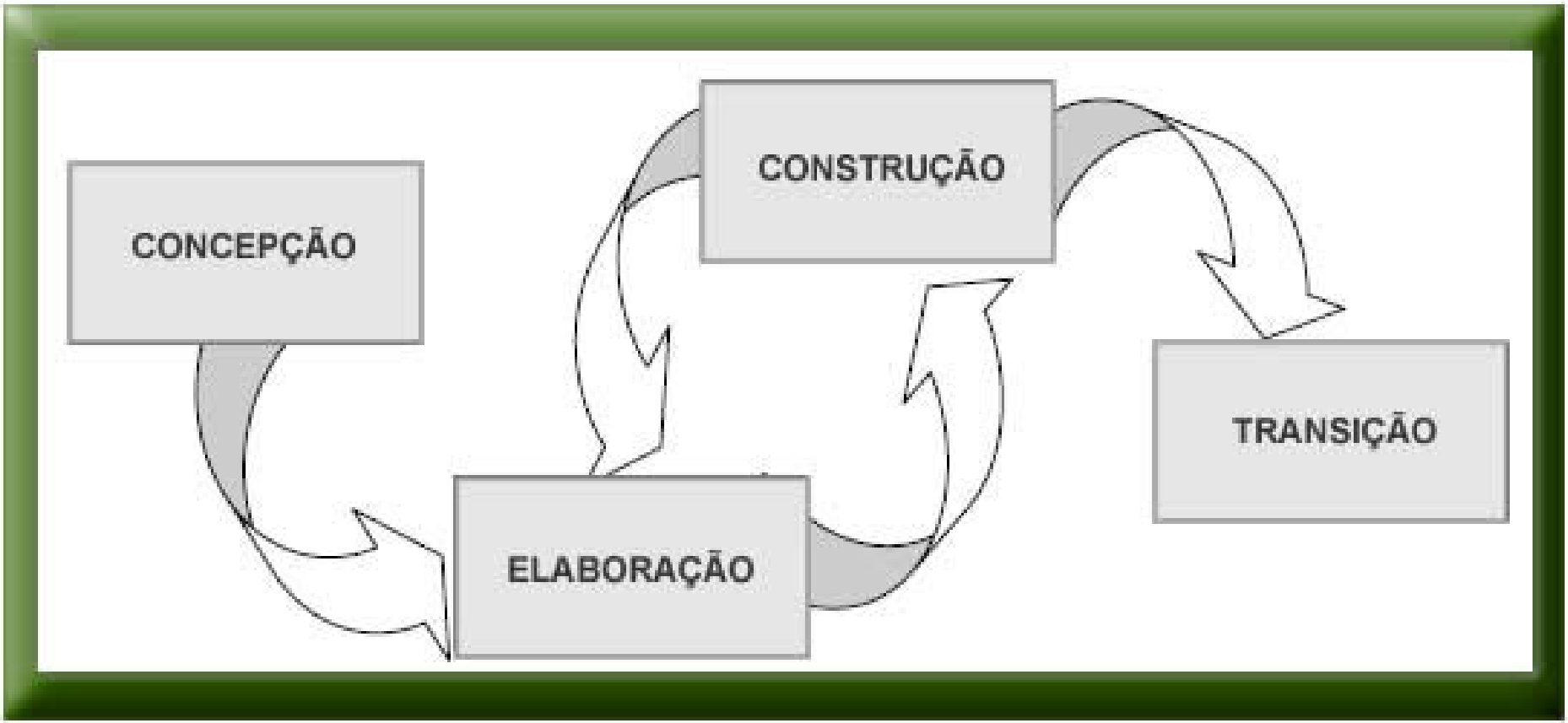
RUP

SCRUM

XP

FDD

O **RUP (Rational Unified Process)** ou Processo Unificado Rational, é um processo proprietário de Engenharia de software criado pela Rational Software Corporation, adquirida pela IBM, ganhando um novo nome **IRUP,** abreviação de **IBM Rational Unified Process** e tornando-se uma *brand* na área de Software, fornecendo técnicas a serem seguidas pelos membros da equipe de desenvolvimento de software com o objetivo de aumentar a sua produtividade no processo de desenvolvimento.



Iniciação ou Concepção: ênfase no escopo do sistema;
Elaboração: ênfase na arquitetura;
Construção: ênfase no desenvolvimento;
Transição: ênfase na implantação.

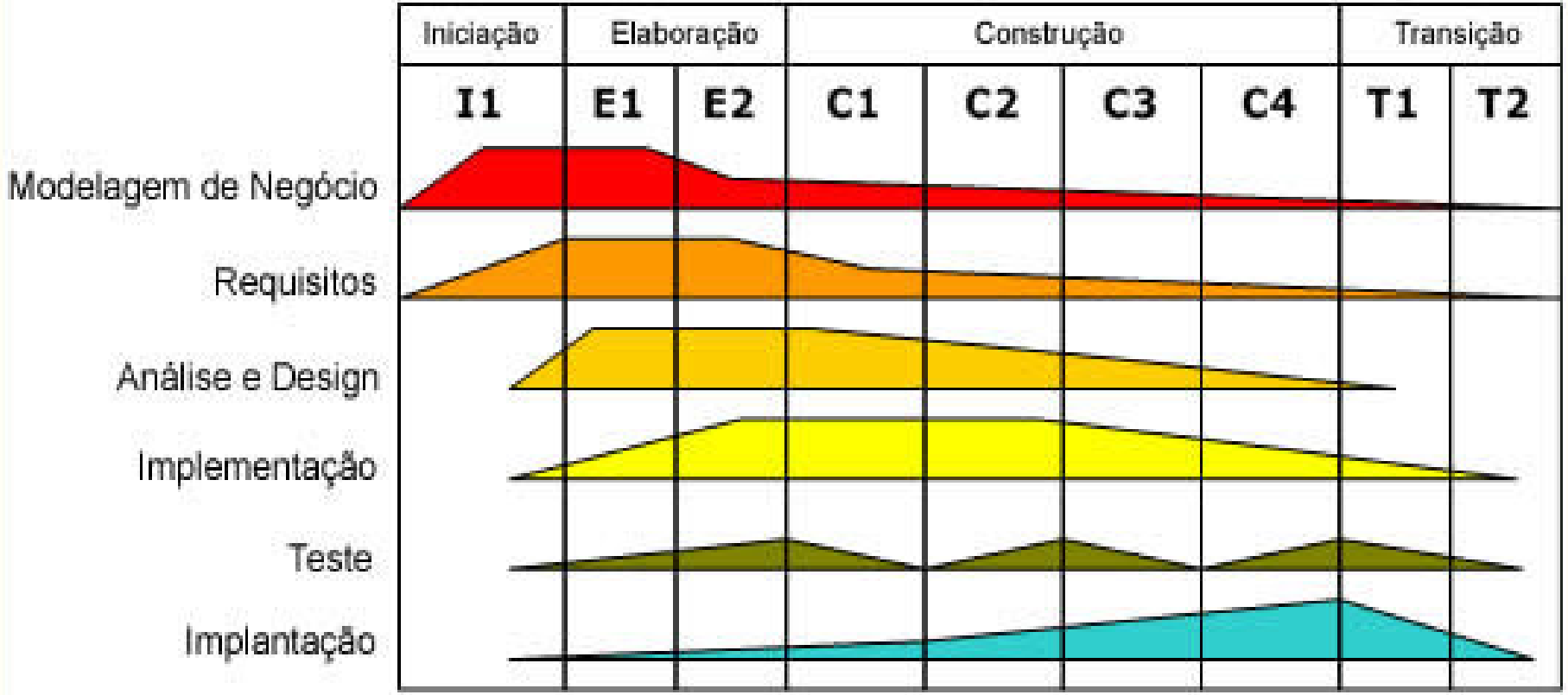
RUP - processo de engenharia de software que fornece uma abordagem disciplinada para assumir tarefas e responsabilidades.

Objetivo do RUP: assegurar a produção de software de alta qualidade dentro de prazos e orçamentos previsíveis.

Derivado dos trabalhos sobre UML, ele traz elementos de todos os modelos genéricos de processo, apoia a iteração e ilustra boas práticas de especificação e projeto.

O RUP usa a abordagem da orientação a objetos em sua concepção e é projetado e documentado utilizando a notaçãoUML (*Unified Modeling Language*) para ilustrar os processos em ação. Utiliza técnicas e práticas aprovadas comercialmente.

FASES DO RUP


Iniciação
Elaboração
Construção
Transição
I1
E1
E2
C1
C2
C3
C4
T1
T2
Modelagem de Negócio
Requisitos
Análise e Design
Implementação
Teste
Implantação

Fase de Concepção / Iniciação: Esta fase do RUP abrange as tarefas de comunicação com o cliente e planejamento. É feito um plano de projeto avaliando os possíveis riscos, as estimativas de custo e prazos, estabelecendo as prioridades, levantamento dos requisitos do sistema e preliminarmente analisá-lo. Assim, haverá uma anuência das partes interessadas na definição do escopo do projeto, onde são examinados os objetivos para se decidir sobre a continuidade do desenvolvimento.

Fase de Elaboração: Abrange a Modelagem do modelo genérico do processo.

O objetivo desta fase é analisar de forma mais detalhada a análise do domínio do problema, revisando os riscos que o projeto pode sofrer e a arquitetura do projeto começa a ter sua forma básica. Indagações como

- "O plano do projeto é confiável?“
- "Os custos são admissíveis?" são esclarecidas nesta etapa.

Fase de Construção: Desenvolve ou adquire os componentes de Software.
O principal objetivo desta fase é a construção do sistema de software, com foco no desenvolvimento de componentes e outros recursos do sistema.
É na fase de Construção que a maior parte de codificação ocorre.

Fase de Transição: Abrange a entrega do software ao usuário e a fase de testes. O objetivo desta fase é disponibilizar o sistema, tornando-o disponível e compreendido pelo usuário final. As atividades desta fase incluem o treinamento dos usuários finais e também a realização de testes da versão beta do sistema visando garantir que o mesmo possua o nível adequado de qualidade.

Métodos concorrentes no campo da engenharia de software incluem o:

- "Cleanroom" (considerado pesado)
- Métodos Ágeis (leves) como a Programação Extrema (XP-Extreme Programming), Scrum, FDD e outros.

XP

eXtreme Programming, ou **XP**, é uma metodologia ágil para equipes pequenas e médias e que irão desenvolver software com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software. Os princípios básicos da metodologia **XP** são *feedback* rápido, simplicidade, mudanças incrementais e trabalho de qualidade. Dentre as variáveis de controle em projetos (custo, tempo, qualidade e escopo), há um foco explícito em escopo.

É necessário priorizar funcionalidades que representem maior valor possível para o negócio. Desta forma, caso seja necessário a diminuição de escopo, as funcionalidades menos valiosas serão adiadas ou canceladas.

A **XP** incentiva o controle da qualidade como variável do projeto, pois o pequeno ganho de curto prazo na produtividade, ao diminuir qualidade, não é compensado por perdas a médio e longo prazo.

Os 5 valores fundamentais da metodologia **XP** são: comunicação, simplicidade, *feedback*, coragem e respeito e os princípios básicos são: *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de qualidade.

Dentre as variáveis de controle em projetos (custo, tempo, qualidade e escopo), há um foco explícito em escopo. Assim, priorizamos as funcionalidades que representem maior valor possível para o negócio porque se for necessário diminuir o escopo, as funcionalidades menos valiosas serão adiadas ou canceladas.

A **XP** incentiva o controle da qualidade como variável do projeto, pois o pequeno ganho de curto prazo na produtividade, ao diminuir qualidade, não é compensado por perdas a médio e longo prazo.

SCRUM

SCRUM

Inicialmente, o Scrum foi concebido como um estilo de gerenciamento de projetos em empresas de fabricação de automóveis e produtos de consumo, por Takeuchi e Nonaka. Eles notaram que projetos usando equipes pequenas e multidisciplinares (*cross-functional*) produziram os melhores resultados, e associaram estas equipes altamente eficazes à formação Scrum do Rugby (utilizada para reinício do jogo em certos casos).

Scrum não é um processo prescribente, ou seja, ele não descreve o que fazer em cada situação. Ele é usado para trabalhos complexos nos quais é impossível predizer tudo o que irá ocorrer.

Apesar de Scrum ter sido destinado para gerenciamento de projetos de software, ele pode ser utilizado em equipes de manutenção de software ou como uma abordagem geral de gerenciamento de projetos/programas.

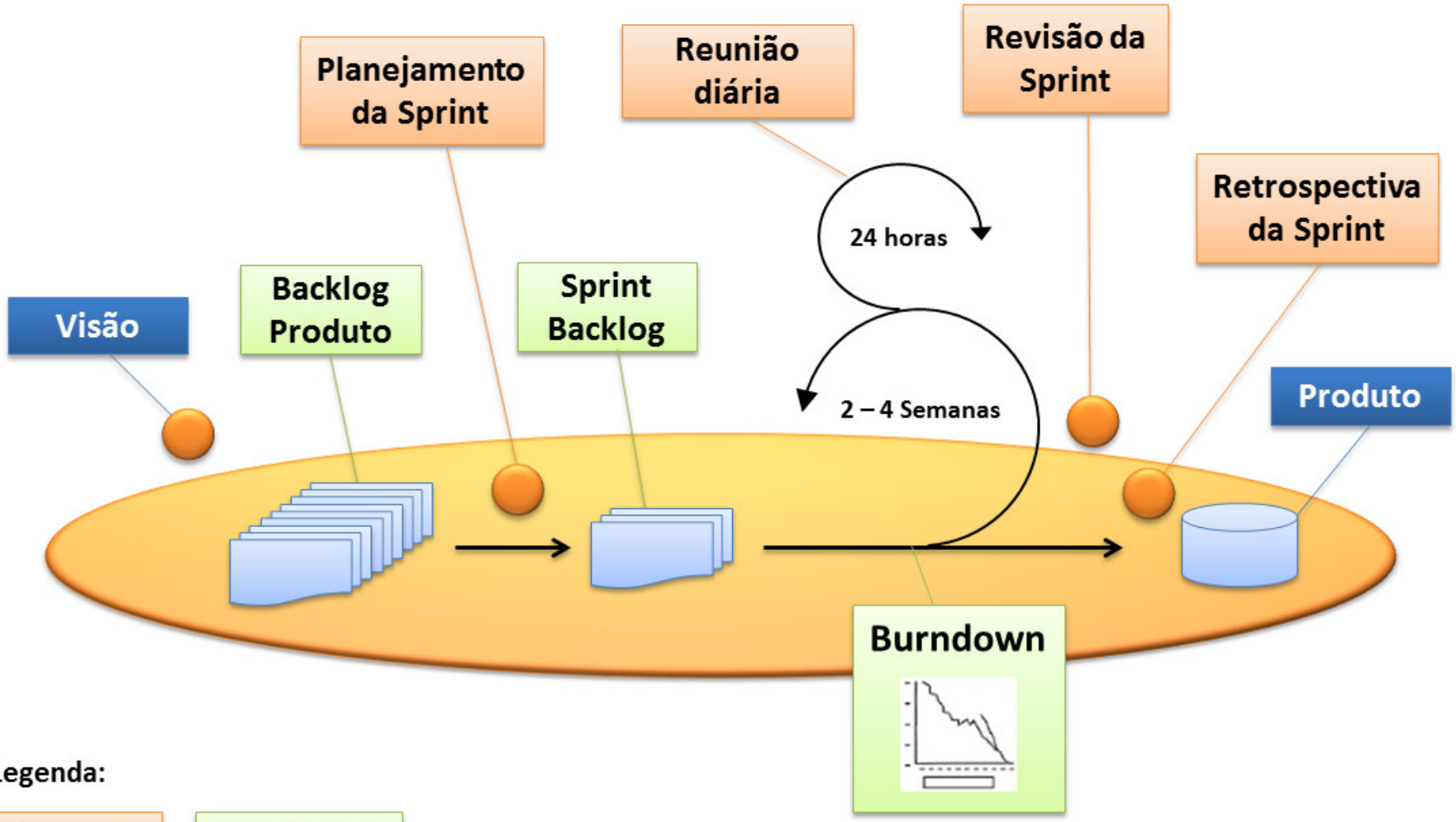
O **Scrum** é um processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software.
Scrum possui foco no gerenciamento de projeto da organização onde é difícil planejar à frente.
Mecanismos do Controle de Processo, onde ciclos de feedback constituem o núcleo da técnica de gerenciamento que são usadas em oposição ao tradicional gerenciamento de comando e controle. É uma forma de planejar e gerenciar projetos trazendo a autoridade da tomada de decisão a níveis de propriedade de operação e certeza.

Scrum é um esqueleto de processos que contém grupos de práticas e papéis pré-definidos. Os principais papéis são:

- o **ScrumMaster**, que mantém os processos (normalmente no lugar de um gerente de projeto);
- o **Proprietário do Produto**, ou **Product Owner**, que representa os *stakeholders* e o negócio;
- a **Equipe**, ou **Team**, um grupo multifuncional com cerca de 7 pessoas e que fazem a análise, projeto, implementação, teste etc.

Sprint (corrida, tiro)

Um sprint é a unidade básica de desenvolvimento em Scrum. Sprints tendem a durar entre uma semana e um mês, e são um esforço dentro de uma caixa de tempo (ou seja, restrito a uma duração específica) de comprimento constante. Cada sprint é precedido por uma reunião de planejamento, onde as tarefas para o sprint são identificadas e um compromisso estimado para o objetivo do sprint é definido e seguido por uma reunião de revisão ou de retrospectiva, onde o progresso é revisto e lições para os próximos sprints são identificadas.



Legenda:

Cerimônia | Artefatos

Papéis

- Product Owner (PO)
- Scrum Master (SM)
- Equipe Scrum

Cerimônias

- Planejamento da Sprint
- Reunião diária
- Revisão da Sprint
- Retrospectiva da Sprint

Artefatos

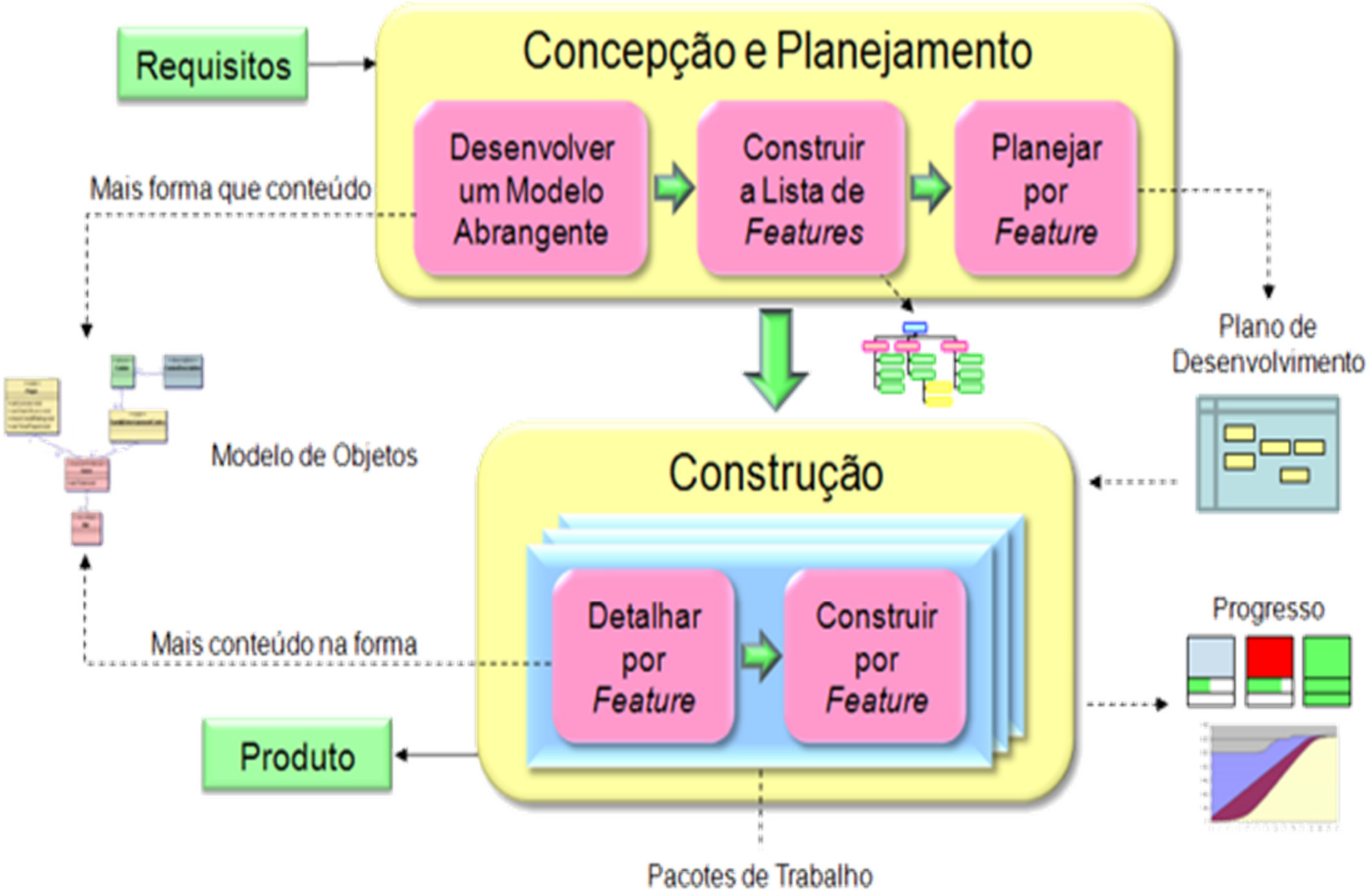
- Product Backlog
- Sprint Backlog
- Burndown (gráfico)

FDD

O **Desenvolvimento Guiado por Funcionalidades** (do inglês, *Feature Driven Development*; FDD) é uma das seis metodologias ágeis originais do desenvolvimento de software.

A FDD é, classicamente, descrita por cinco processos:

Desenvolver um Modelo Abrangente
Construir Lista de Funcionalidades
Planejar por Funcionalidade
Detalhar por Funcionalidade
Construir por Funcionalidade


Requisitos
Concepção e Planejamento
Desenvolver um Modelo Abrangente
Construir a Lista de Features
Planejar por Feature
Mais forma que conteúdo
Plano de Desenvolvimento
Modelo de Objetos
Construção
Detalhar por Feature
Construir por Feature
Progresso
Mais conteúdo na forma
Produto
Pacotes de Trabalho

Desenvolver um Modelo Abrangente:
pode envolver desenvolvimento de requisitos, análise orientada por objetos, modelagem lógica de dados e outras técnicas para entendimento do domínio de negócio em questão. O resultado é um modelo de objetos (e/ou de dados) de alto nível, que guiará a equipe durante os ciclos de construção.

Construir uma Lista de Funcionalidades: decomposição funcional do modelo do domínio, em três camadas típicas: áreas de negócio, atividades de negócio e passos automatizados da atividade (funcionalidades). O resultado é uma hierarquia de funcionalidades que representa o produto a ser construído (também chamado de *product backlog*, ou lista de espera do produto).

Planejar por Funcionalidade: abrange a estimativa de complexidade e dependência das funcionalidades, também levando em consideração a prioridade e valor para o negócio/cliente. O resultado é um plano de desenvolvimento, com os pacotes de trabalho na seqüência apropriada para a construção.

Detalhar por Funcionalidade:
já dentro de uma iteração de construção, a equipe detalha os requisitos e outros artefatos para a codificação de cada funcionalidade, incluindo os testes. O projeto para as funcionalidades é inspecionado. O resultado é o modelo de domínio mais detalhado e os esqueletos de código prontos para serem preenchidos.

Construir por Funcionalidade: cada esqueleto de código é preenchido, testado e inspecionado. O resultado é um incremento do produto integrado ao repositório principal de código, com qualidade e potencial para ser usado pelo cliente/usuário.