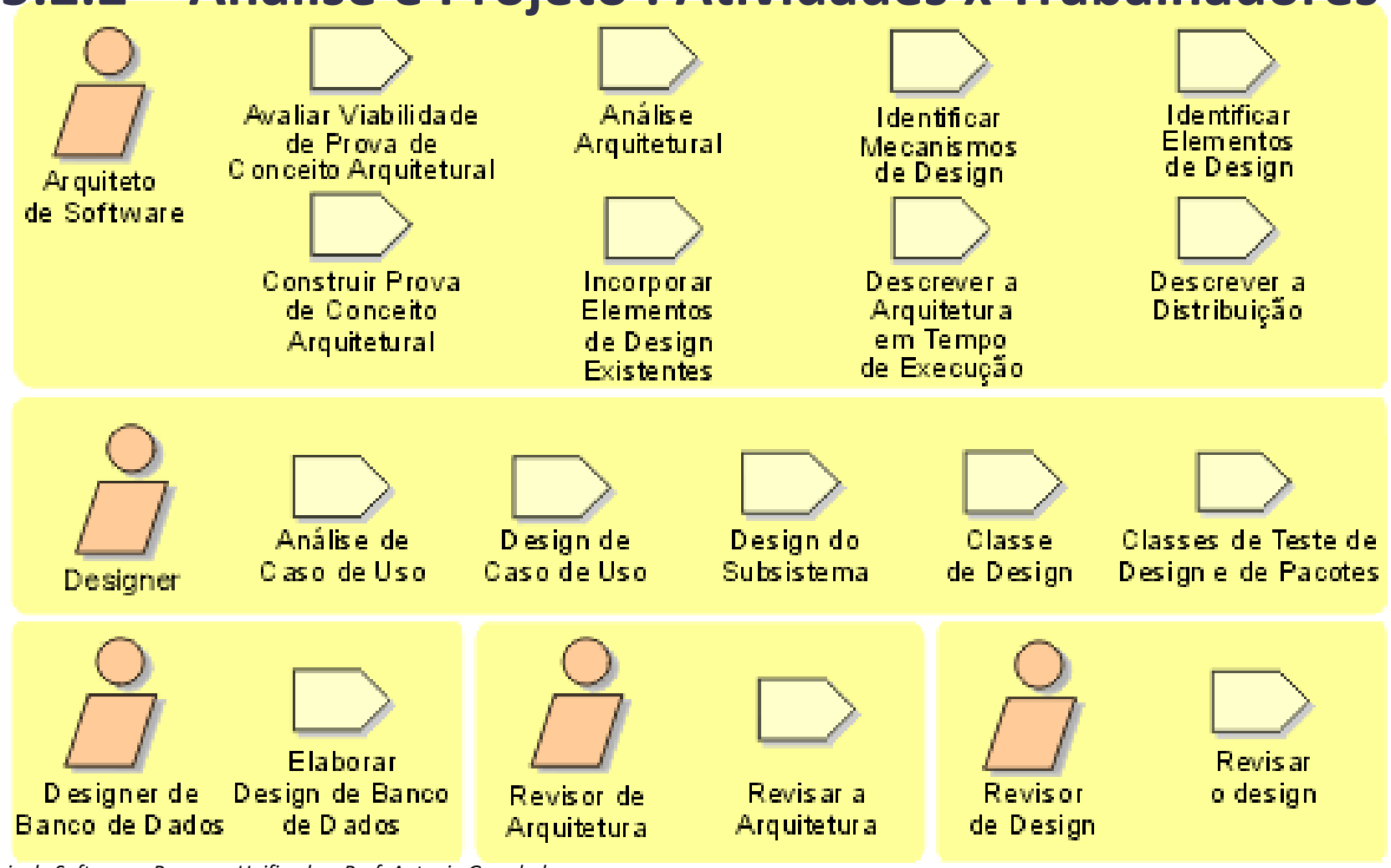
5.2 - Análise - Trabalhadores

- Arquiteto
  - Responsável pelo modelo de análise
- Engenheiro de caso de uso
  - Realização de um ou mais casos de uso da análise
- Engenheiro de componente
  - Especificação dos pacotes e classes

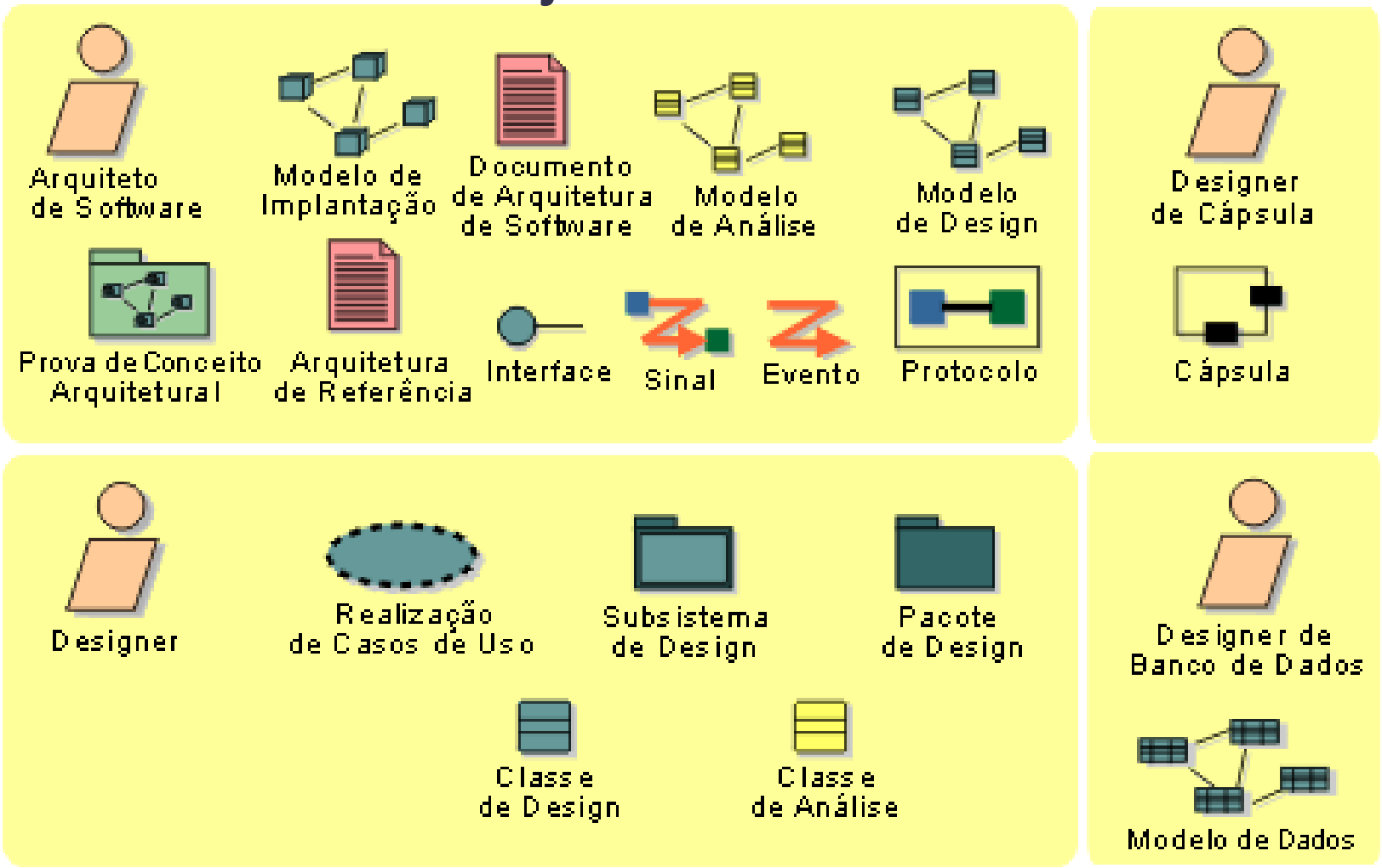
5.2.1- FT Análise e Projeto



5.2.2 – Análise e Projeto : Atividades x Trabalhadores



5.2.3 – Análise e Projeto : Artefatos



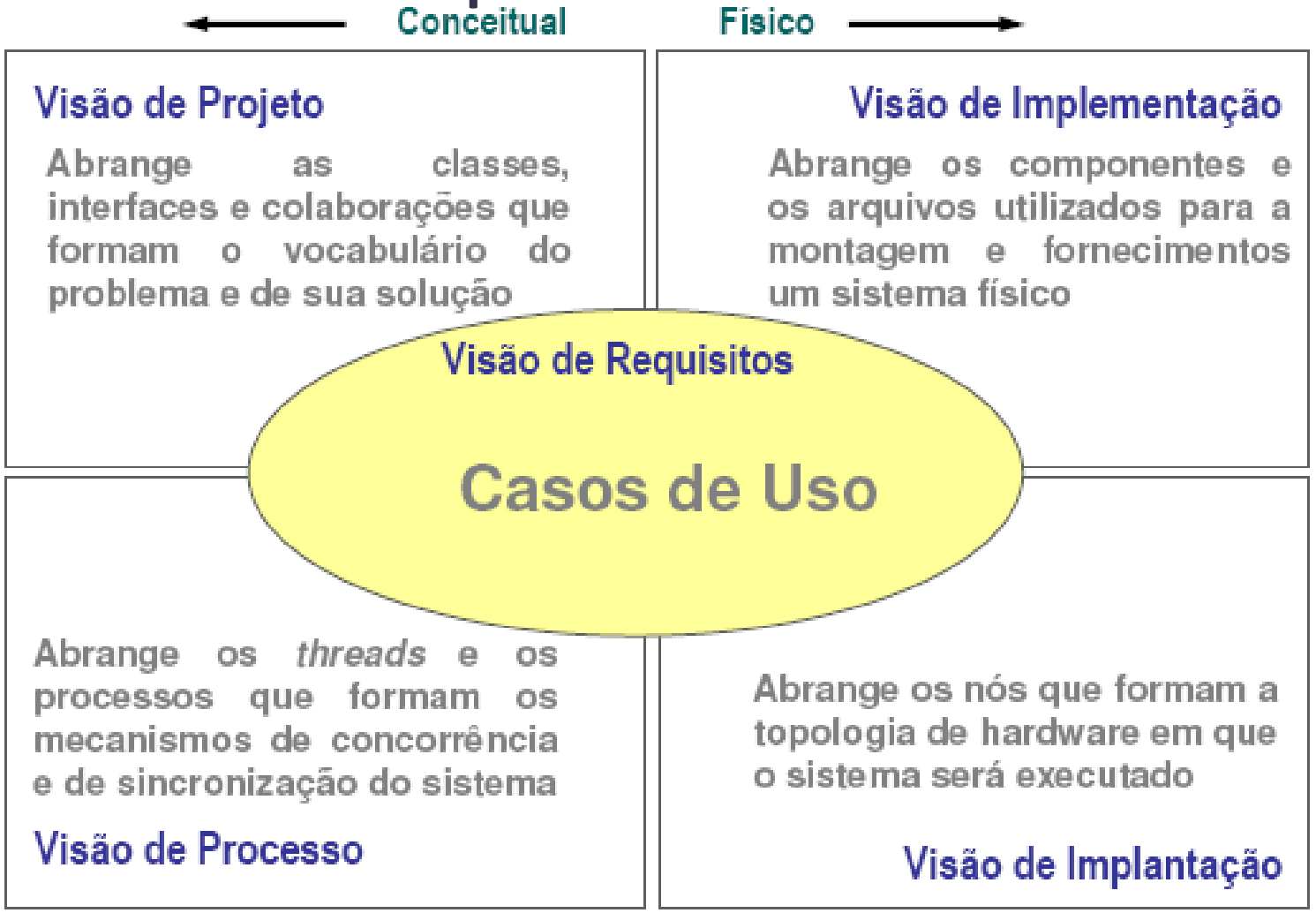
5.3 - Arquitetura de Software

- Arquitetura de software abrange as decisões significativas sobre:
  - organização do sistema de software
  - elementos estruturais, que compõem o sistema, e as interfaces entre eles, juntamente com a interação entre estes elementos
  - composição dos elementos em subsistemas progressivamente maiores
  - estilo da arquitetura

5.3.1 - Papel da Arquitetura

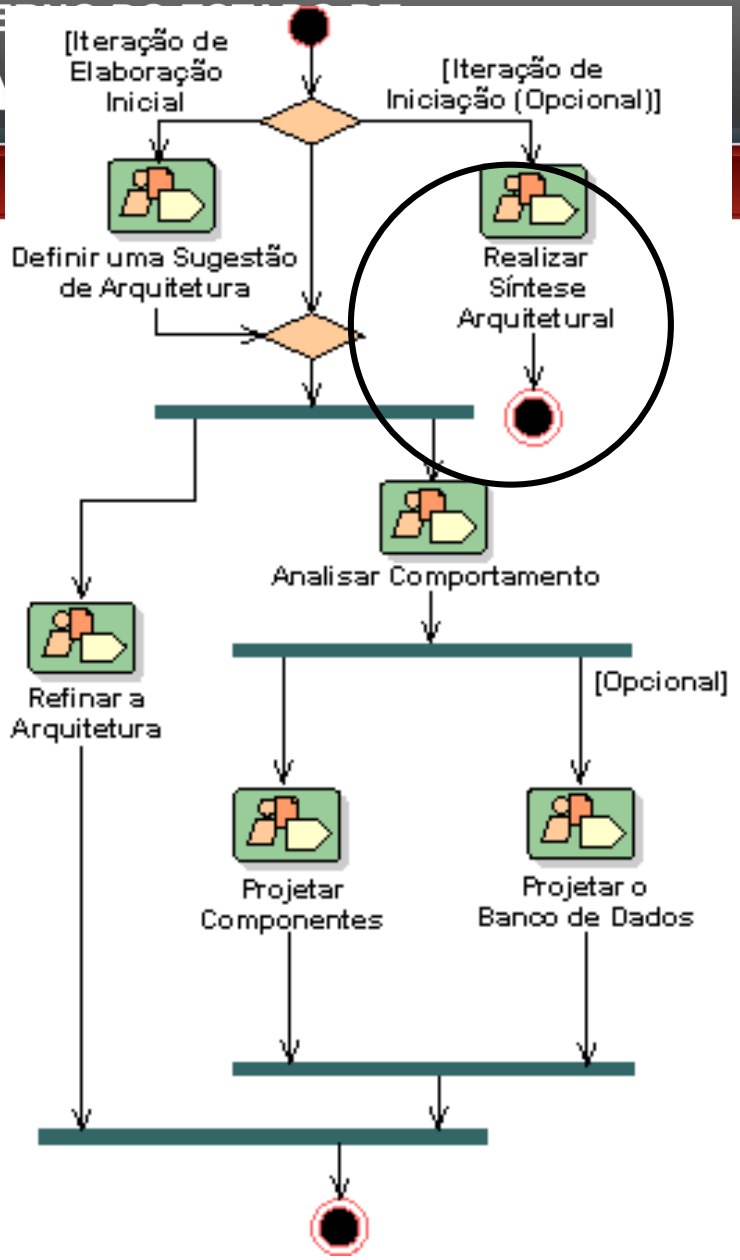
- no desenvolvimento: estabelece as informações da descrição do sistema em cada fase do desenvolvimento
- na gerência: estabelece os pontos de referência para acompanhar o andamento do projeto

5.3.2- Visões da Arquitetura



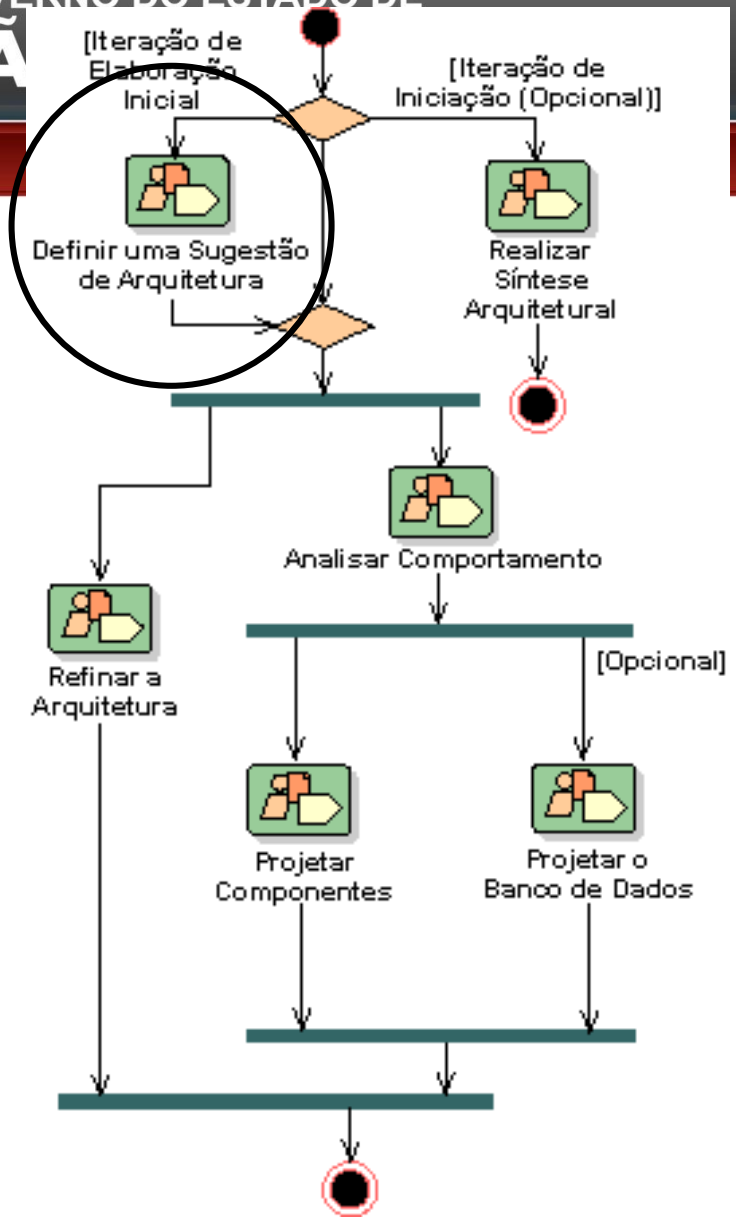
5.3.3-Realizar Síntese da Arquitetura

- Construir e avaliar uma prova de conceito arquitetural
  - Mostrar que existe uma solução possível de satisfazer os requisitos do sistema relevantes à arquitetura



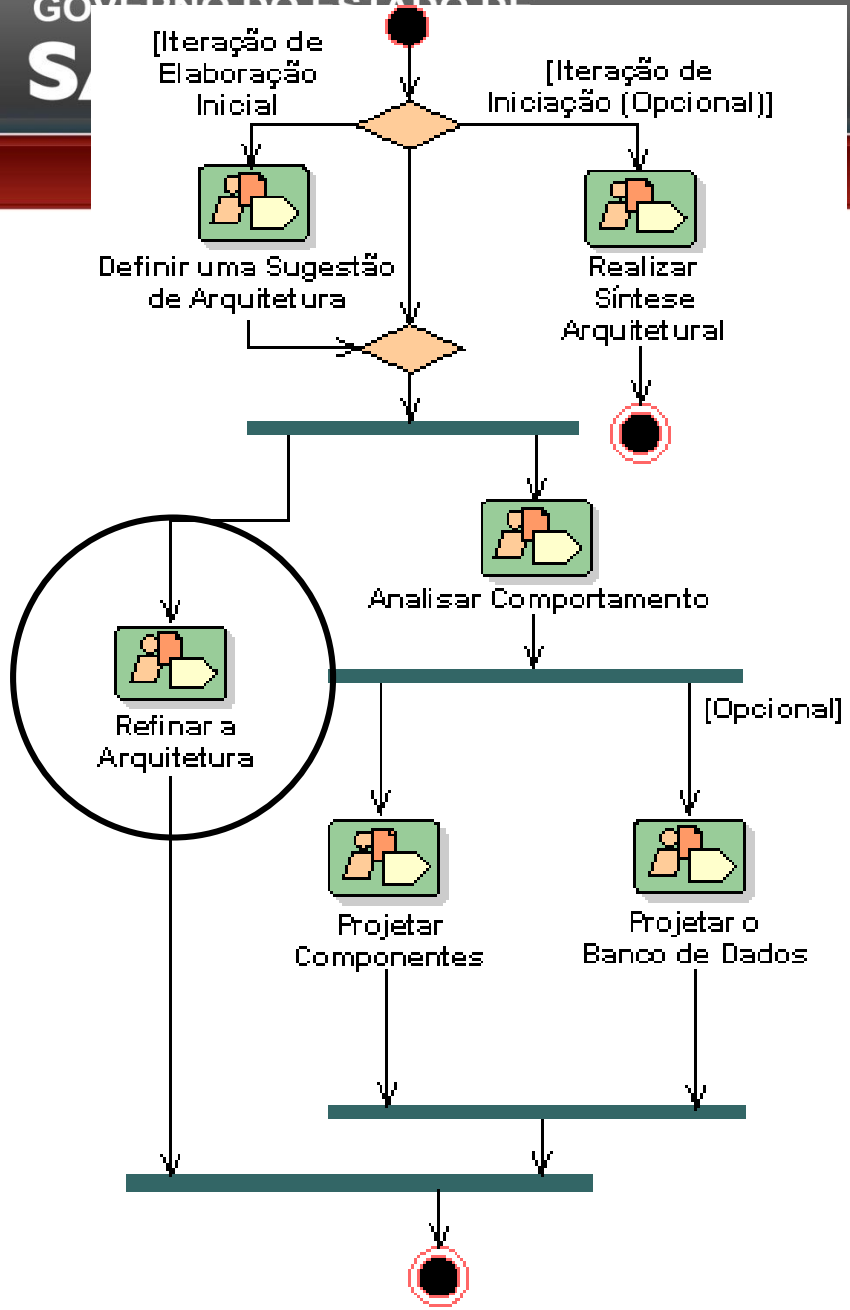
5.3.4-Definir arquitetura candidata

- Criar o esqueleto inicial da arquitetura do sistema
- Identificar classes de análise dos casos de uso arquiteturalmente relevantes
- Atualizar a realização de caso de uso com as interações entre classes de análise



5.3.5-Refinar a arquitetura

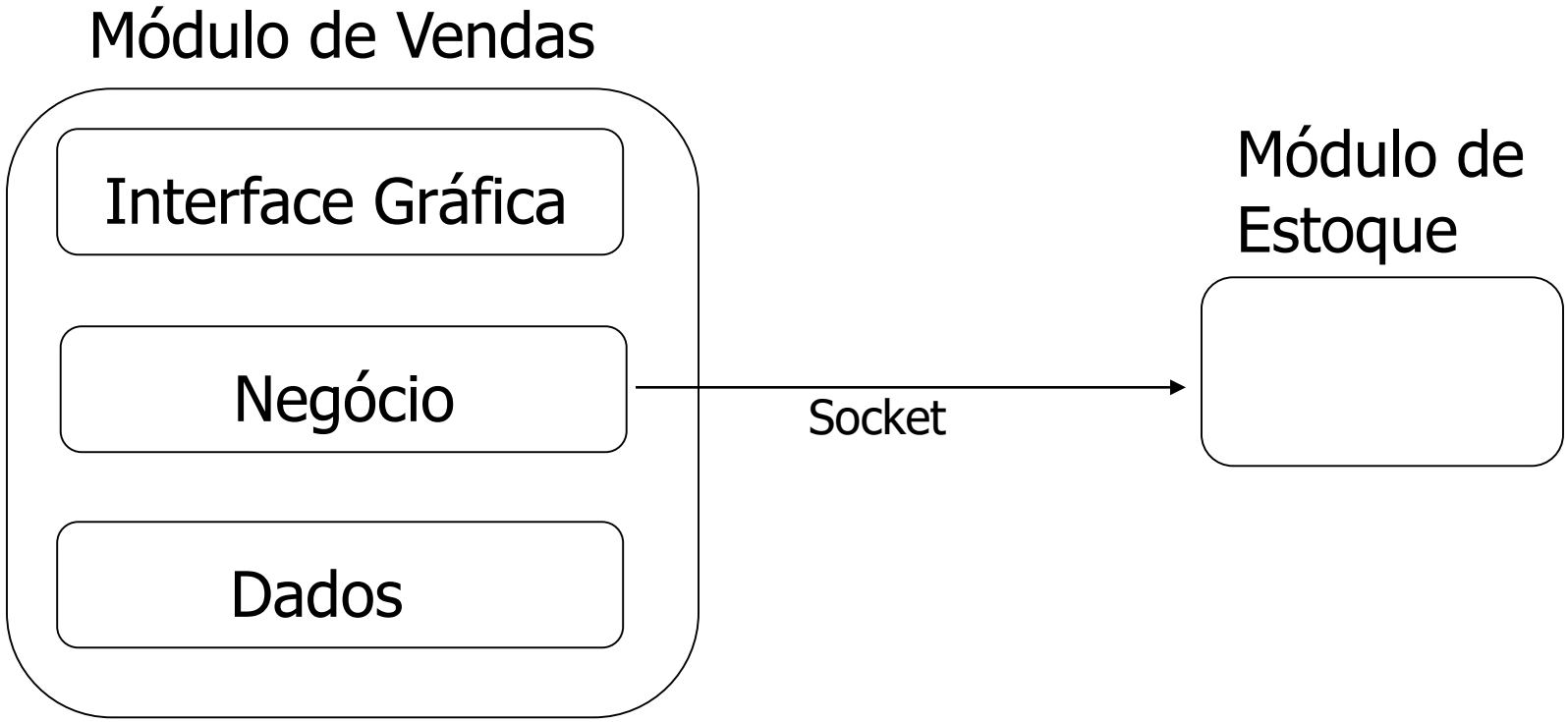
- Permitir uma transição entre os elementos e mecanismos de análise para os de projeto
- Manter a consistência e integração da arquitetura
- Descrever a arquitetura de execução e produção da aplicação



5.3.6-Arquitetura Inicial

- Quais as principais partes do sistema?
- Como elas interagem entre si?
- Que padrões arquiteturais utilizar (no todo ou internamente nas partes) ?
  - MVC
  - Baseado em camadas
  - Canais e filtros
  - Centrado em repositório

5.3.6-Arquitetura Inicial -Exemplo



5.3 - FT - Projeto

Objetivos:

- Incorporar as informações dos requisitos não funcionais e restrições relacionadas com linguagens de programação, reuso de componentes, sistemas operacionais, tecnologias de distribuição, concorrência, interface com usuário, gerência de transação, etc.

5.3 - FT - Projeto

- Planejar as atividades de implementação, decompondo o trabalho de implementação em partes a serem manuseados por diferentes equipes.
- Capturar as principais interfaces entre subsistemas o mais cedo possível.
- Definir uma notação comum para visualização e compreensão do projeto.
- Criar uma estrutura que não seja afetada pela mudança de tecnologia.

5.3 - Projeto - Trabalhadores

- Arquiteto:
  - Responsável pelos modelos de projeto e de implantação
- Engenheiro de caso de uso
  - Realização de um ou mais casos de uso do projeto
- Engenheiro de componente
  - Especificação dos subsistemas e das classes de projeto

5.4 - FT - Implementação

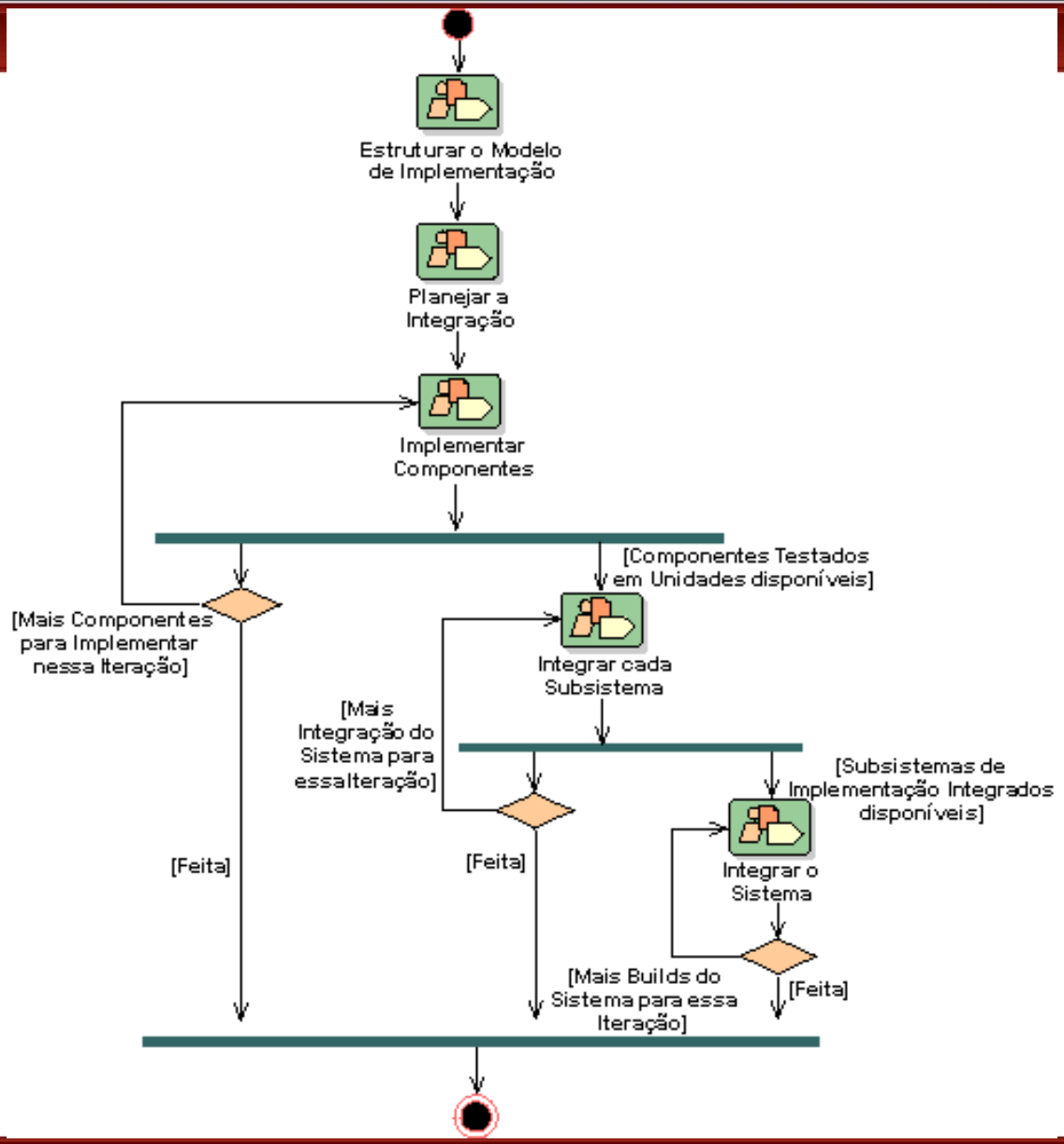
Objetivos:

- Planejar a integração do sistema necessária em cada iteração.
- Distribuir o sistema, através do mapeamento dos componentes executáveis nos nós do Modelo de Implantação.

5.4 - FT - Implementação

- Implementar as classes e os subsistemas definidos durante o projeto.
- Testar individualmente os componentes e integrá-los, através da compilação e ligação, antes de passá-los para a integração e teste do sistema.

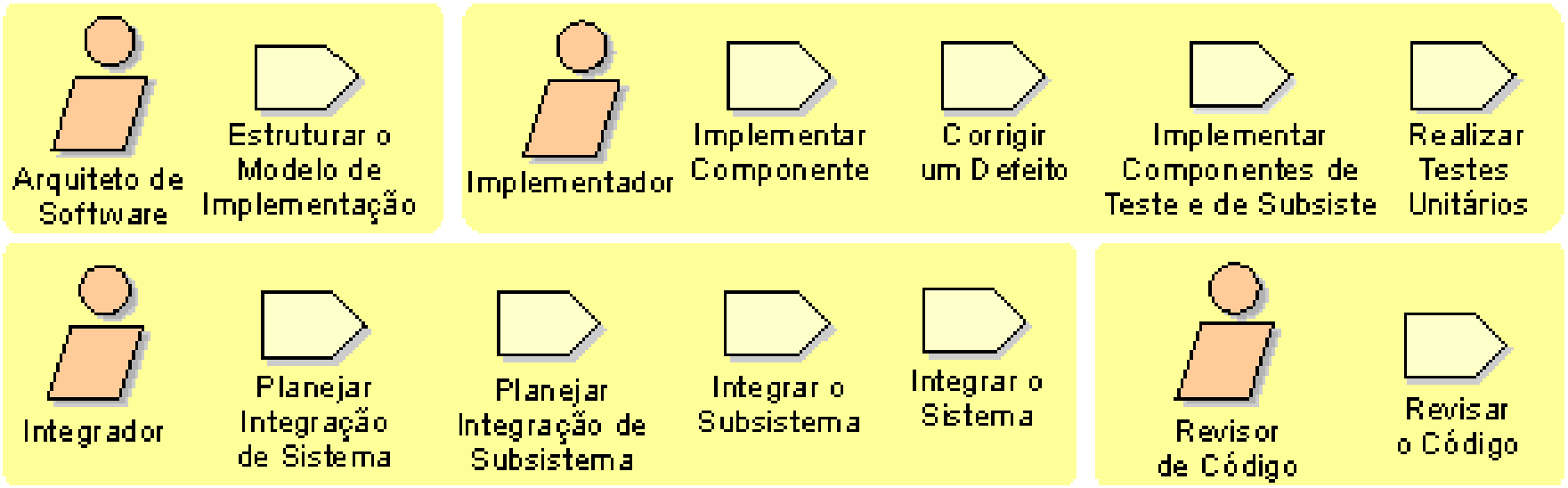
5.4 – FT – Implementação



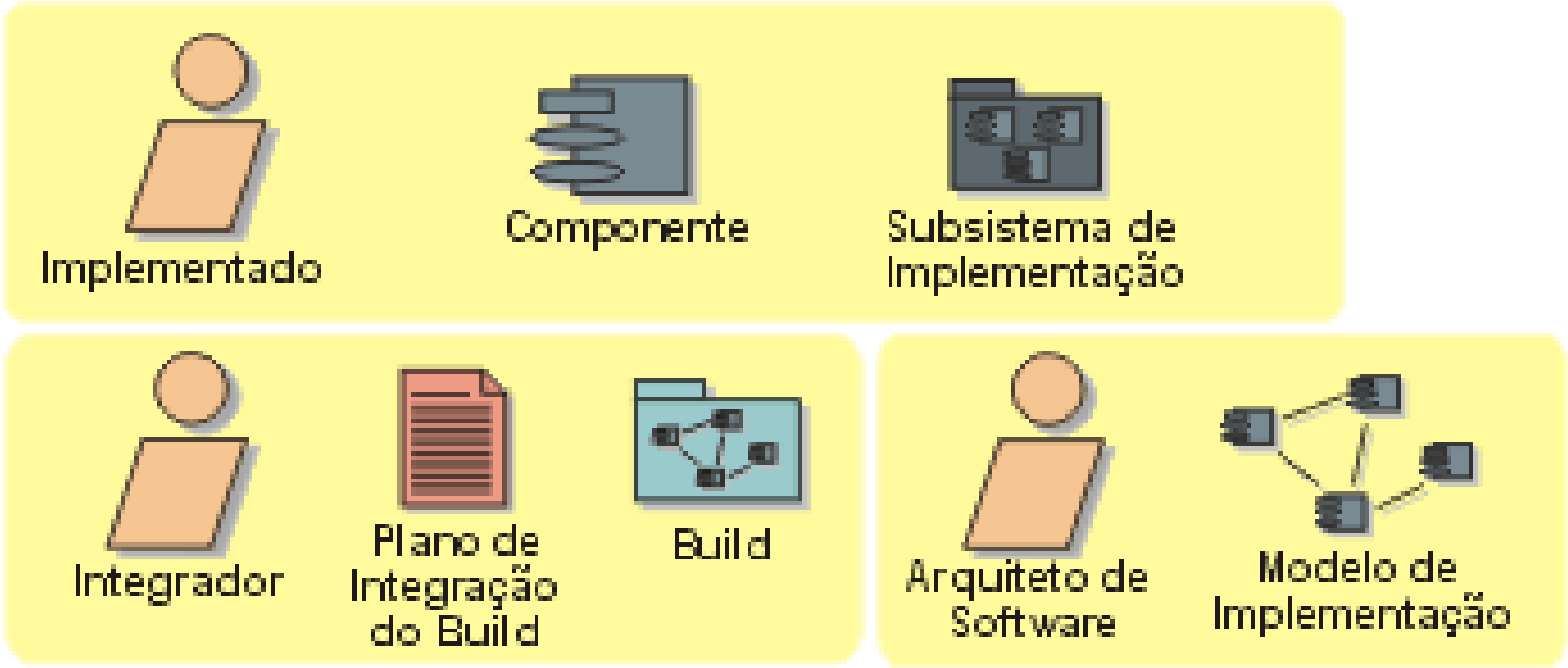
5.4 - Implementação - Trabalhadores

- Arquiteto
  - Responsável pelo modelo de implantação
- Engenheiro de componente
  - Responsável pela implementação de subsistemas e classes de projeto
- Integrador de sistema:
  - Planejamento das seqüências das versões executáveis

5.4 - Implementação – Atividades x Trabalhadores



5.4 - Implementação – Artefatos x Trabalhadores



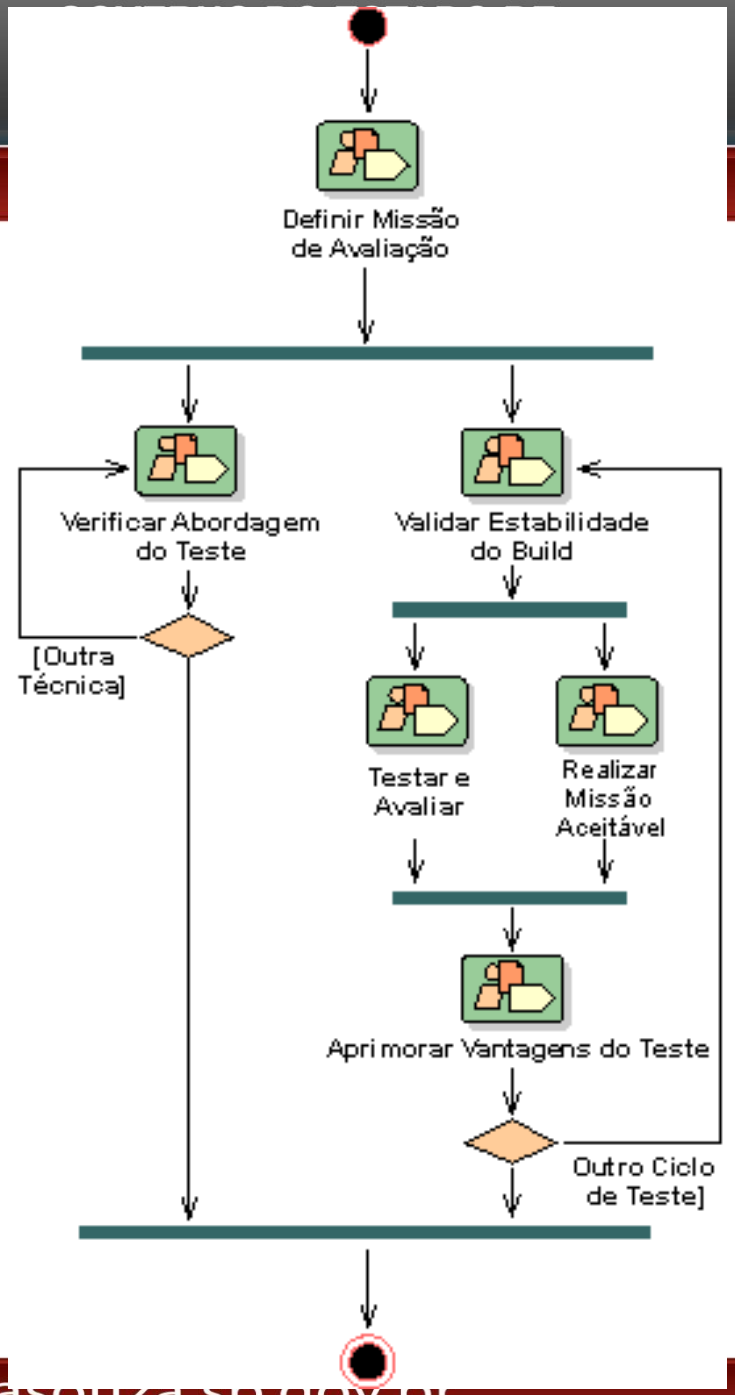
5.5 - FT - Teste

Objetivos:

- Planejar os testes de integração e os testes do sistema.
- Projetar e implementar os testes, através da especificação de:
  - casos de teste
  - procedimento de teste
  - componentes executáveis para suporte de testes.

5.5 – FT teste

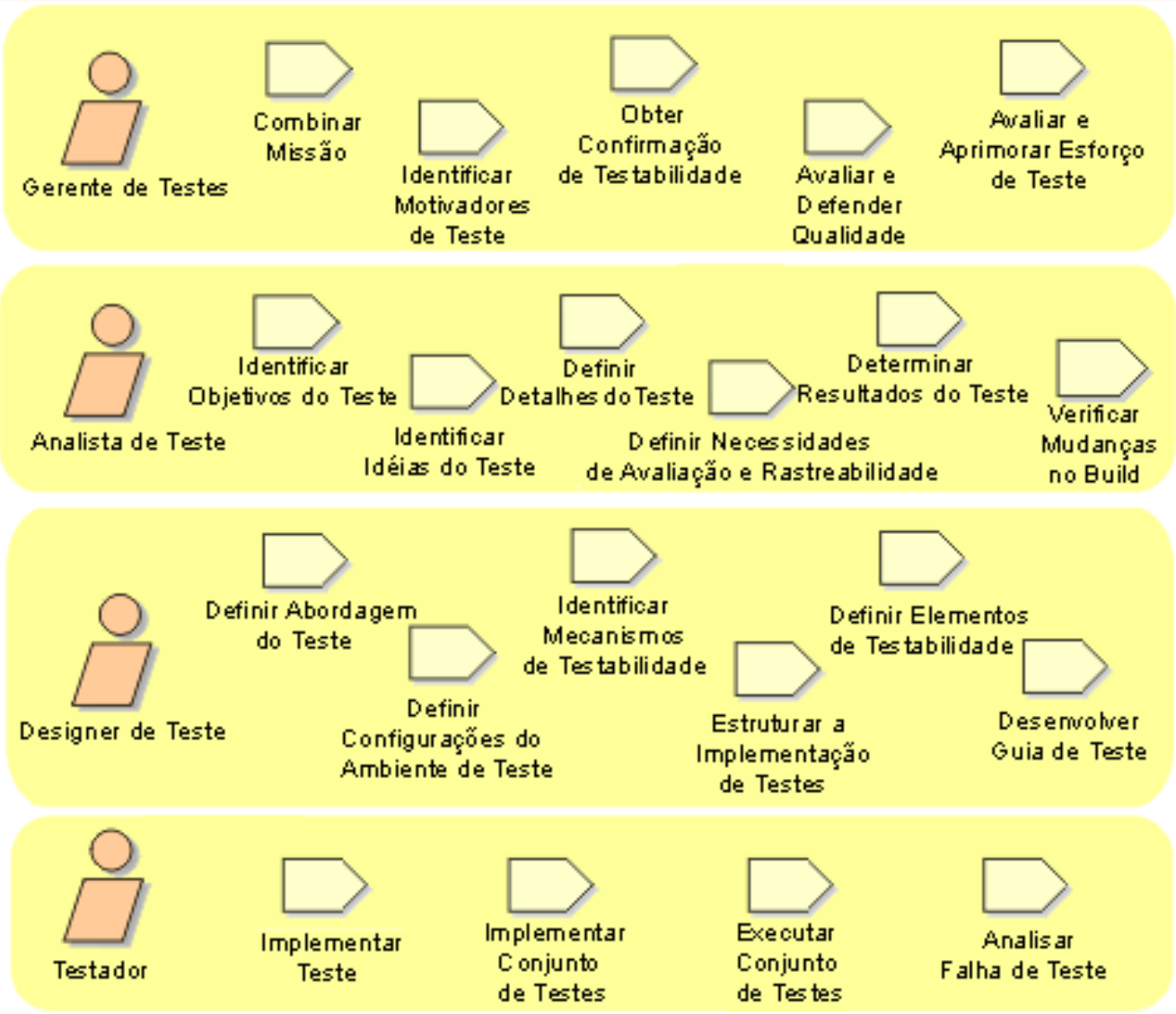
- Realizar os testes e tratar sistematicamente os resultados; fazer a depuração de erros e, eventualmente, retornar os componentes para reavaliação de projeto e implementação.



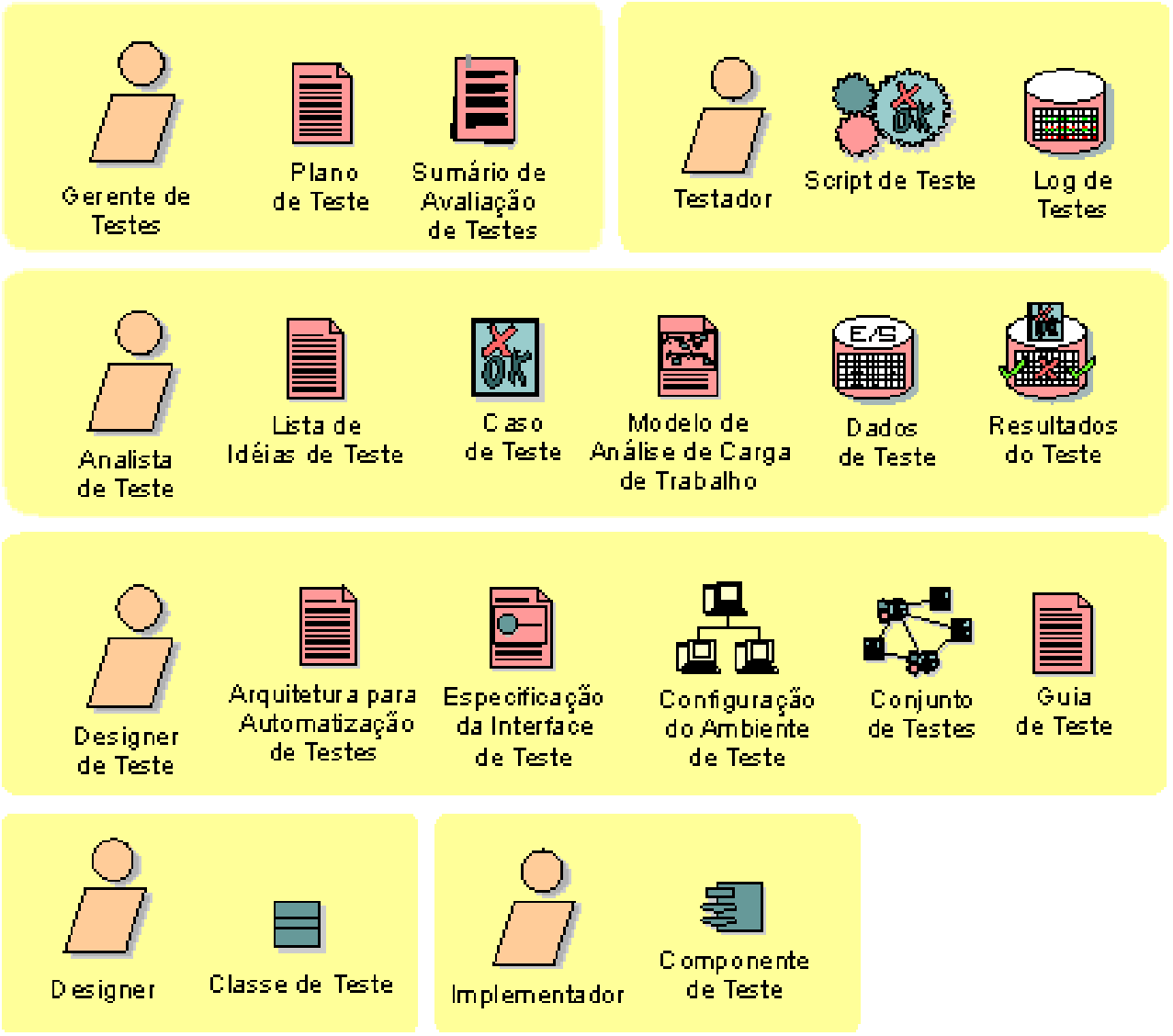
5.5 - Teste - Trabalhadores

- Projetista (designer) de teste
  - Responsável pelo modelo de teste
- Engenheiro de componente
  - Responsável por componentes para automação de procedimentos de teste
- Executor de teste de integração
  - Execução de testes de integração
- Executor de teste de sistema
  - Execução de testes de sistema

5.5 - Teste - Atividades x Trabalhadores

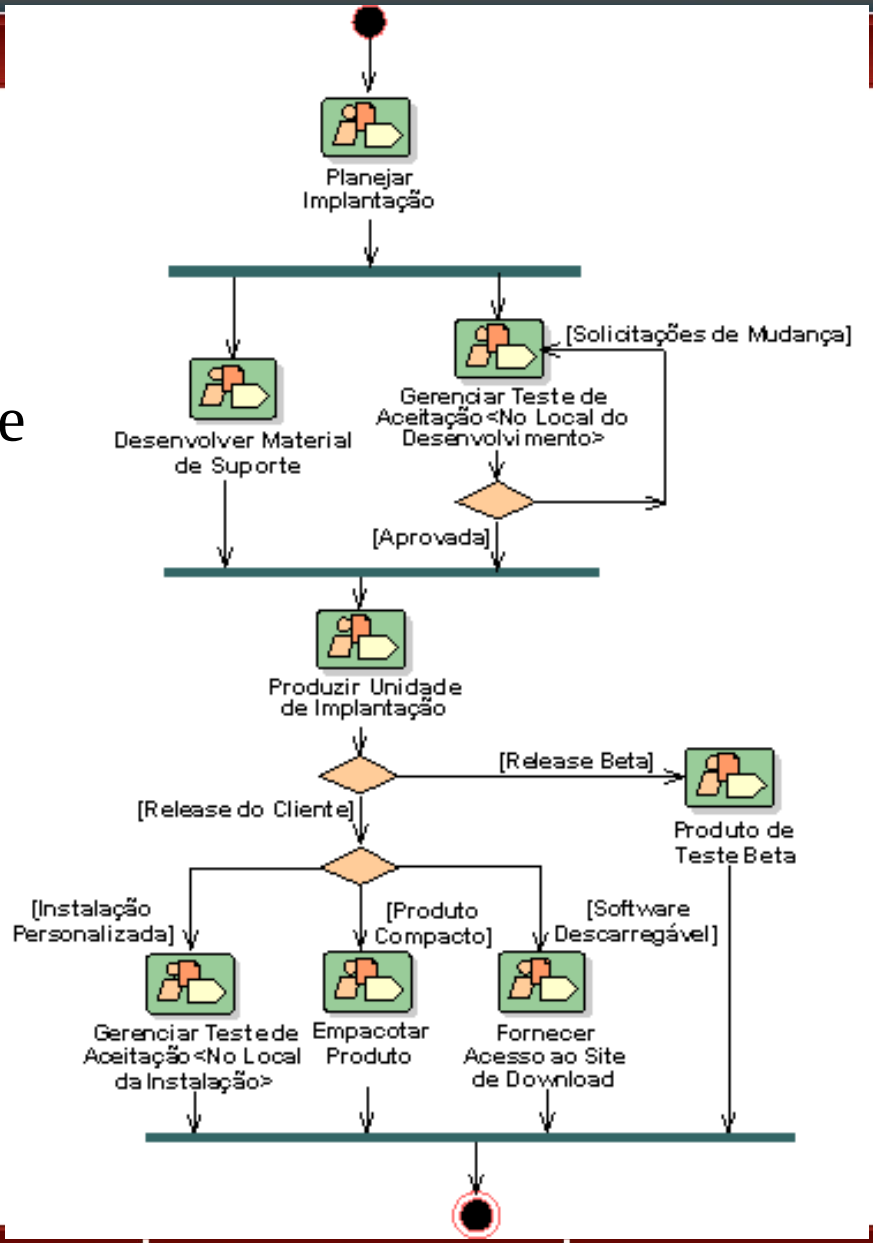


5.5-Teste - Artefatos x Trabalhadores

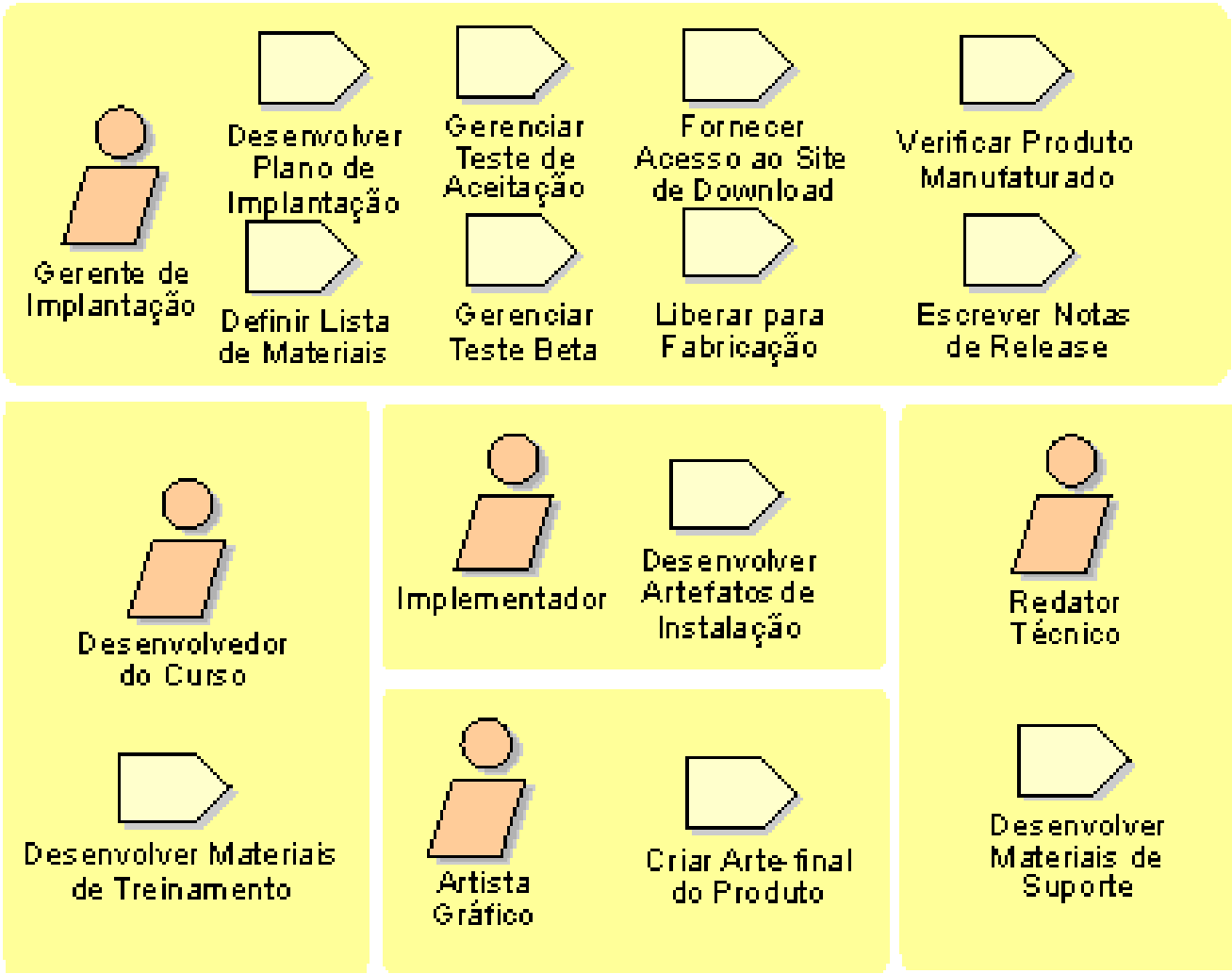


5.6- FT Implantação

Objetivo : Descreve as atividades que garantem que o produto de software será disponibilizado a seus usuários finais.



5.6-Implantação – Atividades x Trabalhadores



5.6-Implantação – Artefatos x Trabalhadores