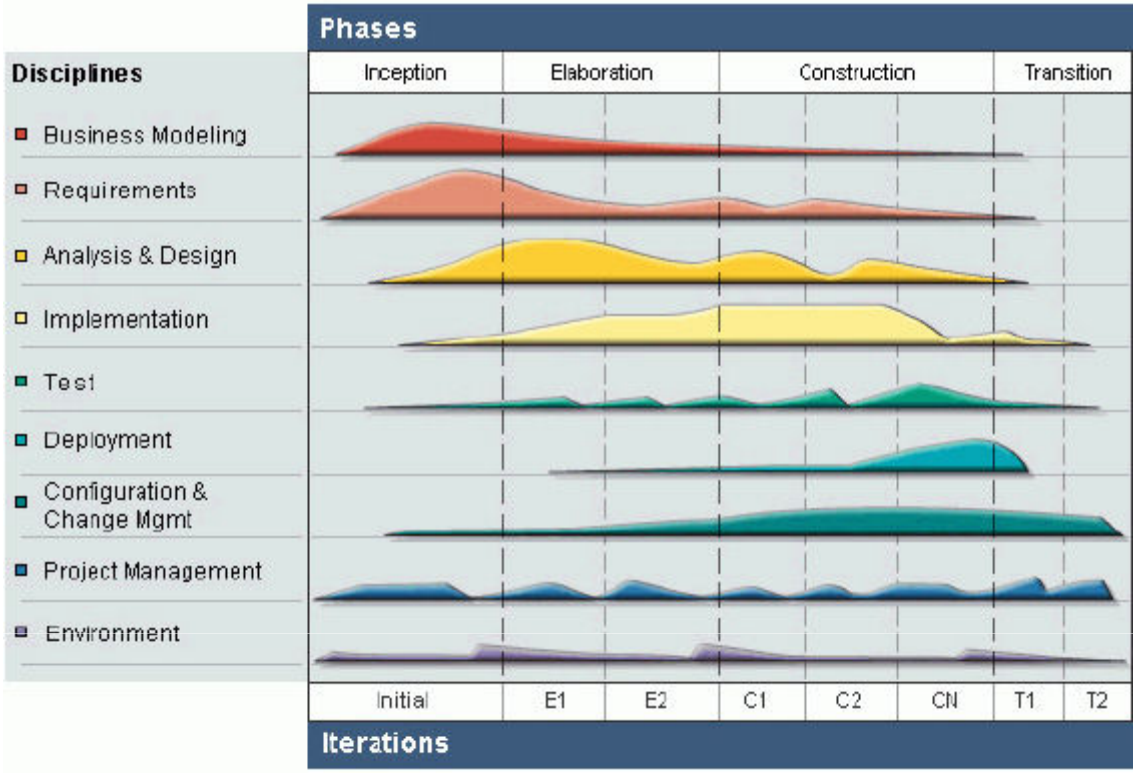


PROCESSO UNIFICADO

INE 5419 – Engenharia de Software II

Prof. Raul Sidnei Wazlawick

UFSC-CTC-INE

2012.1

Conteúdo

- Caracterização do UP
- Fases do UP
- Implementações
  - RUP
  - AUP
  - OpenUP
  - EUP
  - OUM
  - RUP-SE

Processo unificado (UP)

- Criado por Jacobson, Booch, & Rumbaugh, na década de 1990.
- Resultado de mais de 30 anos de experiência acumulada em projetos, notações e processos.
- Primeiro modelo de processo inteiramente adaptado ao uso com a *UML* (*Unified Modeling Language*).
- Concepção foi baseada nas práticas de maior retorno de investimento (ROI) do mercado.

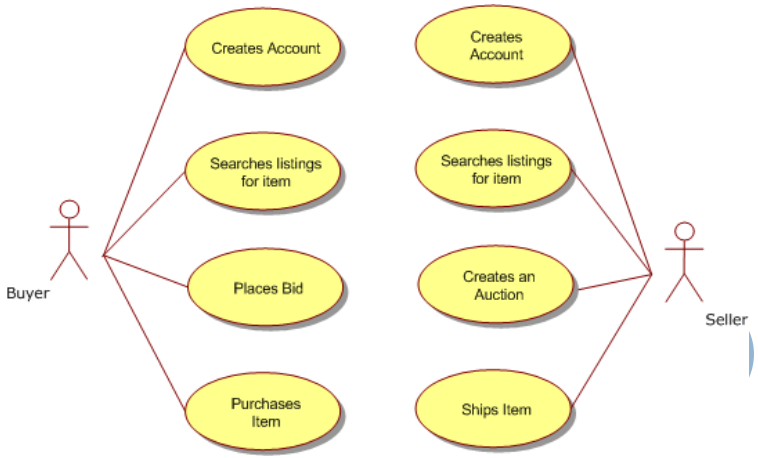


Caracterização

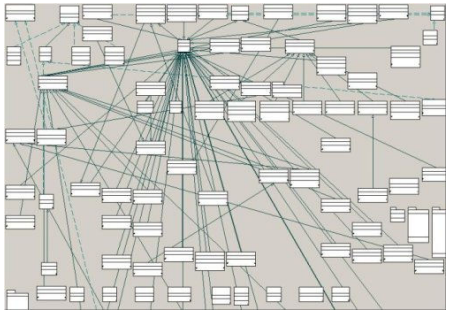
- É dirigido por casos de uso.
- É centrado na arquitetura.
- É iterativo e incremental.
- É focado em riscos.

Dirigido por casos de uso

- Para o UP o conjunto de casos de uso deve definir e esgotar *toda a funcionalidade possível* do sistema.
- Utilidade dos casos de uso:
  - *Definição e validação da arquitetura do sistema*.
  - *Criação dos casos de teste*.
  - *Planejamento das iterações*.
  - *Base para a documentação do usuário*.

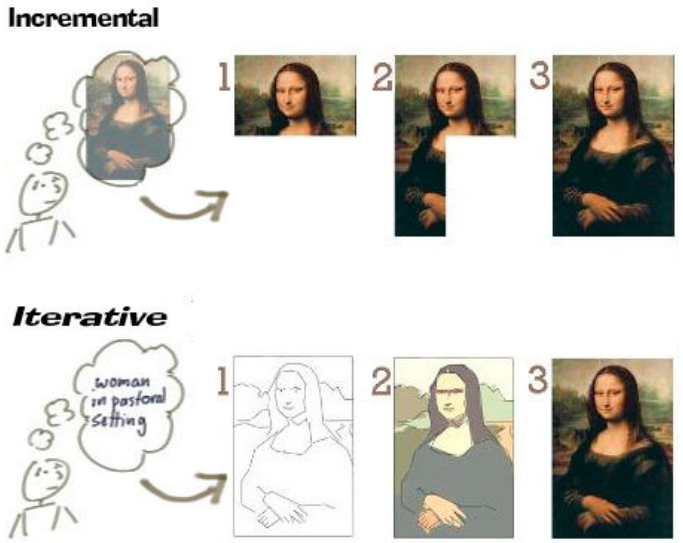


Centrado na arquitetura



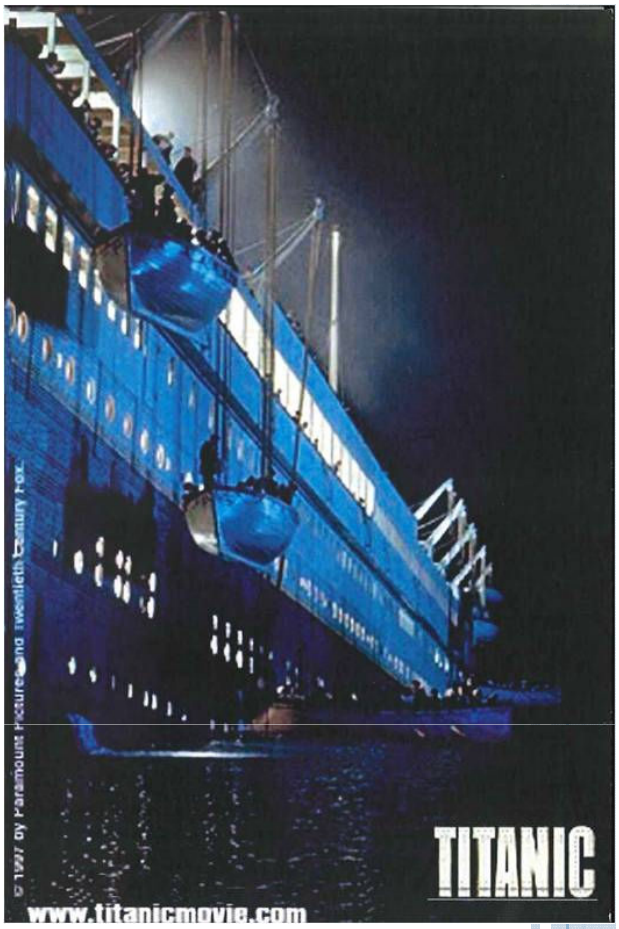
- O UP preconiza que uma sólida arquitetura de sistema deve ser desenvolvida.
- As funcionalidades aprendidas com a elaboração dos diversos casos de uso devem ser integradas a esta arquitetura de forma incremental.
- Segundo UP, a cada ciclo iterativo deve-se incorporar à arquitetura existente as funcionalidades aprendidas com a análise de cada um dos casos de uso abordados no ciclo.
- Assim, fazendo-se a priorização dos casos de uso a partir dos mais críticos ou complexos para os mais triviais e simples, desenvolve-se em um primeiro momento todos os elementos de maior risco para a arquitetura, não ficando muitas surpresas para depois.

ITERATIVO E INCREMENTAL



- UP preconiza o desenvolvimento baseado em ciclos iterativos de duração fixa, onde a cada iteração a equipe incorpora à arquitetura as funcionalidades necessarias para realizar os casos de uso abordados no ciclo.
- Cada ciclo iterativo produz um incremento no *design* do sistema, seja produzindo mais conhecimento sobre seus requisitos e arquitetura, seja produzindo código executável.
- Espera-se que em cada iteração todas as disciplinas previstas sejam executadas com maior ou menor intensidade.
- A integração contínua reduz riscos, facilita os testes e melhora o aprendizado da equipe sobre o sistema, especialmente nos primeiros momentos, quando decisões críticas precisam ser tomadas com relativamente pouco conhecimento sobre o sistema em si.

Focado em riscos



- Em função das priorizações dos casos de uso mais críticos nos primeiros ciclos iterativos, pode-se dizer que o UP é focado em riscos, pois se estes casos de uso são os que apresentam maior risco de desenvolvimento, então devem ser tratados o quanto antes para que os riscos sejam resolvidos enquanto o custo de tratá-los ainda é baixo e o tempo disponível para acomodar surpresas ainda é relativamente grande.
- Os requisitos ou casos de uso de maior risco são os mais imprevisíveis.
  - Assim, estudá-los primeiramente, além de garantir um maior aprendizado sobre o sistema e as decisões arquiteturais mais importantes, também vai fazer com que riscos positivos ou negativos sejam dominados o mais cedo possível.

Fases do UP

- Concepção (*inception*).
- Elaboração (*elaboration*).
- Construção (*construction*).
- Transição (*deployment*).

Concepção

Marco LCO - *Lifecicle Objective Milestone.*

- A fase de concepção no Processo Unificado não deve ser muito longa.
  - De duas semanas a dois meses é o que se recomenda dependendo da dimensão relativa do projeto.
- Nesta etapa são analisados os requisitos de projeto da melhor forma possível.
  - Eles são analisados em abrangência, não em profundidade.
- O conjunto de casos de uso de alto nível é identificado e o esforço necessário para desenvolvê-los calculado.
  - A partir desse cálculo, deve-se fazer um planejamento de longo prazo, procurando acomodar os casos de uso, de acordo com sua prioridade nos diferentes ciclos ao longo do processo de desenvolvimento.
- A fase de concepção envolve também o estudo de viabilidade, pois ao seu final a equipe deve decidir se é viável prosseguir com o projeto, analisadas as questões tecnológicas, de orçamento e de cronograma.

Elaboração
Marco LCA - *Lifecicle Architecture Milestone*

- A fase de elaboração consiste em um detalhamento da análise e realização do projeto para o sistema como um todo.
- A elaboração ocorre em ciclos, com partes de *design* sendo desenvolvidas e integradas ao longo de cada ciclo.
- Objetivos desta fase:
  - Produzir uma arquitetura executável confiável para o sistema.
  - Desenvolver o modelo de requisitos até completar pelo menos 80% dele.
  - Desenvolver um projeto geral para a fase de construção.
  - Garantir que as ferramentas críticas, processos, padrões e regras estão disponíveis para a fase de construção.
  - Entender e eliminar os riscos de alta prioridade do projeto.

- Esta fase permite analisar o domínio do problema de forma mais refinada, permitindo definir uma arquitetura mais adequada e sólida.
- A priorização dos casos de uso mais complexos nesta fase permitirá eliminar ou mitigar os elementos de projeto que apresentam o maior risco.
- Embora o Processo Unificado trabalhe com a perspectiva de acomodação de mudanças, ele procura minimizar seu impacto, mitigando riscos e elaborando uma arquitetura o mais próxima possível do necessário para que as funcionalidades requeridas possam ser desenvolvidas.

CONSTRUÇÃO

MARCO IOC - *INITIAL OPERATIONAL CAPABILITY MILESTONE*

- Na fase de construção, um produto completo e usável deve estar desenvolvido, testado e adequado para uso pelo usuário final.
- A fase de construção também é realizada em ciclos iterativos.
- O projeto é desenvolvido também de forma incremental, com novas funcionalidades sendo adicionadas ao sistema a cada ciclo.
- Objetivos da fase de construção :
  - Descrever os requisitos que ainda faltam.
  - Dar substância ao *design* do sistema.
  - Garantir que o sistema atenda às necessidades dos usuários e que ele se encaixa no contexto geral da organização.
  - Completar o desenvolvimento dos componentes e testá-los, incluindo tanto o software quanto a sua documentação.
  - Minimizar os custos de desenvolvimento pela otimização dos recursos.
  - Obter qualidade adequada tão rápido quanto possível.
  - Desenvolver versões úteis do sistema.

- A fase de construção se caracteriza pela exploração dos casos de uso de baixa e média complexidade, ou seja, os casos de uso padronizados que não vão impactar na arquitetura.
- Como estes casos de uso são padronizados, o esforço de análise e *design* será menor nesta fase, ficando a maior parte do trabalho concentrada na implementação e teste dos componentes da arquitetura dedicados a estes casos de uso.

TRANSIÇÃO
MARCO PR - *PRODUCT RELEASE MILESTONE*

- A fase de transição consiste da colocação do sistema em uso no ambiente final.
- São necessários testes de aceitação e operação, treinamento de usuários e transição de dados a partir de sistemas antigos, que podem ser capturados automaticamente ou digitados.
- Nesta fase também poderá haver a execução do sistema em paralelo com sistemas legados para verificar sua adequação.
- Após a conclusão da fase de transição o sistema entra em evolução, ou seja, depois de aceito e colocado em operação no ambiente final, ele passa a receber atualizações periódicas de forma a corrigir possíveis erros ou implantar novas funcionalidades necessárias.

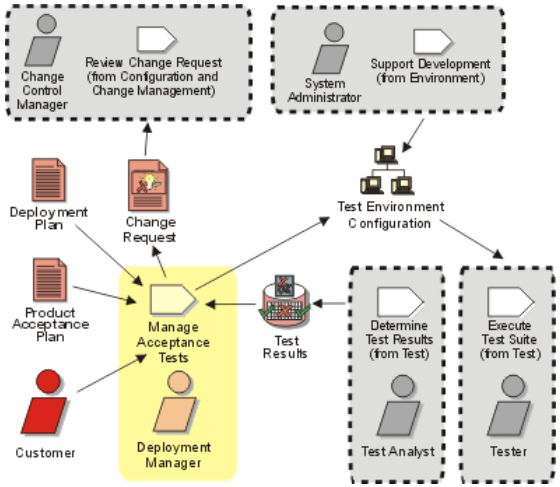
Implementações do UP

- RUP – Rational Unified Process
- AUP - Agile Unified Process
- OpenUp - Open Unified Process
- EUP - Enterprise Unified Process
- OUM - Oracle Unified Method
- RUP-SE - Rational Unified Process–Systems Engineering

RUP – Rational Unified Process

- Mais detalhada e mais antiga implementação do UP.
- Criada por Booch, Rumbaugh e Jacobson através da empresa Rational, desde 2003 subsidiária da IBM.
- Ainda antes de pertencer à IBM, a empresa Rational em 1997 adquiriu várias outras empresas: Verdix, Objectory, Requisite, SQA, Performance Awareness e Pure-Atria, e a partir da experiência acumulada destas empresas, estabeleceu algumas melhores práticas, que seriam a base filosófica para o novo modelo de processo.
- www.rational.com/rup/

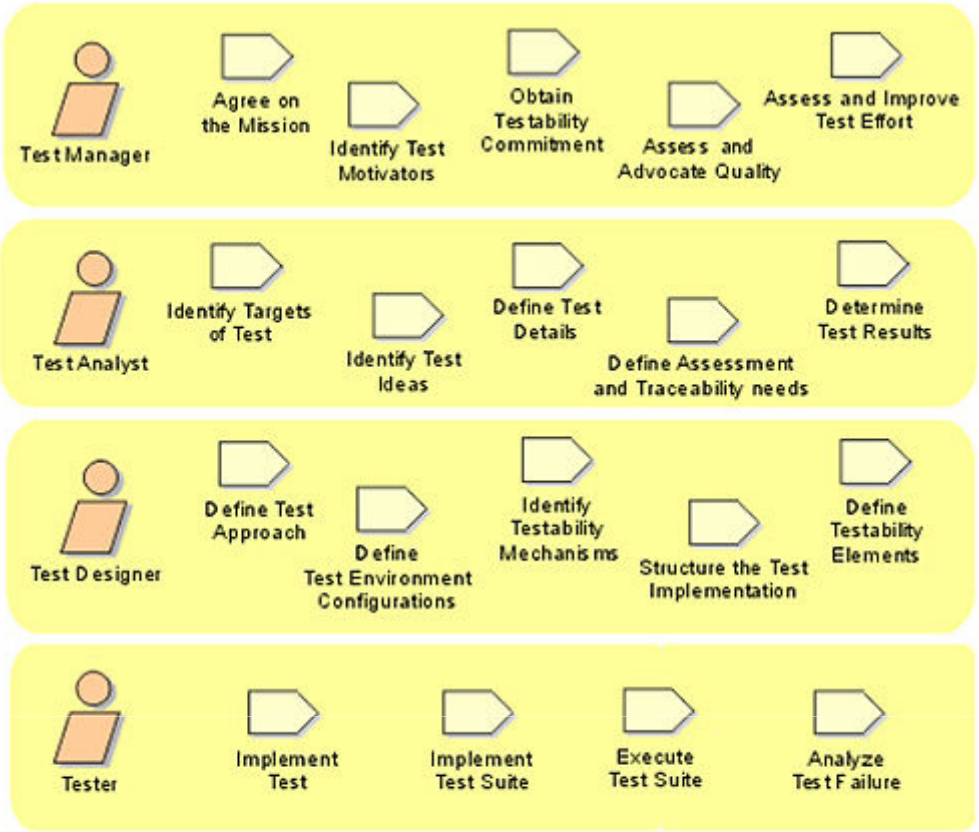
Blocos de construção do RUP



- *Quem*.
  - Um *papel* define um conjunto de habilidades necessário para realizar determinadas atividades.
- *O quê*.
  - O *produto do trabalho* (*work product*) define algo produzido por alguma atividade ou atividade, como diagramas, relatórios ou código funcionando, ou seja, os *artefatos*.
- *Como*.
  - Uma *atividade* descreve uma unidade de trabalho atribuída a um papel que produz um determinado conjunto de artefatos.
- *Quando*.
  - Os *workflows* são grafos que definem as dependências entre as diferentes atividades.

PAPEIS

- Papeis de analista.
- Papeis de desenvolvedor.
- Papeis de testador.
- Papeis de gerente.
- Outros papeis.



Papeis de analista

- Analista de sistemas.
- Designer de negócio.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
- Especificador de requisitos.
- Revisor de requisitos.
- Designer de interface com usuário.

PAPEIS DE ANALISTA

- Analista de sistemas.
  - Ele lidera e coordena a análise de requisitos, definindo o escopo do sistema e seus casos de uso.
  - O analista de sistemas deve ser uma pessoa com boa capacidade de comunicação e negociação, além de ser necessário que ele tenha conhecimento de negócios e tecnologia.
- Designer de negócio.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
- Especificador de requisitos.
- Revisor de requisitos.
- Designer de interface com usuário.

PAPEIS DE ANALISTA

- Analista de sistemas.
- Designer de negócio.
  - Ele faz a modelagem de negócio, usualmente utilizando diagramas de atividade, máquina de estados ou ainda BPMN.
  - Ele atuará nas fases mais iniciais da análise com a produção do modelo de negócio ou ciclo de negócio, que vai colocar o sistema a ser desenvolvido dentro da perspectiva mais ampla da organização.
  - A partir das atividades de negócio identificadas os casos de uso serão posteriormente identificados.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
- Especificador de requisitos.
- Revisor de requisitos.
- Designer de interface com usuário.

PAPEIS DE ANALISTA

- Analista de sistemas.
- Designer de negócio.
- Revisor do modelo de negócios.
  - Ele deve revisar o modelo de negócio verificando se é consistente, coerente e não ambíguo.
  - Deve ter conhecimento profundo do negócio e, se possível da tecnologia a ser usada para sua implementação (o papel poderá ser desempenhado por duas pessoas caso não se consiga uma única pessoa com estas características).
- Analista do processo de negócio.
- Especificador de requisitos.
- Revisor de requisitos.
- Designer de interface com usuário.

Papeis de analista

- Analista de sistemas.
- Designer de negócio.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
  - Ele lidera e coordena a modelagem da organização como casos de uso de negócio.
  - Esta atividade pode ser dispensada, especialmente se não houver necessidade de uma reengenharia de negócio a partir do projeto sendo desenvolvido.
- Especificador de requisitos.
- Revisor de requisitos.
- Designer de interface com usuário.

PAPEIS DE ANALISTA

- Analista de sistemas.
- Designer de negócio.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
- Especificador de requisitos.
  - Ele é responsável pelos requisitos que usualmente são representados como casos de uso, inicialmente de alto nível, e depois expandidos na sua forma essencial.
  - Além dos requisitos funcionais, representados nos casos de uso, o especificador de requisitos deve também especificar os requisitos suplementares, ou seja, os aspectos não funcionais gerais do sistema.
- Revisor de requisitos.
- Designer de interface com usuário.

PAPEIS DE ANALISTA

- Analista de sistemas.
- Designer de negócio.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
- Especificador de requisitos.
- Revisor de requisitos.
  - Ele é responsável pela revisão minuciosa dos casos de uso e especificações suplementares verificando se são completos, coerentes e não ambíguos.
  - Além disso, deve garantir que o documento de casos de uso esteja escrito em conformidade com os padrões adotados na empresa.
- Designer de interface com usuário.

Papeis de analista

- Analista de sistemas.
- Designer de negócio.
- Revisor do modelo de negócios.
- Analista do processo de negócio.
- Especificador de requisitos.
- Revisor de requisitos.
- Designer de interface com usuário.
  - Ele coordena as atividades de prototipação de interfaces e de *design* de interface com usuário.
  - Ele é o responsável pelos requisitos de interface.
  - Ele não implementa a interface, ficando sua responsabilidade restrita ao *design* visual e de usabilidade.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
  - Ele é responsável pela produção e teste de unidade do código fonte.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
  - Ele deve ser responsável por garantir a qualidade do código fonte implementado, verificando se este segue padrões estabelecidos e boas práticas de programação.
  - O revisor deve ser um profundo conhecedor da linguagem de programação utilizada.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
  - Quando os implementadores liberam componentes já testados cabe ao integrador incluí-los em uma versão operacional do sistema para gerar uma nova versão (ou *build*).
  - O integrador deverá elaborar e realizar os testes de integração caso essa atividade não tenha sido atribuída ao analista de teste.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
  - O arquiteto é responsável pelo *design* das camadas e/ou partições do sistema, ou seja, da sua estrutura em nível mais alto, incluindo o *design* de pacotes, componentes e sua distribuição em diferentes nodos de processamento.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
  - O revisor de arquitetura deve avaliar se o trabalho do arquiteto de software efetivamente leva a uma arquitetura sólida e estável.
  - Visto que a arquitetura costuma conter as decisões mais cruciais para o sucesso de um sistema, este papel é importante para garantir que as decisões do arquiteto sejam as mais efetivas possíveis.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
  - O *design*er toma decisões sobre o *design* das classes, seus atributos e associações, bem como de métodos a serem implementados.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
  - Ele deve revisar o *design* verificando se as melhores práticas e padrões foram adotadas.
- Designer de banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
  - Ele é responsável pelo *design* das tabelas e códigos associados ao banco de dados.
- Designer de cápsula.

Papeis de desenvolvedor

- Implementador.
- Revisor de código.
- Integrador.
- Arquiteto de software.
- Revisor de arquitetura.
- Designer.
- Revisor de design.
- Designer de banco de dados.
- Designer de cápsula.
  - Este papel só existe em projetos de sistemas de tempo real.
  - Ele é responsável por assegurar que o sistema responda prontamente a eventos de acordo com os requisitos de tempo real.

Papeis de testador

- Designer de teste.
- Analista de teste.
- Testador.
- Gerente de teste.

PAPEIS DE TESTADOR

- Designer de teste.
  - Ele é responsável por definir as técnicas e estratégias de teste a serem adotadas.
  - Não deve ser confundido com o *analista* de teste que elabora os testes de sistema; o *design*er de teste define a estratégia de teste, mas não os testes reais.
- Analista de teste.
- Testador.
- Gerente de teste.

Papeis de testador

- Designer de teste.
- Analista de teste.
  - Ele é responsável pelo projeto e elaboração dos casos de teste a serem aplicados ao sistema.
  - Usualmente o teste de unidade é feito pelo próprio programador e o teste de integração pelo integrador, sendo que o analista de teste costuma atuar principalmente no teste de sistema.
- Testador.
- Gerente de teste.

Papeis de testador

- Designer de teste.
- Analista de teste.
- Testador.
  - O testador é responsável pela realização efetiva dos testes.
  - Ele verifica a melhor abordagem a ser utilizada de acordo com as estratégias definidas pelo *design*er de teste e analista de teste e implementa o teste além de executá-lo para verificar se o componente ou o sistema passam no teste.
  - Ele deve registrar os resultados dos testes e no caso de componentes que não passem nos testes deve informar os respectivos responsáveis para providências de correção.
- Gerente de teste.

Papeis de testador

- Designer de teste.
- Analista de teste.
- Testador.
- Gerente de teste.
  - Ele coordena toda a atividade de teste do sistema.

Papeis de gerente

- Engenheiro de processo.
- Gerente de projeto.
- Gerente de controle de mudança.
- Gerente de configuração.
- Gerente de implantação.
- Revisor do projeto.

PAPEIS DE GERENTE

- Engenheiro de processo.
  - Ele é o responsável pela aplicação do processo de desenvolvimento, devendo configurar e ajustar o processo de acordo com as necessidades da equipe e dos projetos.
- Gerente de projeto.
- Gerente de controle de mudança.
- Gerente de configuração.
- Gerente de implantação.
- Revisor do projeto.

Papeis de gerente

- Engenheiro de processo.
- Gerente de projeto.
  - Ele é o responsável por um ou mais projetos específicos, devendo planejar as atividades e alocar os recursos físicos e humanos, bem como acompanhar o projeto garantindo que prazos e orçamentos sejam cumpridos e tomando decisões de correção de rumo quando necessário.
- Gerente de controle de mudança.
- Gerente de configuração.
- Gerente de implantação.
- Revisor do projeto.

Papeis de gerente

- Engenheiro de processo.
- Gerente de projeto.
- Gerente de controle de mudança.
  - Ele é o responsável pelo controle das requisições de mudança, seja do cliente, seja da própria equipe de desenvolvimento, e acompanha o atendimento a elas.
- Gerente de configuração.
- Gerente de implantação.
- Revisor do projeto.

Papeis de gerente

- Engenheiro de processo.
- Gerente de projeto.
- Gerente de controle de mudança.
- Gerente de configuração.
  - Ele planeja e disponibiliza o ambiente para que os desenvolvedores e integradores possam realizar suas atividades.
  - Ele garante que todos tenham acesso aos artefatos necessários.
- Gerente de implantação.
- Revisor do projeto.

Papeis de gerente

- Engenheiro de processo.
- Gerente de projeto.
- Gerente de controle de mudança.
- Gerente de configuração.
- Gerente de implantação.
  - Ele planeja e acompanha a implantação do sistema junto aos clientes e usuários.
- Revisor do projeto.

PAPEIS DE GERENTE

- Engenheiro de processo.
- Gerente de projeto.
- Gerente de controle de mudança.
- Gerente de configuração.
- Gerente de implantação.
- Revisor do projeto.
  - Ele é responsável pela revisão dos planos e avaliações do projeto ao longo do seu desenvolvimento.

Outros papeis

- Interessados (ou envolvidos).
- Desenvolvedor de curso.
- Artista gráfico.
- Especialista em ferramentas.
- Administrador do sistema.
- Redator técnico.

Outros papeis

- Interessados (ou envolvidos).
  - São todos aqueles afetados pelo sistema ou seu desenvolvimento.
- Desenvolvedor de curso.
- Artista gráfico.
- Especialista em ferramentas.
- Administrador do sistema.
- Redator técnico.

Outros papeis

- Interessados (ou envolvidos).
- Desenvolvedor de curso.
  - É o responsável pela criação de material de treinamento para usuários.
- Artista gráfico.
- Especialista em ferramentas.
- Administrador do sistema.
- Redator técnico.

Outros papeis

- Interessados (ou envolvidos).
- Desenvolvedor de curso.
- Artista gráfico.
  - É o responsável pela criação da arte final do produto, de sua embalagem e de outros artefatos correlatos.
- Especialista em ferramentas.
- Administrador do sistema.
- Redator técnico.

Outros papeis

- Interessados (ou envolvidos).
- Desenvolvedor de curso.
- Artista gráfico.
- Especialista em ferramentas.
  - Ele dá suporte aos desenvolvedores pela seleção, instalação e treinamento no uso das ferramentas de apoio ao desenvolvimento e gerenciamento dos projetos.
- Administrador do sistema.
- Redator técnico.

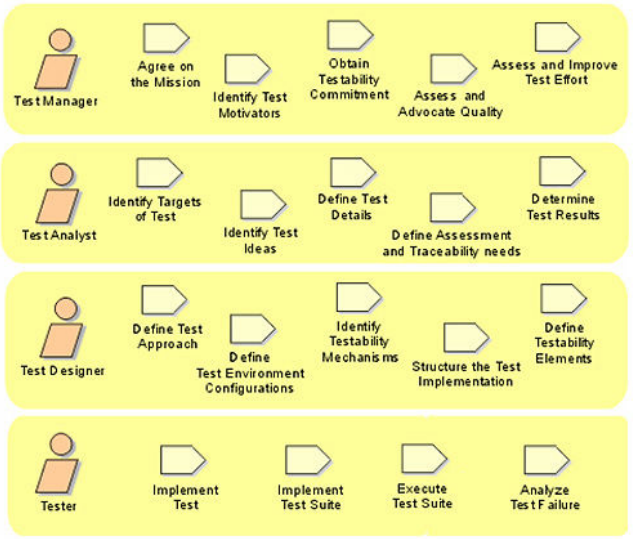
Outros papeis

- Interessados (ou envolvidos).
- Desenvolvedor de curso.
- Artista gráfico.
- Especialista em ferramentas.
- Administrador do sistema.
  - Ele mantém o ambiente de desenvolvimento, cuidando do hardware, versões de software, rede, etc.
- Redator técnico.

Outros papeis

- Interessados (ou envolvidos).
- Desenvolvedor de curso.
- Artista gráfico.
- Especialista em ferramentas.
- Administrador do sistema.
- Redator técnico.
  - Ele é o responsável pela redação final tanto dos manuais quanto de partes do sistema orientadas a texto.

ATIVIDADES

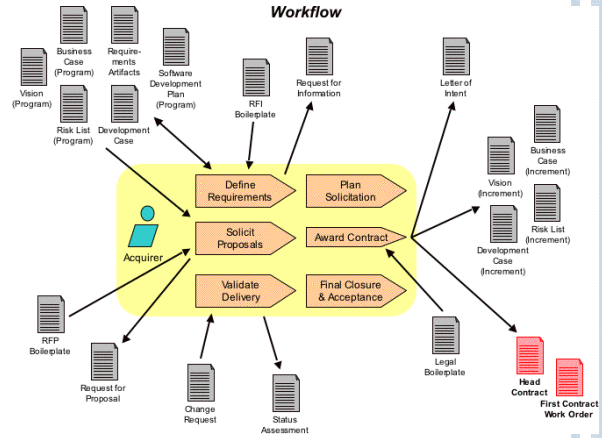


- *Atividades* são unidades de trabalho executadas por um indivíduo que exerce um papel dentro do processo.
- Toda atividade deve produzir um resultado palpável em termos de criação ou alteração consistente de artefatos como modelos, elementos (classes, atores, código etc.) ou planos.
- Uma atividade não deve ser muito curta (poucas horas) nem muito longa (vários dias).
  - Sugere-se pensar em atividades que possam ser realizadas em períodos de 1 a 3 dias, porque esta duração facilita o acompanhamento.
- Uma atividade pode ser detalhada em *passos*.

Tipos de passos de atividade

- Passos de *pensamento*:
  - a pessoa exercendo o papel compreende a natureza da atividade, obtém e examina os artefatos de entrada e formula a saída.
- Passos de *realização*:
  - a pessoa exercendo o papel cria ou atualiza artefatos.
- Passos de *revisão*:
  - a pessoa exercendo o papel inspeciona os resultados em função de algum critério.

ARTEFATOS



- Um artefato pode ser um diagrama, modelo, elemento de modelo, texto, código fonte, código executável etc., ou seja, qualquer tipo de produto criado ao longo do processo de desenvolvimento de software.
- Artefatos podem ser por sua vez compostos por outros artefatos, como por exemplo, uma classe contida no modelo conceitual.
- Como os artefatos mudam com o passar do tempo, pode ser interessante submetê-los a um controle de versões. Porém, o controle de versão normalmente é exercido no artefato de mais alto nível e não em elementos individuais.
- Os artefatos são as entradas para as atividades e também são suas saídas.
- Artefatos de saída ainda podem ser classificados em *entregas*, que são artefatos entregues ao cliente.
- Artefatos (especialmente os de texto) também podem ser definidos por *templates*, isto é, modelos de documentos que dão uma forma geral ao artefato.

Tipos de artefatos

- Artefatos de gerenciamento
- Artefatos de gerenciamento de configuração e mudança
- Artefatos de ambiente
- Artefatos de modelo de negócio
- Artefatos de requisitos
- Artefatos de design
- Artefatos de implementação
- Artefatos de teste
- Artefatos de implantação

WORKFLOWS

- As atividades a serem executadas dentro de cada disciplina são definidas a partir de grafos direcionados chamados *workflows*.
- Um *workflow* define um conjunto de atividades e um conjunto de papéis responsáveis por cada atividade.
- Além disso, o *workflow* indica as dependências entre as diferentes atividades, ou seja, quais atividades dependem logicamente de quais outras atividades para poderem ser executadas.
- Essa dependência pode se dar em diferentes níveis de intensidade, porém, sendo algumas absolutamente necessárias e outras meramente sugeridas.

TIPOS DE WORKFLOW

- *Workflow núcleo* (*core*),
  - que define a forma geral de condução de uma dada disciplina.
- *Workflow detalhe*,
  - que apresenta um refinamento do *workflow* núcleo, indicando atividades em um nível mais detalhado, bem como artefatos de entrada e saída de cada atividade.
- *Planos de iteração*,
  - que consistem em uma instanciação do processo para uma iteração específica.
  - Embora o RUP tenha uma descrição geral dos *workflows* para cada atividade, elas usualmente ocorrem de forma diferente em projetos diferentes e mesmo em ciclos diferentes dentro do mesmo projeto.
  - Assim, o plano de iteração consiste em especificar atividades concretas a serem realizadas de fato dentro de uma iteração planejada.

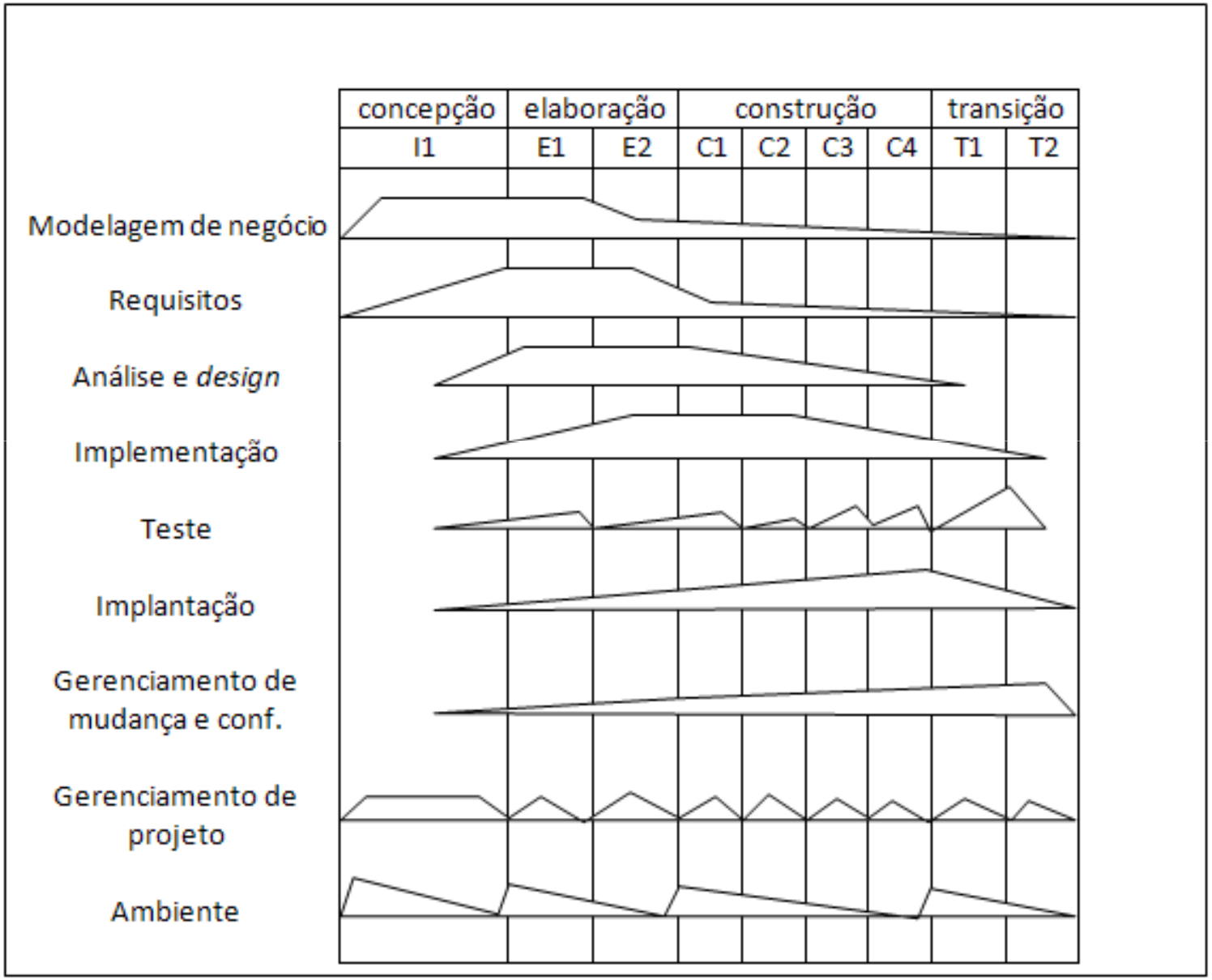
Outros elementos RUP

- *Procedimentos* (*guidelines*)
  - Mostram um detalhamento das atividades de forma que não apenas pessoas acostumadas ao processo, mas também novatos, possam saber o que devem fazer.
- *Templates*
  - São modelos ou protótipos de artefatos. Eles podem ser usados para criar os respectivos artefatos. Usualmente devem estar disponíveis na ferramenta usada para criar e gerenciar o artefato.
- *Mentores de ferramenta*
  - Consistem na descrição detalhada de como realizar uma atividade em uma ferramenta específica.

Disciplinas RUP

- Disciplinas de projeto:
  - Modelagem de negócio.
  - Requisitos.
  - Análise e *design*.
  - Implementação.
  - Teste.
  - Implantação.
- Disciplinas de suporte :
  - Gerenciamento de mudança e configuração.
  - Gerenciamento de projeto.
  - Ambiente.

ARQUITETURA RUP



Gerenciamento de projeto

- Gerenciar um projeto consiste em balancear objetivos que competem entre si, gerenciar riscos e superar restrições, com o objetivo de obter um produto que atenda às necessidades dos clientes (os que pagam pelo desenvolvimento) e dos usuários finais.

Plano de projeto (ou plano de fase)

- O *plano de medição* (*measurement plan*),
  - que estabelece as métricas a serem usadas para definir o andamento do projeto ao longo de sua execução.
- O *plano de gerenciamento de riscos*,
  - que detalha como os riscos serão tratados ao longo do projeto, criando atividades de mitigação e monitoramento de riscos e atribuindo responsabilidades quando necessário.
- A *lista de riscos*,
  - que é uma lista ordenada dos riscos conhecidos e ainda não resolvidos, em ordem decrescente de importância, juntamente com planos de mitigação e contingência quando for o caso.
- O *plano de resolução de problemas*,
  - que descreve como problemas identificados ao longo do projeto devem ser reportados, analisados e resolvidos.
- O *plano de aceitação de produto*,
  - que descreve como o produto será avaliado pelo cliente para verificar se satisfaz as necessidades.

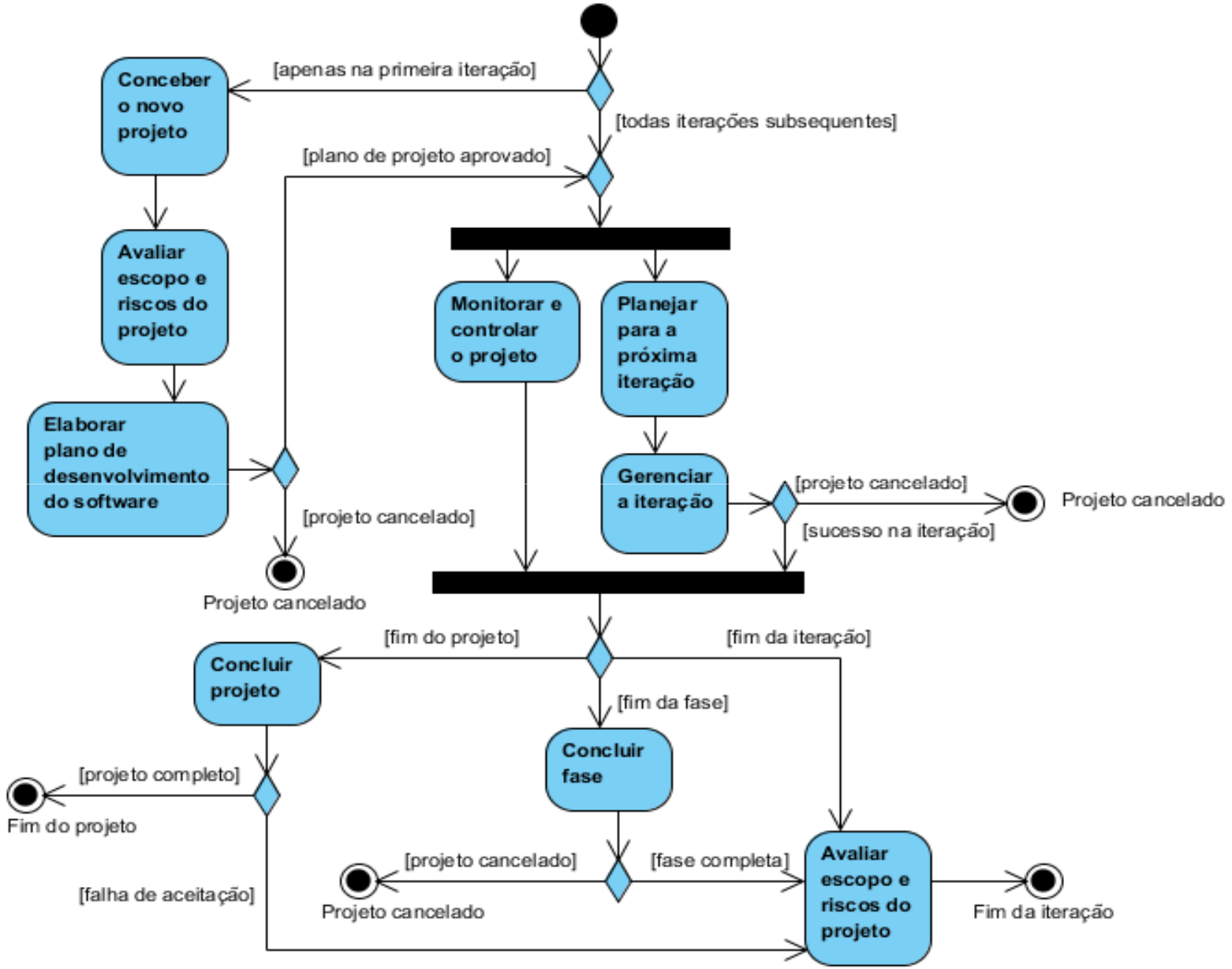
Planos de iteração

- São planos mais detalhados do que o plano de fase.
- Cada plano de iteração inclui um cronograma de atividades atribuídas a responsáveis, com recursos alocados, prazos e dependências.

Entradas para construir um plano de iteração

- O ***plano da fase*** corrente incluído no plano de projeto.
- Informação sobre o ***status do projeto*** (por exemplo, atrasado, em dia, com grandes riscos em aberto, etc.).
- Uma ***lista de casos de uso*** que pelo plano da fase devem ser finalizados na iteração corrente.
- Uma ***lista dos riscos*** que devem ser tratados na iteração corrente.
- Uma ***lista das modificações*** que devem ser incorporadas ao produto na iteração corrente (defeitos e erros encontrados ou mudanças de funcionalidade).

WORKFLOW DE GERENCIAMENTO DE PROJETO



Modelagem de negócio

- Entender a estrutura e a dinâmica da organização alvo na qual o software será utilizado.
- Entender os problemas atuais na organização alvo e identificar potenciais melhorias que podem ser produzidas com o software.
- Certificar que clientes, usuários e desenvolvedores tenham um conhecimento comum da organização alvo.
- Derivar os requisitos para suportar estas melhorias.

Cenários

- Organograma.
- Modelagem de domínio.
- Uma empresa, vários sistemas.
- Modelo de negócio genérico.
- Novo negócio.
- Renovação.

Cenários

- Organograma.
  - Pode-se querer apenas construir um organograma da organização para saber quais são os setores e responsabilidades.
  - Neste caso a modelagem de negócio usualmente acontece apenas na fase de concepção.
- Modelagem de domínio.
- Uma empresa, vários sistemas.
- Modelo de negócio genérico.
- Novo negócio.
- Renovação.

Cenários

- Organograma.
- Modelagem de domínio.
  - Se o objetivo for construir aplicações para gerenciar e apresentar informação, então usualmente faz-se a modelagem de negócio juntamente com a modelagem do domínio, ou seja, é um modelo de informação estático onde os *workflows* da empresa não são considerados.
  - A modelagem de domínio usualmente é feita nas fases de concepção e elaboração.
- Uma empresa, vários sistemas.
- Modelo de negócio genérico.
- Novo negócio.
- Renovação.

Cenários

- Organograma.
- Modelagem de domínio.
- Uma empresa, vários sistemas.
  - Pode-se estar a ponto de desenvolver toda uma família de sistemas para uma empresa.
  - Neste caso, a modelagem de negócio vai tratar não apenas de um sistema, mas de vários projetos e poderá inclusive ser tratada como um projeto a parte.
  - Ela ajudará a descobrir os requisitos dos sistemas individuais, bem como determinar uma arquitetura comum para a família de sistemas.
- Modelo de negócio genérico.
- Novo negócio.
- Renovação.

Cenários

- Organograma.
- Modelagem de domínio.
- Uma empresa, vários sistemas.
- Modelo de negócio genérico.
  - Se o objetivo for a construção de um ou mais aplicativos que sirvam a um grupo de empresas, então a modelagem de negócio poderá ser útil para ajudar a alinhar as empresas a uma visão de negócio, ou, se isso não for possível, obter uma visão de negócio onde as especificidades das empresas seja visível.
- Novo negócio.
- Renovação.

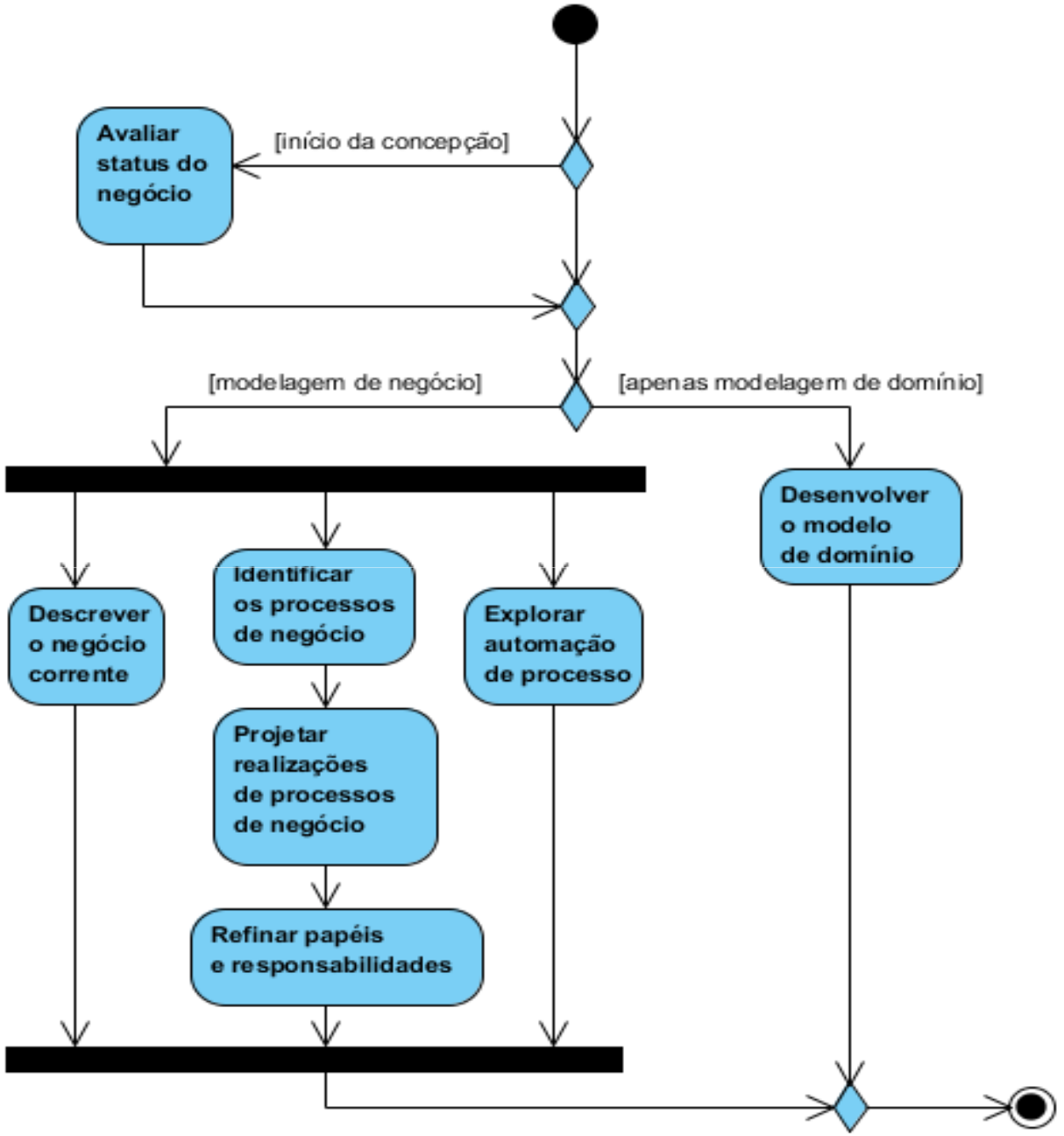
Cenários

- Organograma.
- Modelagem de domínio.
- Uma empresa, vários sistemas.
- Modelo de negócio genérico.
- Novo negócio.
  - Se uma organização resolve iniciar um novo negócio e todo um conjunto de sistemas de informação deve ser desenvolvido para dar suporte a ele, então um esforço significativo de modelagem de negócio deve ser realizado.
  - Neste caso, o objetivo da modelagem de negócio não é apenas encontrar requisitos, mas verificar a viabilidade efetiva do novo negócio.
  - Assim, a modelagem de negócio nesta situação usualmente é um projeto a parte.
- Renovação.

Cenários

- Organograma.
- Modelagem de domínio.
- Uma empresa, vários sistemas.
- Modelo de negócio genérico.
- Novo negócio.
- Renovação.
  - Se uma organização resolve renovar completamente seu modo de fazer negócio, então a modelagem de negócio será um projeto a parte executado em várias etapas:
    - visão do novo negócio,
    - engenharia reversa do negócio existente,
    - engenharia direta do novo negócio e
    - instalação do novo negócio.

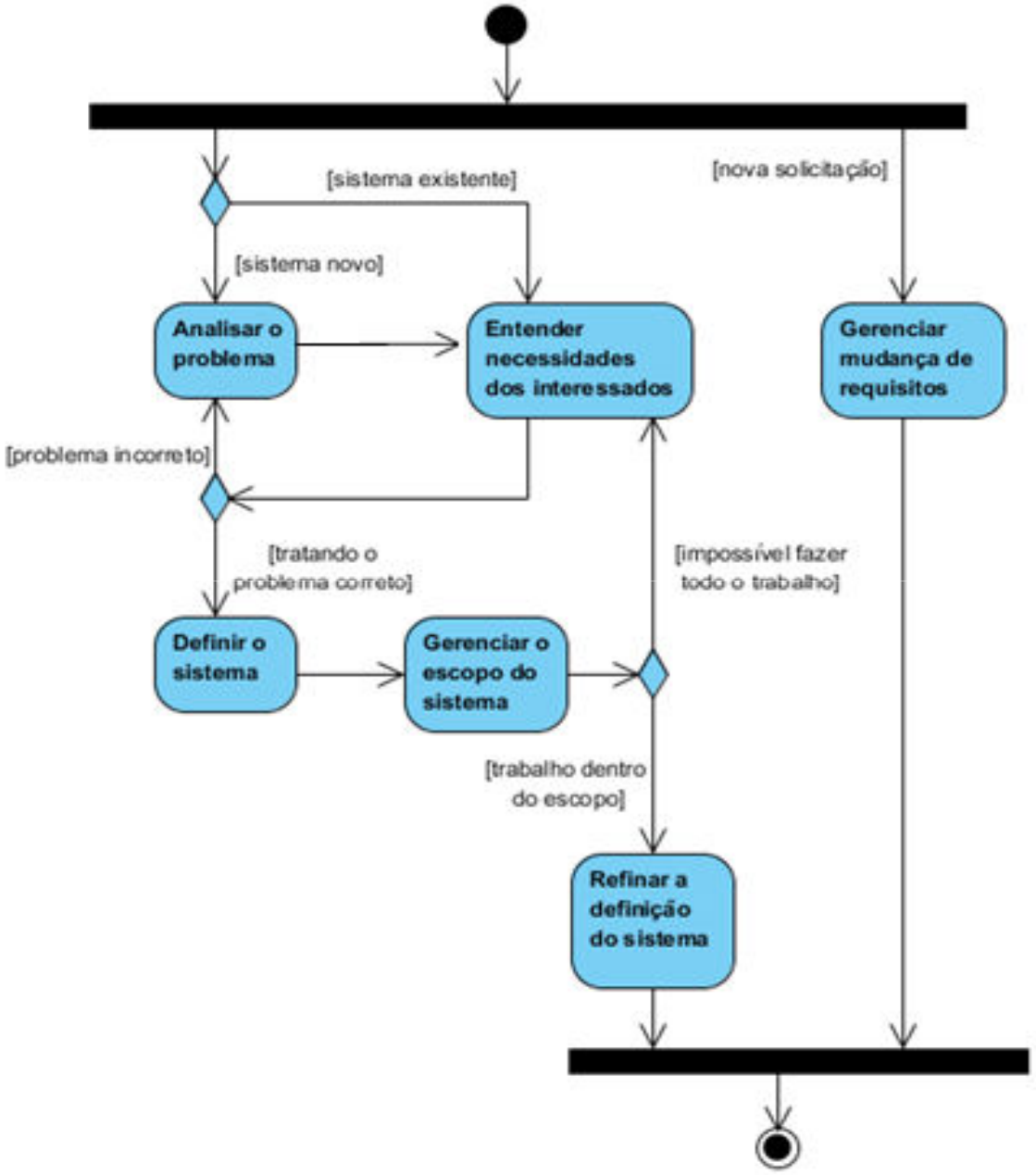
WORKFLOW DA MODELAGEM DE NEGÓCIO



Requisitos

- Objetivos:
  - Estabelecer e manter concordância com o cliente e outros interessados sobre o que o sistema deve fazer, incluindo o porquê.
  - Fornecer aos desenvolvedores uma melhor compreensão sobre os requisitos do sistema.
  - Delimitar escopo do sistema, ou seja, o que pertence e o que não pertence ao sistema.
  - Prover uma base para o planejamento técnico das iterações.
  - Prover uma base para a estimação de custo e tempo de desenvolvimento.
  - Definir uma interface com usuário focada em suas necessidades e objetivos.

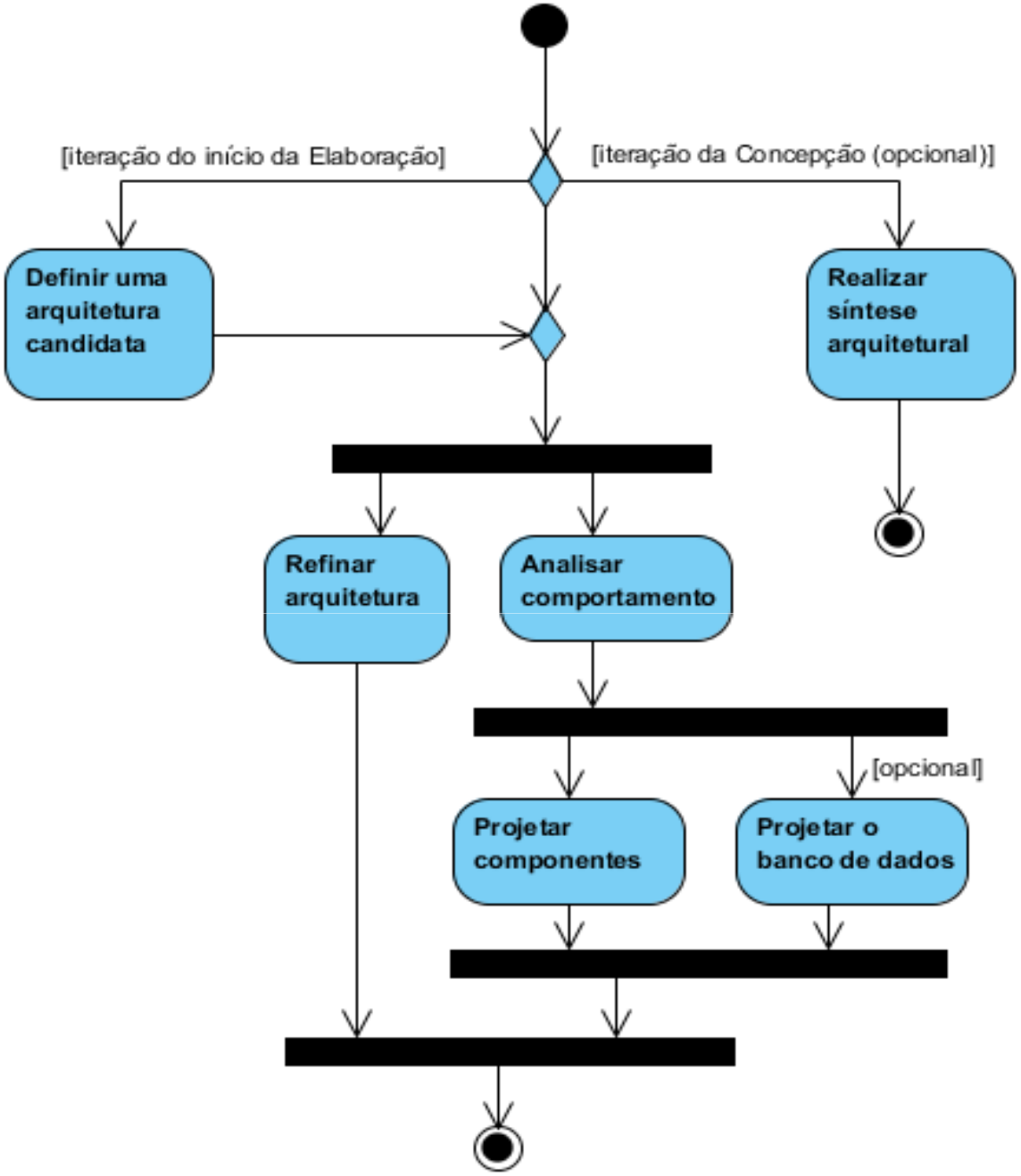
WORKFLOW DE REQUISITOS



Análise e design

- Análise implica no estudo do problema de forma mais aprofundada. Requisitos são detalhados e incluídos em modelos de análise como o modelo conceitual, de interação e funcional. Já o *design* consiste em apresentar uma possível solução tecnológica para o modelo de análise.
- Para o RUP a disciplina de análise e *design* tem como característica principal a modelagem, ou seja, a transformação dos requisitos em modelos úteis para a geração de código.

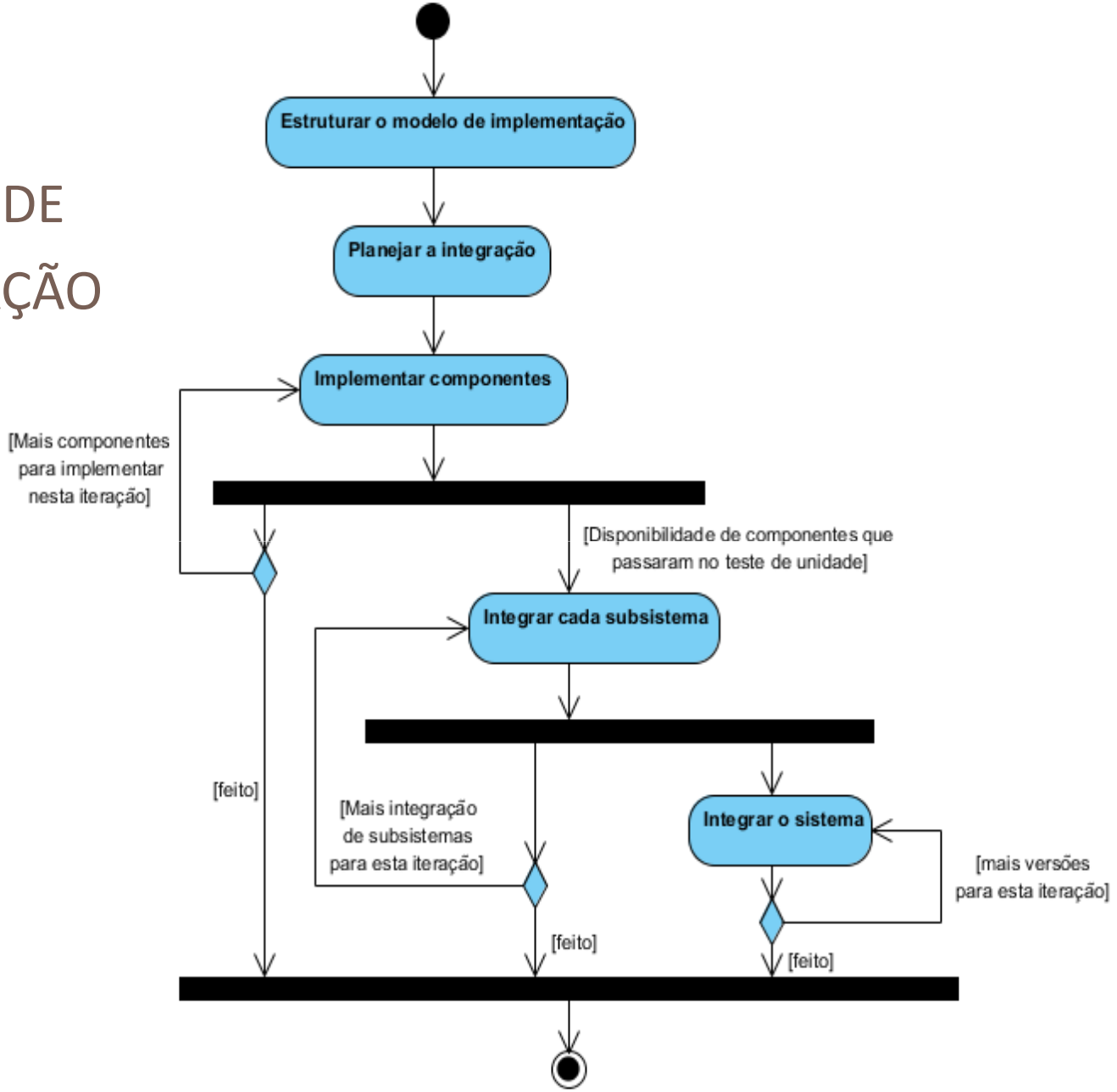
WORKFLOW DE ANÁLISE E DESIGN



IMPLEMENTAÇÃO

- *Versões* (*builds*).
  - São versões operacionais de um sistema que introduzem novas funcionalidades.
  - Versões devem ser controladas e rastreadas para que alterações possam ser desfeitas em caso de necessidade.
  - Projetos típicos apresentam novas versões regularmente; pelo menos uma versão por semana é desejável, podendo chegar a uma por dia.
- *Integração*.
  - A integração consiste na combinação de pelo menos duas partes de software independentes.
  - A integração no RUP deve ser contínua, como nos métodos ágeis e diferente da integração por fase que ocorre no modelo cascata com subprojetos.
  - RUP prevê que no mínimo uma integração por ciclo deva acontecer, mas o ideal é que aconteçam várias integrações por dia.
- *Protótipos*.
  - Devem ser usados primordialmente para reduzir risco, seja para obter uma melhor compreensão dos requisitos, seja para obter provas de conceito de arquitetura ou usabilidade.

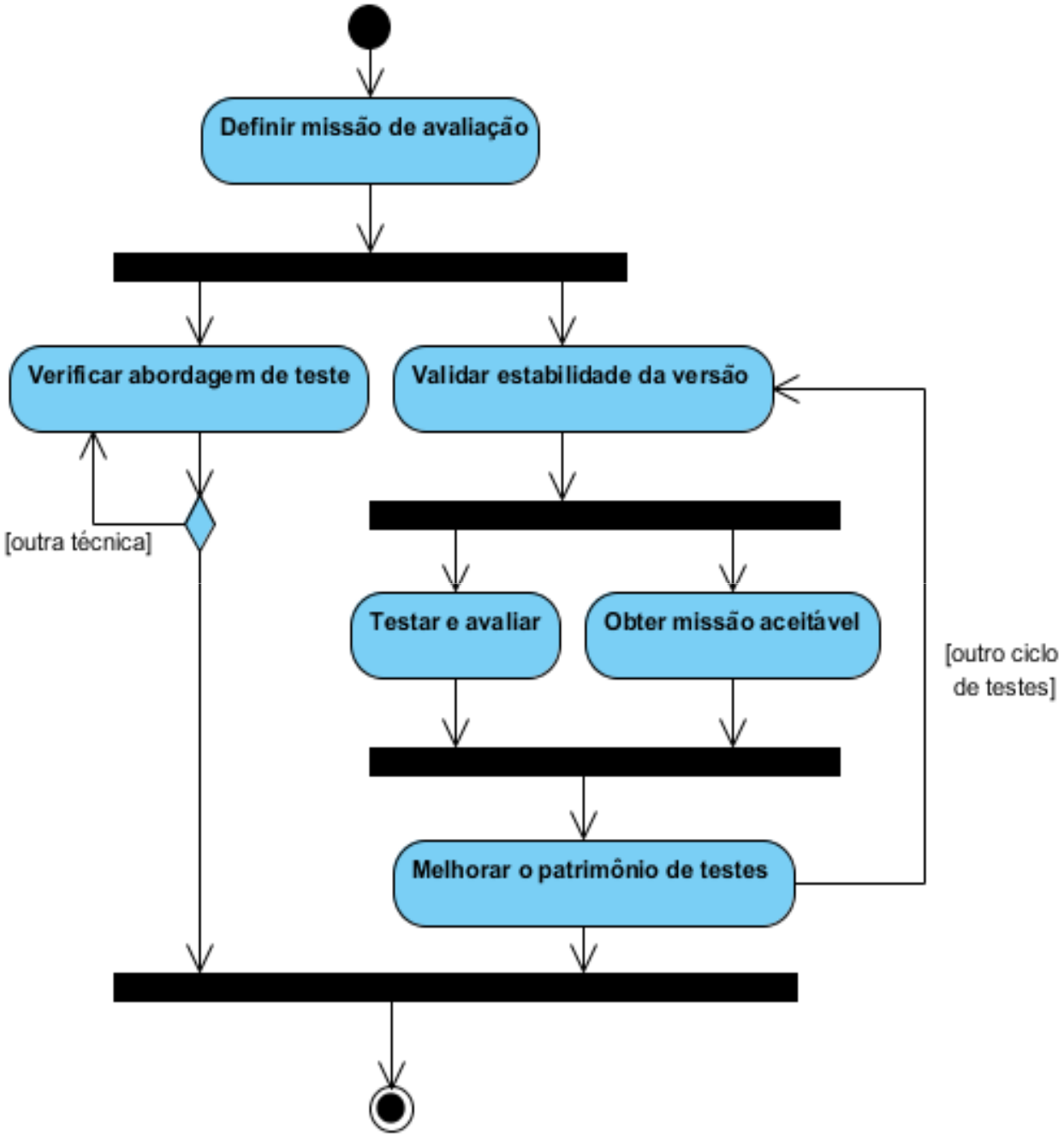
WORKFLOW DE IMPLEMENTAÇÃO



Teste

- A disciplina de teste no RUP exclui os testes de unidade, que são executados pelo programador na disciplina de implementação.
- O propósito da disciplina de testes é:
  - Verificar a interação entre objetos.
  - Verificar se todos os componentes foram integrados adequadamente.
  - Verificar se todos os requisitos foram corretamente implementados.
  - Verificar e garantir que defeitos tenham sido identificados e tratados antes da entrega do produto final.
  - Garantir que todos os defeitos tenham sido consertados, retestados e estancados.

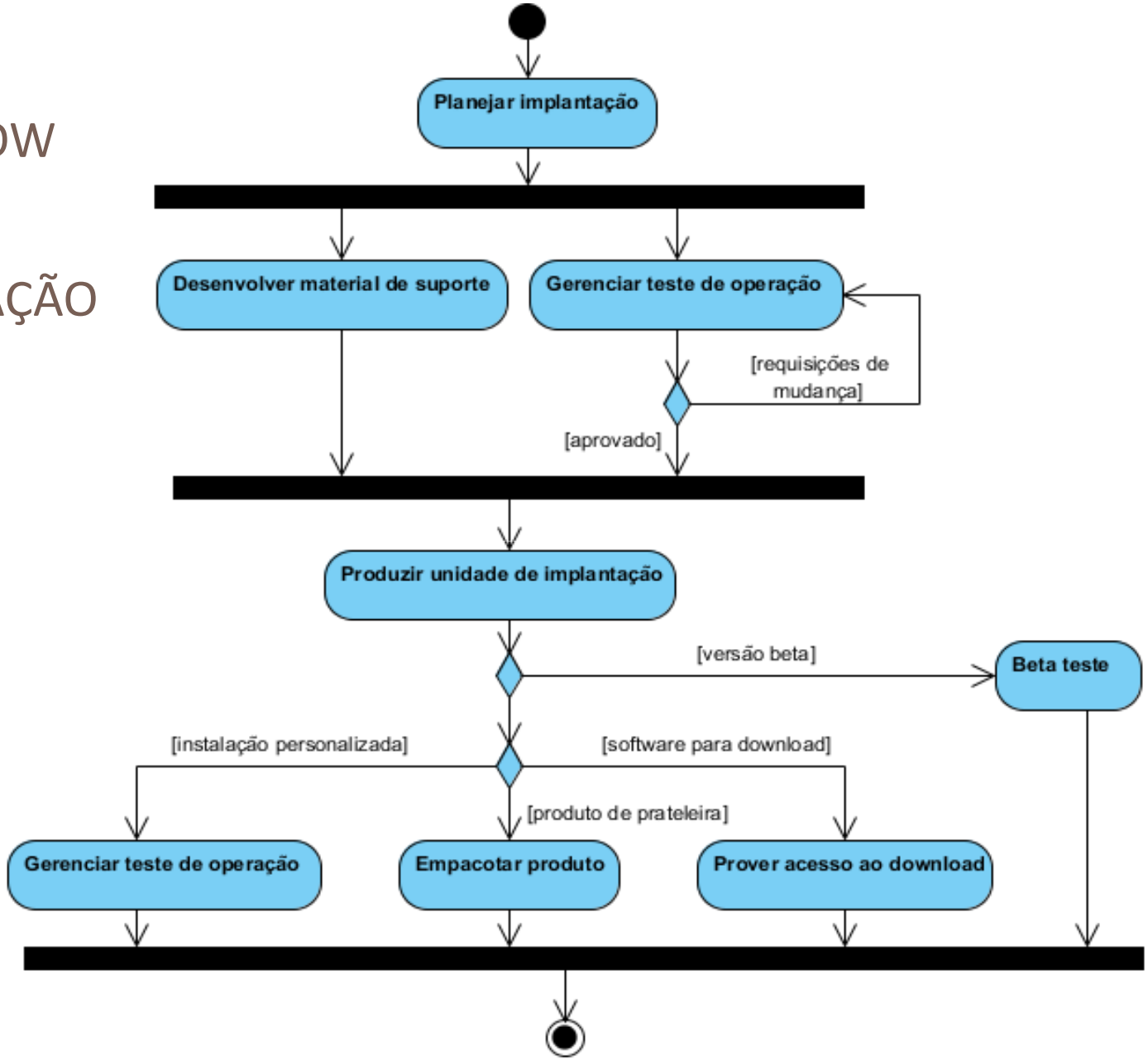
WORKFLOW TESTE



IMPLANTAÇÃO

- Existem vários tipos de implantação de software conforme a maneira como ele é distribuído, que vão desde sistemas personalizados a serem implantados diretamente nos computadores do cliente até software disponibilizado para *download* pela internet.
- A diferença entre os casos consiste no grau de envolvimento da empresa desenvolvedora do software com a instalação do produto no ambiente final.

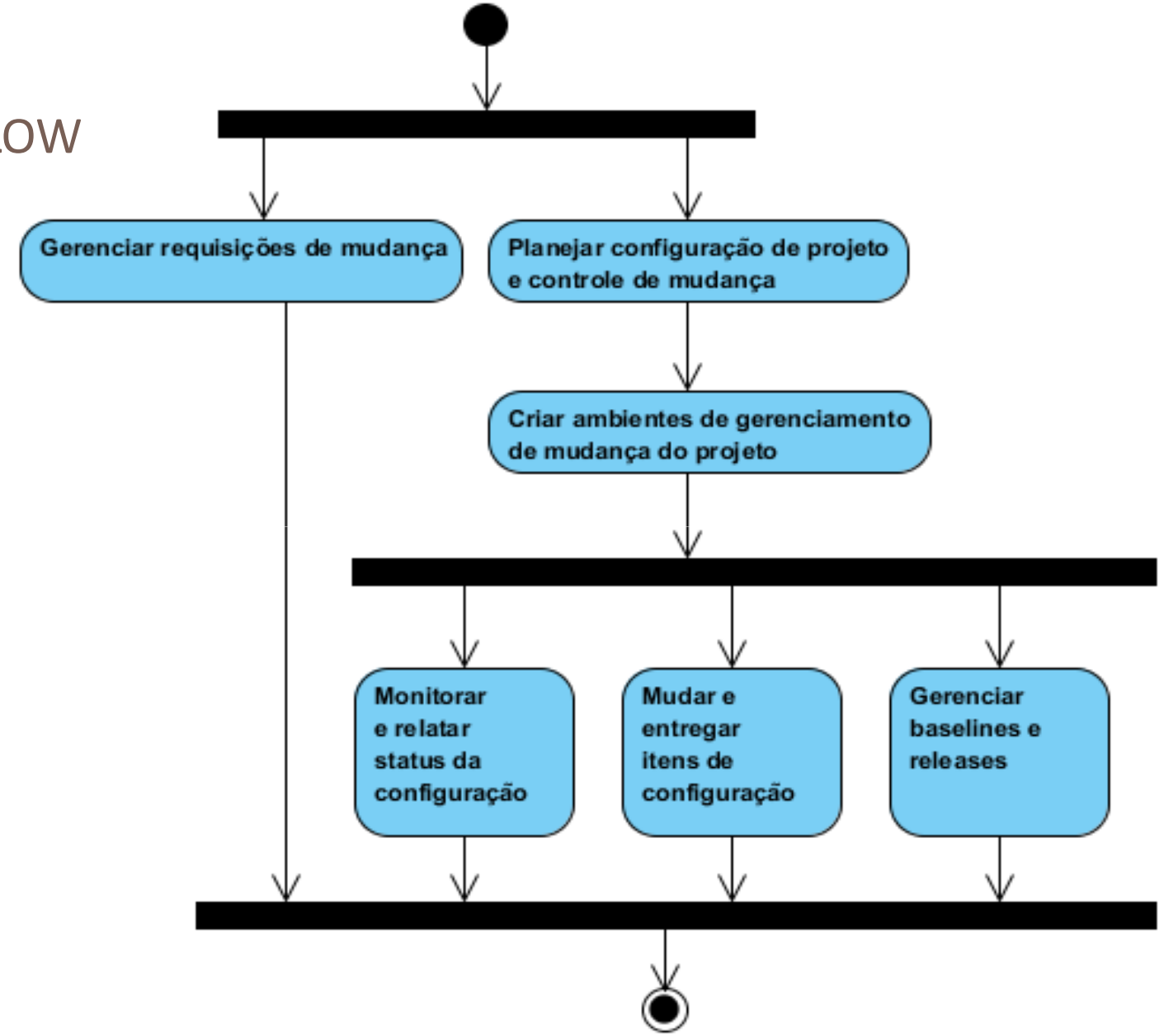
WORKFLOW DE IMPLANTAÇÃO



GERENCIAMENTO DE CONFIGURAÇÃO E MUDANÇA

- *Gerenciamento de configuração*.
  - Responsável pela estruturação sistemática dos produtos. Artefatos como documentos e diagramas devem estar sob controle de versão e mudanças feitas devem ser visíveis e localizáveis.
- *Gerenciamento de requisições de mudança*.
  - A área de CRM (*Change Request Management*) cuidará do controle das requisições de mudança em artefatos que produzem as diferentes versões destes.
- *Gerenciamento de status e medição*.
  - Requisições de mudança têm estados como: novo, atribuído, retido, concluído e entregue. Elas também podem ter atributos como causa, fonte, natureza, prioridade, etc. O gerenciamento de *status* coloca todos estes atributos em um sistema de informação tal que o andamento de cada solicitação seja verificável a todo momento pelo gerente do projeto.

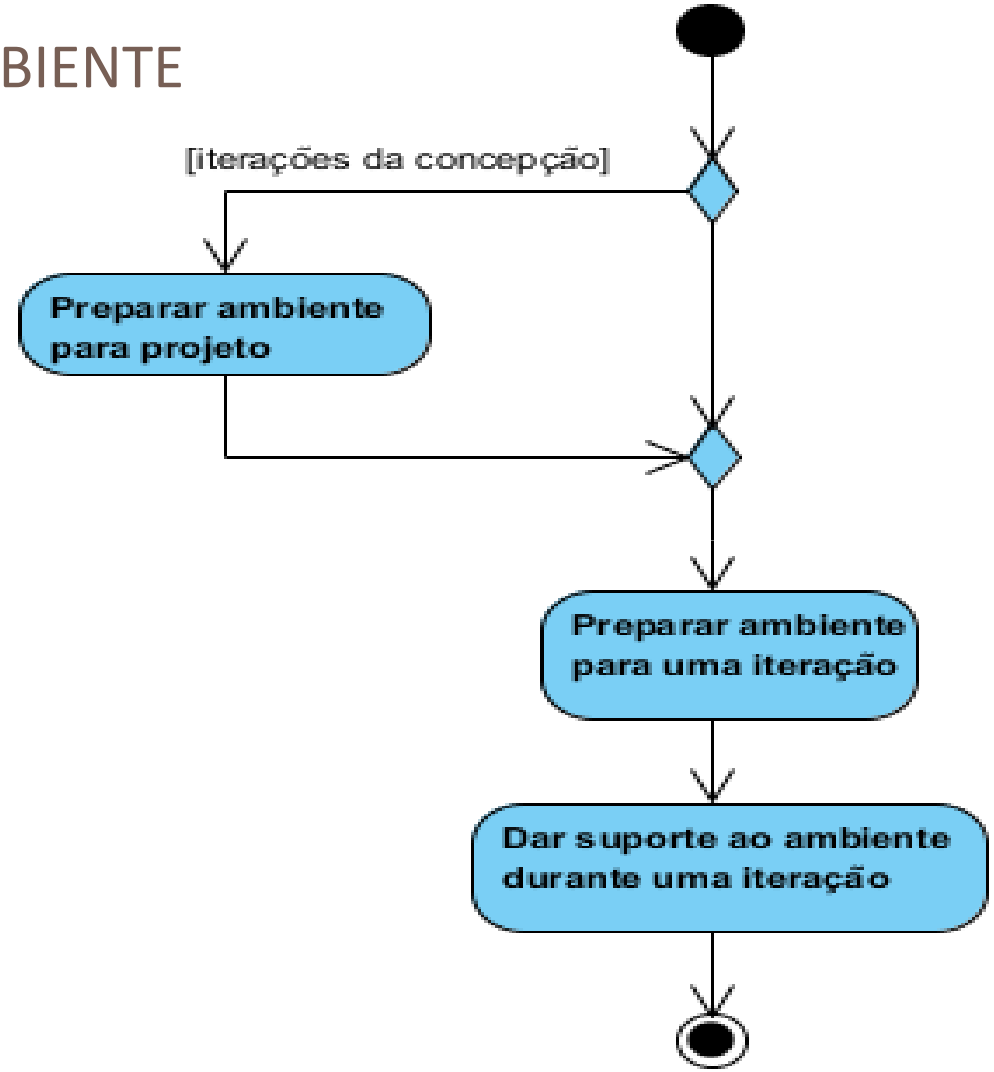
WORKFLOW


Gerenciar requisições de mudança
Planejar configuração de projeto e controle de mudança
Criar ambientes de gerenciamento de mudança do projeto
Monitorar e relatar status da configuração
Mudar e entregar itens de configuração
Gerenciar baselines e releases

Ambiente

- Trata principalmente da configuração do próprio processo a ser usado para desenvolver o projeto.
- Em função do processo específico escolhido essa disciplina deve também tratar das ferramentas de apoio necessárias para que a equipe tenha sucesso no projeto.

WORKFLOW DE AMBIENTE



AUP – Agile Unified Process

- É uma versão simplificada do RUP desenvolvida por Ambler.
- *A*plica técnicas ágeis como desenvolvimento dirigido por testes (TDD – *Test Driven Development*), modelagem ágil e refatoração.
- www.agilemodeling.com/style/

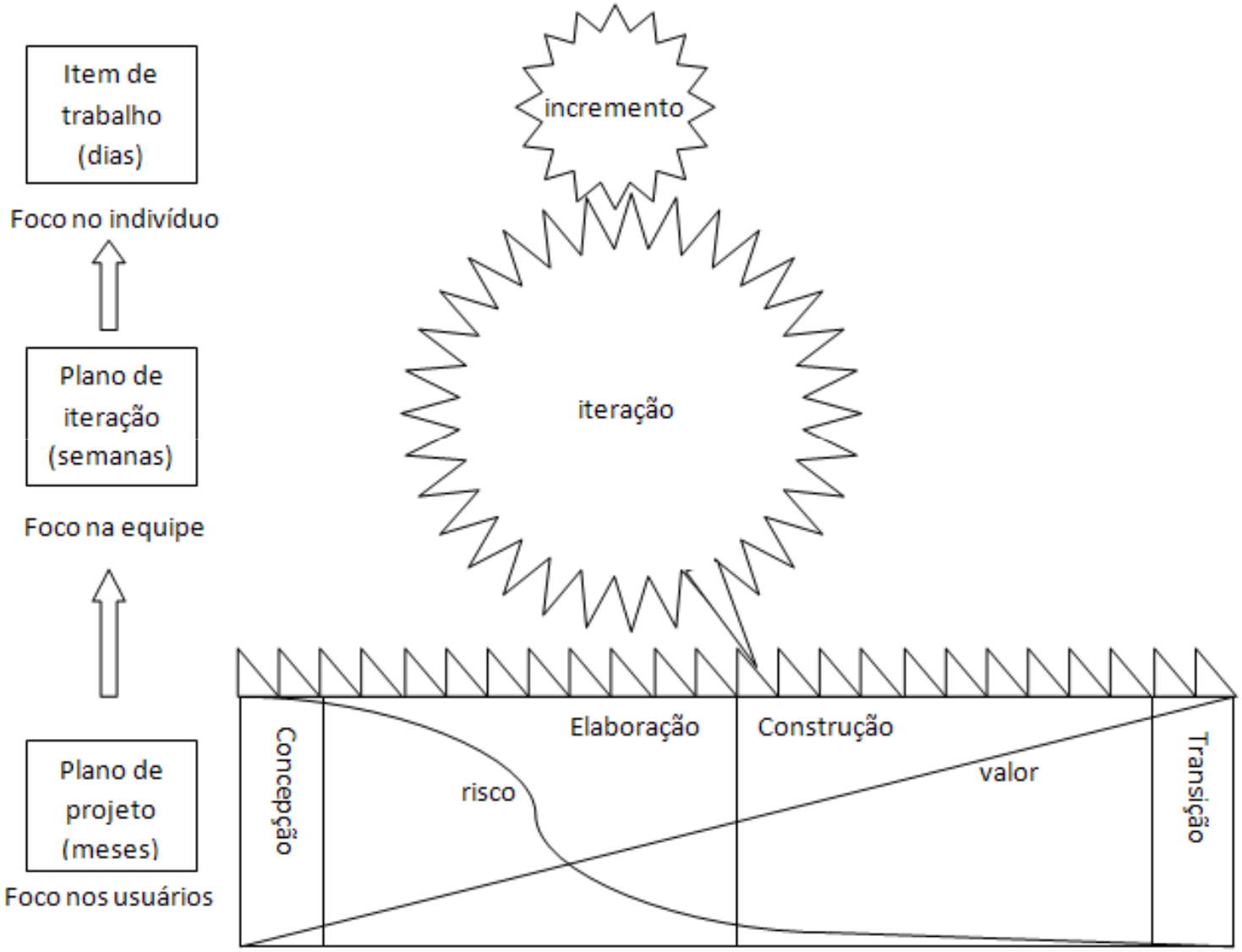
APENAS 7 DISCIPLINAS:

- *Modelagem*.
  - Entender o negócio da empresa através de modelos produzidos que buscam também identificar possíveis soluções para estes problemas. Um guia de modelagem AUP usando diagramas UML pode ser encontrado na página do AUP na Internet.
- *Implementação*.
  - Transformar os modelos em código executável e criar testes de unidade.
- *Teste*.
  - Efetuar testes de integração, de sistema e de aceitação para garantir que o sistema tenha qualidade e implemente os requisitos corretos corretamente.
- *Implantação*.
  - Planejar as entregas de sistema para tornar o sistema acessível para usuários.
- *Gerenciamento de configuração*.
  - Gerenciar as versões e o acesso aos artefatos do projeto.
- *Gerenciamento de projeto*.
  - Controlar as atividades de projeto, garantindo que atividades sejam atribuídas às pessoas certas e executadas no prazo.
- *Ambiente*.
  - Dar suporte à equipe garantindo que as ferramentas, guias e padrões estejam disponíveis quando necessário.

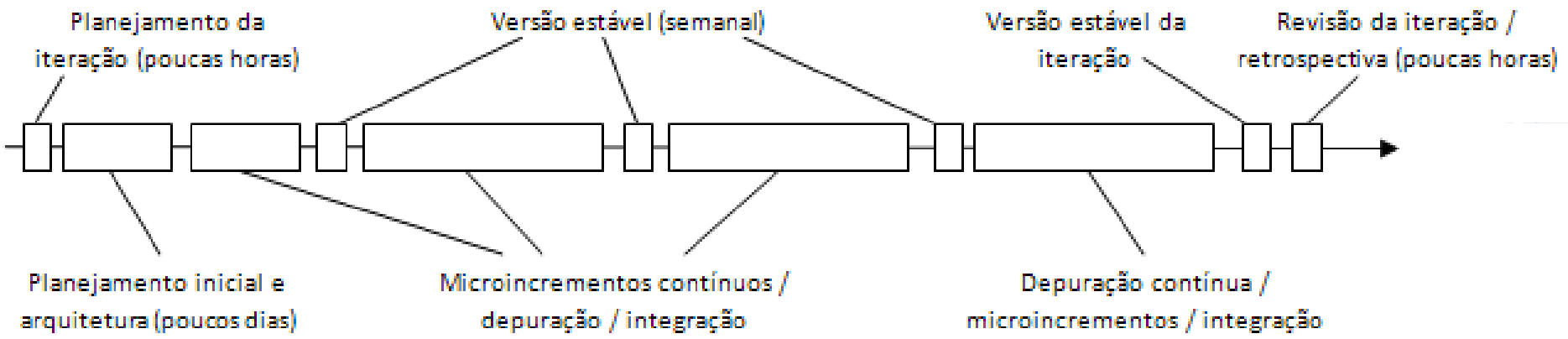
OpenUp – Open Unified Process

- É uma implementação aberta do *UP* desenvolvida como parte do *Eclipse Process Framework* (*EPF*).
- Anteriormente era conhecido como *Basic Unified Process* (*BUP*) ou *OpenUP/Basic*.
- *BUP* foi originada pela IBM que abriu a definição de uma versão mais leve do *RUP*.
- Entre 2005 e 2006 essa versão foi abraçada pela Fundação Eclipse e passou a ser um de seus projetos.
- Aceita, embora de forma simplificada, a maioria dos princípios *UP*.
  - Porém, é um método independente de ferramenta e de baixa cerimônia, ou seja, não é exigida grande precisão e detalhes nos documentos.
- Em português: epf.eclipse.org/wikis/openuppt/index.htm

Ciclo de vida OpenUP



CICLO DE VIDA DA ITERAÇÃO OPENUP



EUP – Enterprise Unified Process

- O *Enterprise Unified Process*, ou *EUP*, foi inicialmente definido por Ambler e Constantine em 1999 e posteriormente refinado por Ambler, Nalbone e Vizdos (2005).
- O modelo EUP vê o desenvolvimento de software não apenas como um projeto a ser executado, mas como algo intrínseco ao ciclo de vida da própria empresa.
- EUP foi proposto como uma extensão ao modelo RUP para prover, além das fases de RUP duas novas fases para tratar a evolução ou suporte ao sistema e a aposentadoria do sistema.
- Além destas duas fases, várias novas disciplinas relacionadas à empresa foram adicionadas.

NOVAS FASES EUP

- *Produção*.
  - O desenvolvimento de sistemas usualmente não acaba quando o produto é entregue e colocado em uso.
  - A fase de produção trata exatamente das atividades que ocorrem após a transição, incluindo o suporte, correção e ajustes e evolução do sistema.
- *Aposentadoria*.
  - A fase de aposentadoria consiste na retirada de um sistema de operação.
  - É a fase final de qualquer sistema.
  - Sistemas antigos retirados de operação para serem substituídos por sistemas novos podem causar sérios danos à empresa se o processo não for gerenciado adequadamente.

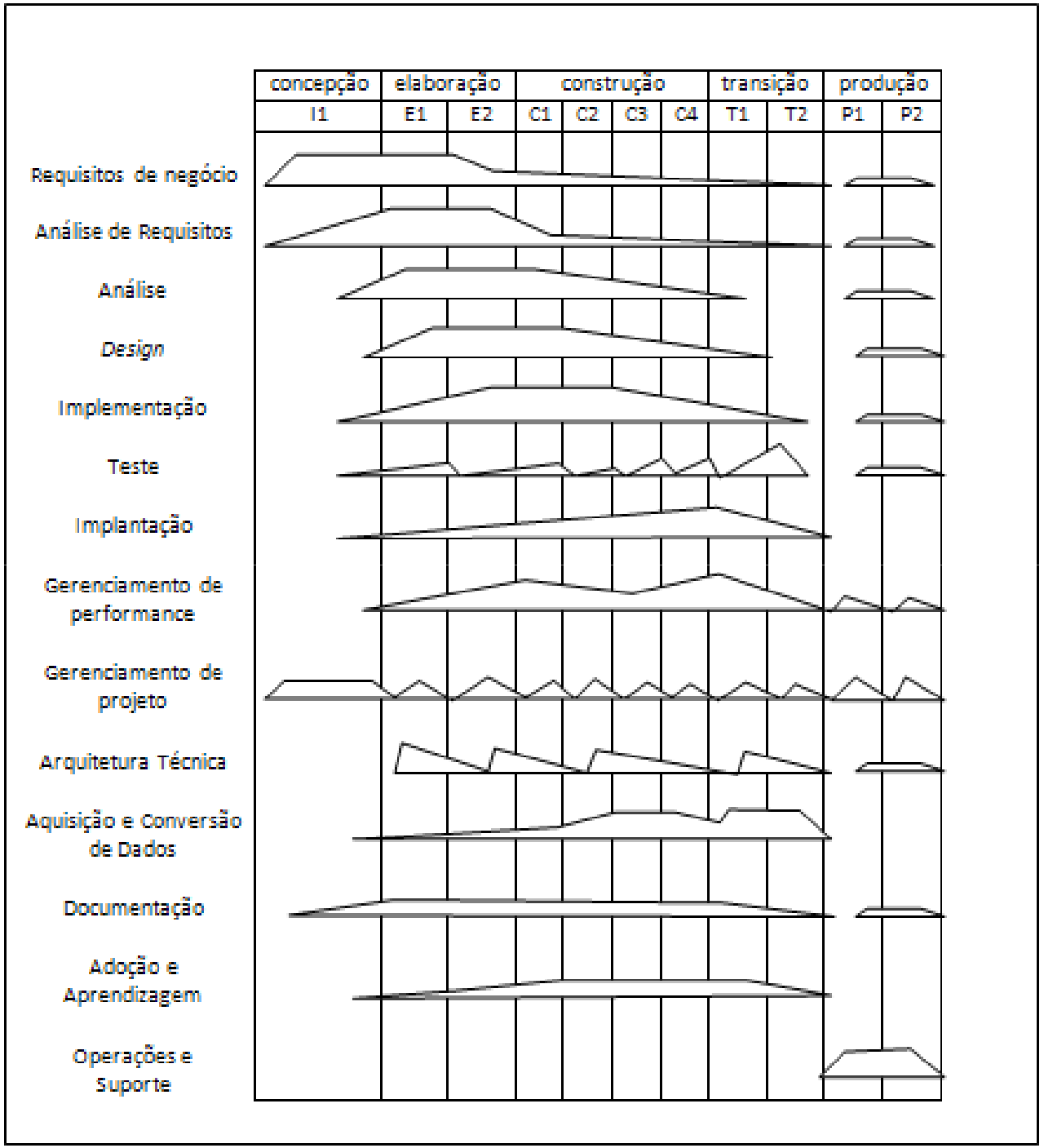
NOVAS DISCIPLINAS EUP

- *Modelagem de negócio de empresa*.
  - O RUP já apresenta uma disciplina de modelagem de negócio, mas do ponto de vista do sistema a ser desenvolvido.
  - A modelagem de negócio de empresa do EUP é mais abrangente, incluindo todos os processos da empresa e, desta forma, relações entre diferentes sistemas.
- *Gerenciamento de portfólio*.
  - Um portfólio é uma coleção e projetos de software em andamento e concluídos.
- *Arquitetura de empresa*.
  - A arquitetura de empresa define como ela trabalha.
  - Essa disciplina é especialmente útil se a empresa possuiu muitos produtos de software.
  - Deve haver consistência na forma como eles são desenvolvidos, negociados e entregues.
  - A arquitetura de empresa é a chave para compreender isso como um processo.
- *Reuso estratégico*.
  - O reuso estratégico vai além do reuso que se consegue dentro de um único projeto.
  - Ele se estende entre diferentes projetos.
- *Gerenciamento de pessoas*.
  - Esta disciplina define uma abordagem para gerenciamento de recursos humanos da área de Tecnologia de Informação.
  - É preciso gerenciar o pessoal, contratar, demitir, substituir, alocar pessoas a projetos e investir em seu crescimento.
- *Administração de empresa*.
  - Esta disciplina define como a empresa cria, mantém, gerencia, e entrega produtos físicos e informações de forma segura.
- *Melhoria de processo de software*.
  - Esta disciplina trata da adequação e evolução do processo de software para a empresa como um todo, não apenas a adequação do processo a cada projeto individual.

OUM – Oracle Unified Method

- É um *framework* de processo de desenvolvimento de software iterativo e incremental adequado a uso com produtos Oracle: bancos de dados, aplicações e *middleware*.
- www.oracle.com/consulting/library/briefs/oracle-unified-method.pdf

CICLO DE VIDA OUM



RUP-SE

Rational Unified Process–Systems Engineering

- É uma extensão do modelo RUP para Engenharia de Sistemas (Cantor, 2003).
- É uma versão de RUP especialmente adequada para desenvolvimento de sistemas de grande porte, envolvendo, software, hardware, pessoas e componentes de informação.
- www.ibm.com/developerworks/rational/library/content/RationalEdge/aug03/f_rupse_mc.pdf

PROJETOS PARA OS QUAIS RUP-SE É ADEQUADO

- São grandes o suficiente para comportar várias equipes de desenvolvimento trabalhando em paralelo.
- Necessitam desenvolvimento concorrente de hardware e software.
- A arquitetura é impactada por questões relativas à implantação.
- Incluem a reengenharia de uma infraestrutura de tecnologia informação para suportar a evolução do negócio.