Desenvolvimento de Sistemas Software

Aula Teórica 2: Métodos de Desenvolvimentos

Desenvolvimento de Sistemas Software
António Nestor Ribeiro, José Creissac Campos, F. Mário Martins

Resumo da aula anterior…

- A tecnologia só por si não resolve os problemas (e estes têm vindo a aumentar!)
- A tecnologia é apenas uma ferramenta, é necessário saber como utilizá-la
- Hoje em dia a *Crise do Software* tem muito a ver com a qualidade do produto final.
- São necessários métodos de desenvolvimento que garantam produtividade e qualidade.

método

do Lat. methodu < Gr. méthodos, caminho para chegar a um fim

s. m., processo racional que se segue para chegar a um fim; modo ordenado de proceder; processo; ordem; conjunto de procedimentos técnicos e científicos;

(http://www.priberam.pt/dlpo/)

O desenvolvimento de software não pode ser encarado como *arte*,
mas como Engenharia.
Necessitamos de métodos e ferramentas apropriados.

Abordagem de Engenharia ao Desenvolvimento de Sistemas Software

Compreendida a missão (desenvolver Sistemas Software úteis e eficazes usando procedimentos de Engenharia), compreendido o enquadramento (para as organizações, sejam de comércio, serviços, ou indústria), e conhecendo até "alguma história" de insucesso, o que precisamos de saber e aprender para que se possa inverter tal história e, de facto, deixar a arte e enveredar pela engenharia e, assim, por um maior rigor ?

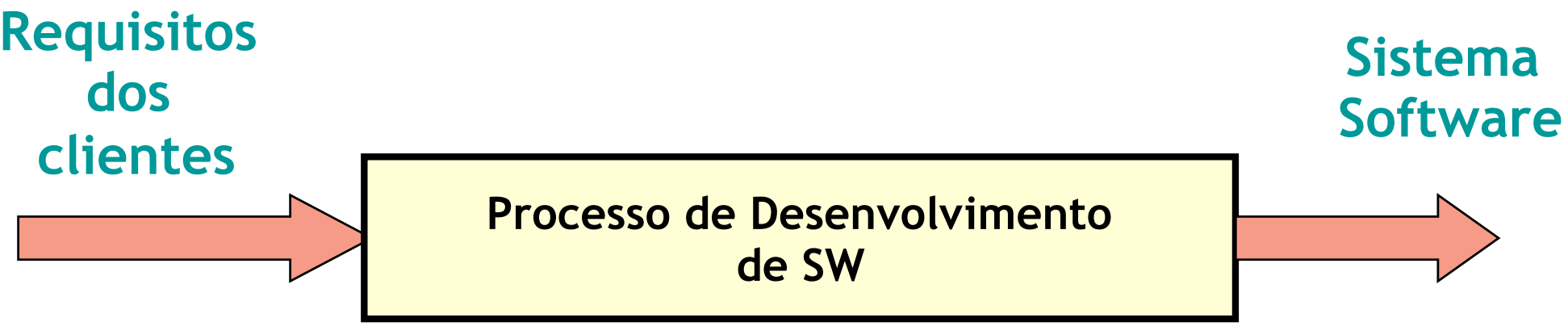
1.- Adoptar um Processo de desenvolvimento de Sistemas Software que nos garanta tais metas, em especial depois de conhecermos a história dos processos desenvolvidos; O processo adoptado deverá dar, justificadamente, muitas mais garantias de sucesso no desenvol-vimento dos actuais e futuros SI (ou Sistemas Software);

2.- Adoptar/Criar Métodos, ou seja, regras sobre "como fazer", desde como fazer a captura de requisitos até como fazer a instalação, os testes e a manutenção;

3.- Adoptar Ferramentas que representam o suporte automático ou semi-automático aos processos e aos métodos.

Processo de Desenvolvimento de Software

Um Processo de Desenvolvimento de Software consiste de uma **estruturação das várias disciplinas ou fases** que estão contidas na filosofia de desenvolvimento de software adoptada por uma dada organização para o desenvolvimento do produto sistema software.
Mas, fundamentalmente, consiste em definir **QUEM** no projecto está a fazer **O QUÊ**, **QUANDO** o deve fazer e **DURANTE** quanto tempo, e como se devem atingir os objectivos definidos.

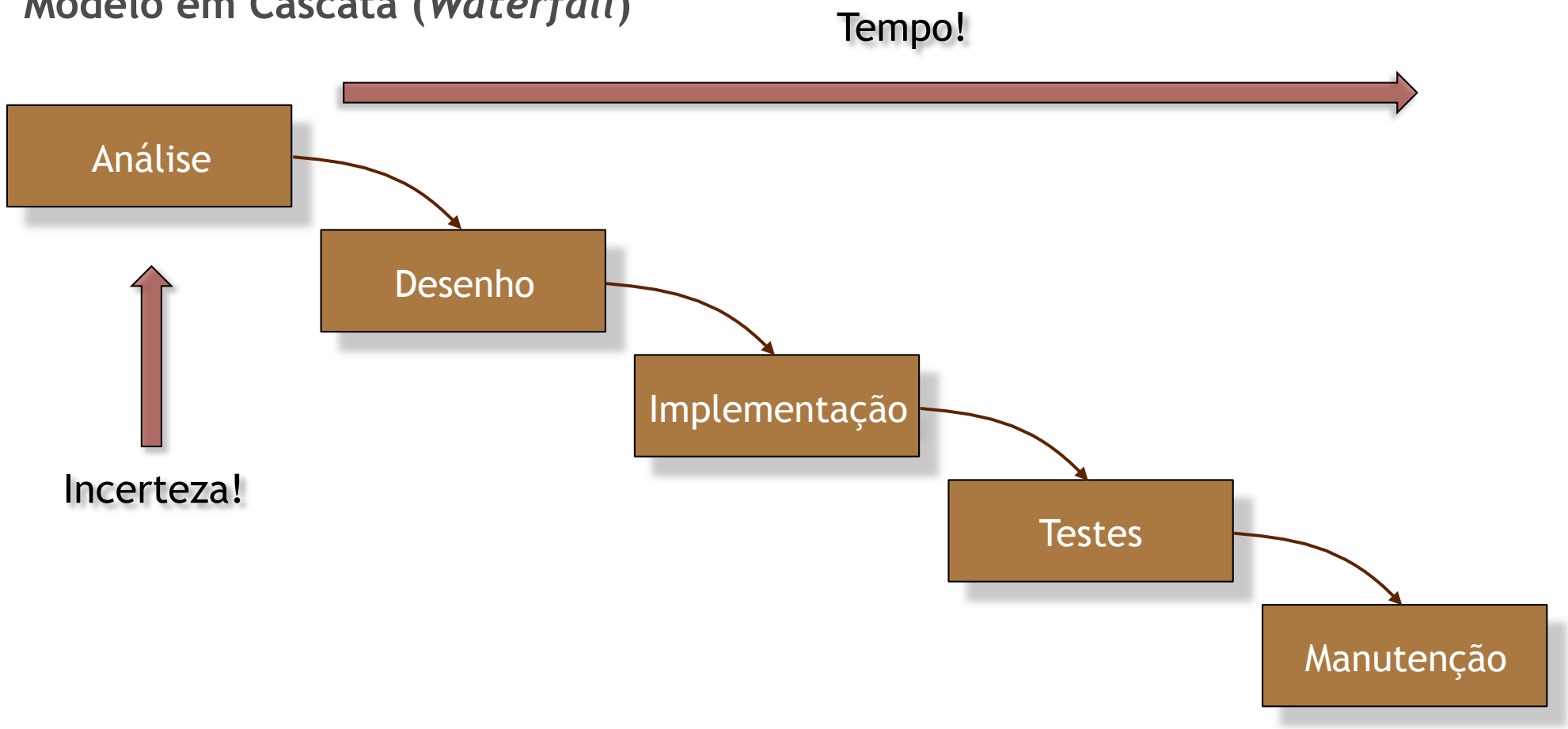


Etapas típicas do Processo de Desenvolvimento de Software

- Analisar o problema
  - Definir o *quê*
- Conceber a solução
  - Definir o *como*
- Implementar a solução
  - Fazer
- Testar a solução
  - Verificar e validar

Métodos de Desenvolvimento Sequênciais

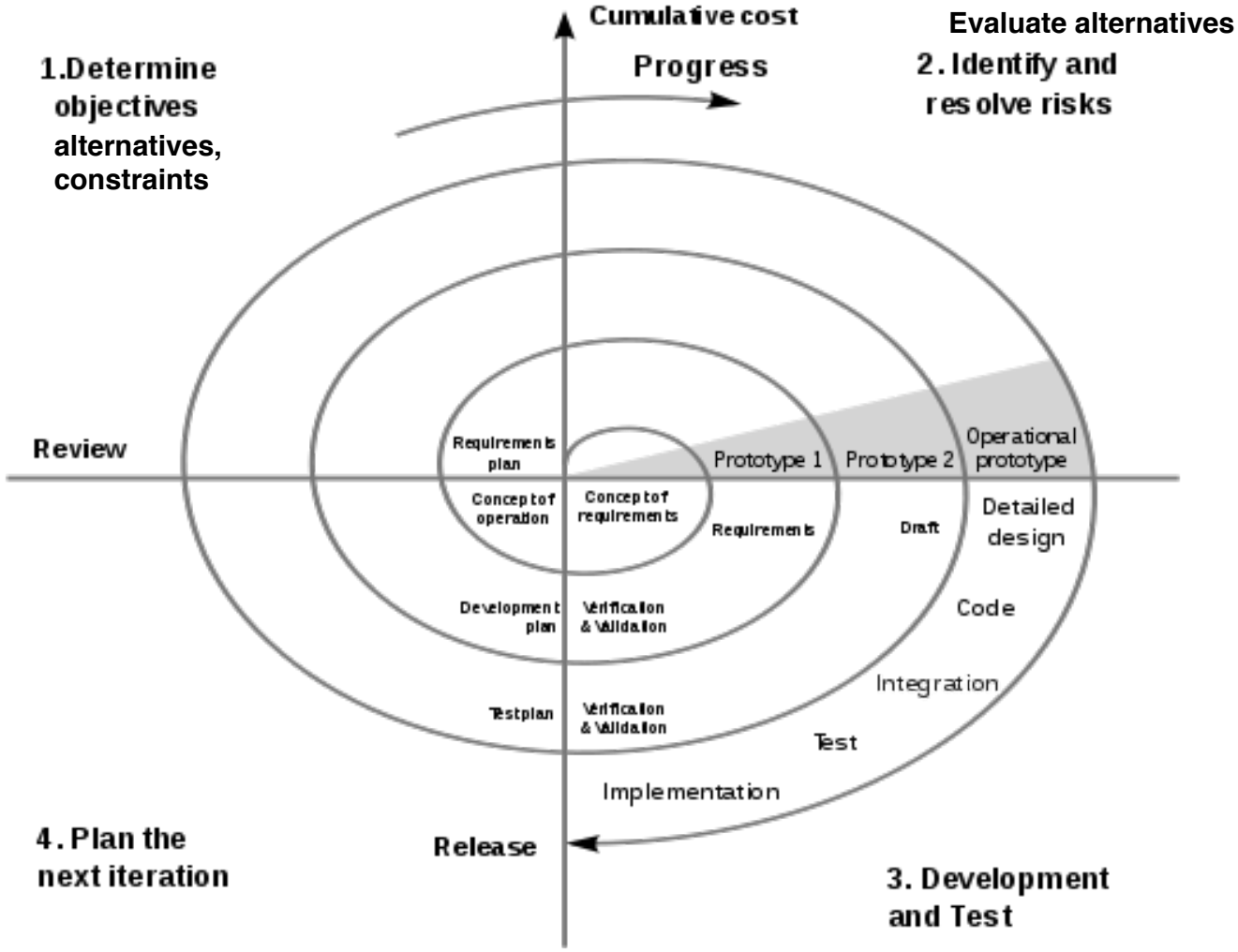
Modelo em Cascata (*Waterfall*)



- Este tipo de processo define uma série de etapas executadas sequencialmente.

Métodos de Desenvolvimento Iterativos

Modelo em Espiral (Boehm, 1988)



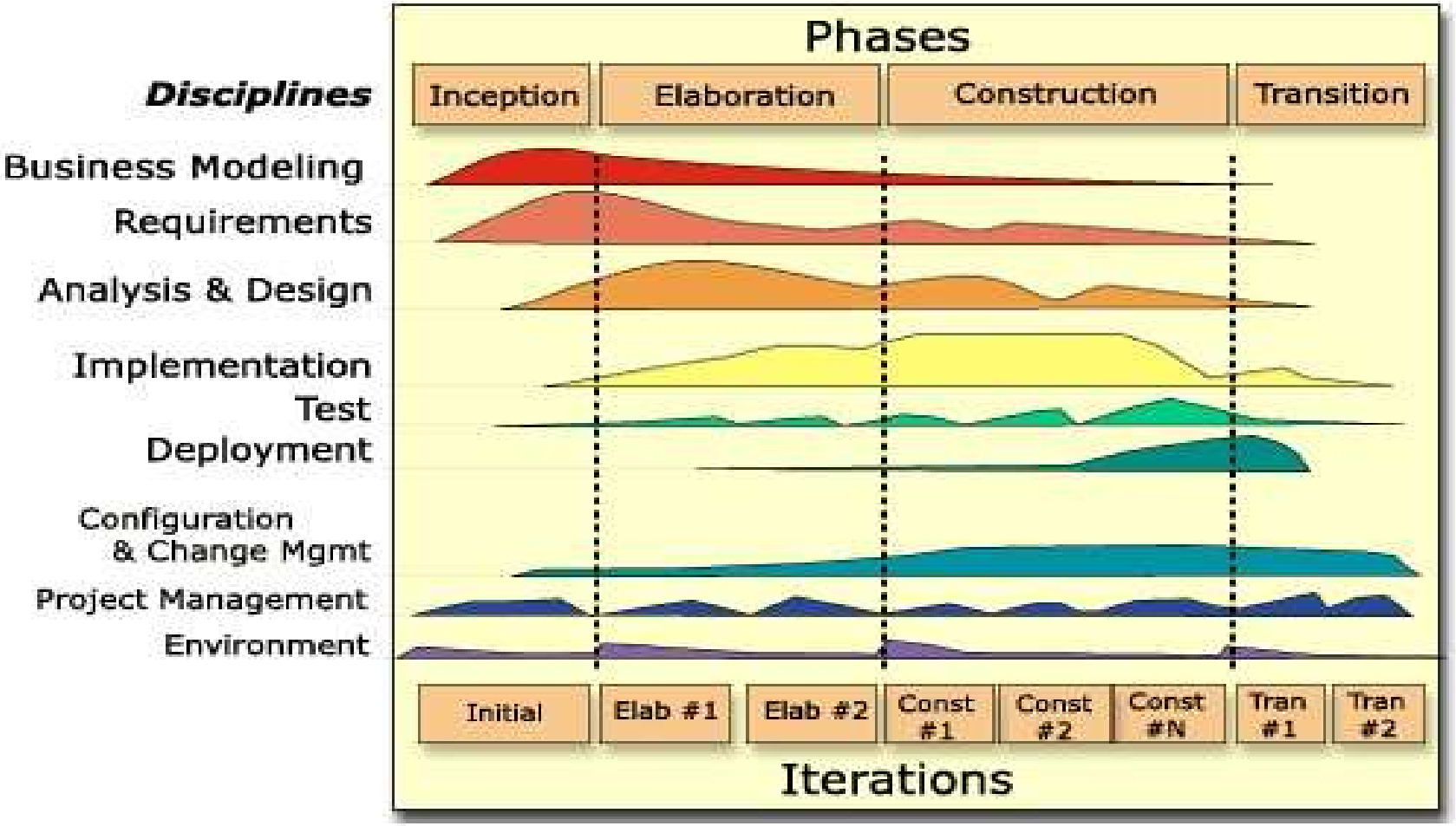
- Neste tipo de processo reconhece-se a necessidade de iterar para controlar riscos.

Modelo em Espiral

- Vantagens
  - maior capacidade de lidar com incerteza, maior controlo de risco
  - promove a inclusão de objectivos de qualidade no processo de desenvolvimento
  - permite demonstração/exploração via protótipos do sistema desde cedo
  - flexibilidade da abordagem – no limite pode tornar-se no modelo em cascata!

- Problemas
  - Dependência da qualidade da avaliação de risco
  - Análise de risco requer competências muito específicas
  - Funciona melhor para projectos grandes
  - Custo pode disparar (análise de risco)
  - Processo rígido

(Rational) Unified Process



Agile Manifesto (Beck et al. 2001)

Ao desenvolver e ao ajudar outros a desenvolver software,
temos vindo a descobrir melhores formas de o fazer.
Através deste processo começámos a valorizar:

Indivíduos e interacções mais do que processos e ferramentas
Software funcional mais do que documentação abrangente
Colaboração com o cliente mais do que negociação contratual
Responder à mudança mais do que seguir um plano

Ou seja, apesar de reconhecermos valor nos itens à direita,
valorizamos mais os itens à esquerda.

Os Doze Princípios do Software Ágil

1. A nossa maior prioridade é, desde as primeiras etapas do projecto, satisfazer o cliente através da entrega rápida e contínua de software com valor.
2. Aceitar alterações de requisitos, mesmo numa fase tardia do ciclo de desenvolvimento. Os processos ágeis potenciam a mudança em benefício da vantagem competitiva do cliente.
3. Fornecer frequentemente software funcional. Os períodos de entrega devem ser de poucas semanas a poucos meses, dando preferência a períodos mais curtos.
4. O cliente e a equipa de desenvolvimento devem trabalhar juntos, diariamente, durante o decorrer do projecto.
5. Desenvolver projectos com base em indivíduos motivados, dando-lhes o ambiente e o apoio de que necessitam, confiando que irão cumprir os objectivos.
6. O método mais eficiente e eficaz de passar informação para e dentro de uma equipa de desenvolvimento é através de conversa pessoal e directa.
7. A principal medida de progresso é a entrega de software funcional.
8. Os processos ágeis promovem o desenvolvimento sustentável. Os promotores, a equipa e os utilizadores deverão ser capazes de manter, indefinidamente, um ritmo constante.
9. A atenção permanente à excelência técnica e um bom desenho da solução aumentam a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho que não é feito – é essencial.
11. As melhores arquitecturas, requisitos e desenhos surgem de equipas auto-organizadas.
12. A equipa reflecte regularmente sobre o modo de se tornar mais eficaz, fazendo os ajustes e adaptações necessárias.

Agile Methods - eXtreme Programming (XP)

- **Fine scale feedback**
  - Pair programming
  - Planning game
  - Test driven development
  - Whole team
- **Continuous process**
  - Continuous integration
  - Design improvement
  - Small releases
- **Shared understanding**
  - Coding standard
  - Collective code ownership
  - Simple design
  - System metaphor
- **Programmer welfare**
  - Sustainable pace

XP - prós e contras

- Vantagens
  - Focado no código
  - Código consistente e elegível
  - Focado nos testes
  - Software disponível incrementalmente
  - Focado na solução mais simples
  - Forte involvimento do cliente
- Dificuldades
  - Exige disciplina, nem sempre bem aceite
  - Preço e duração do projecto?
  - Dificuldades com projectos ou equipas grandes
  - Qualidade não é medida nem planeada (característica emergente?)
  - Testes podem perder a '*big picture*'
  - Muita duplicação de código / muito refactoring

Etapas típicas do Processo de Desenvolvimento de Software

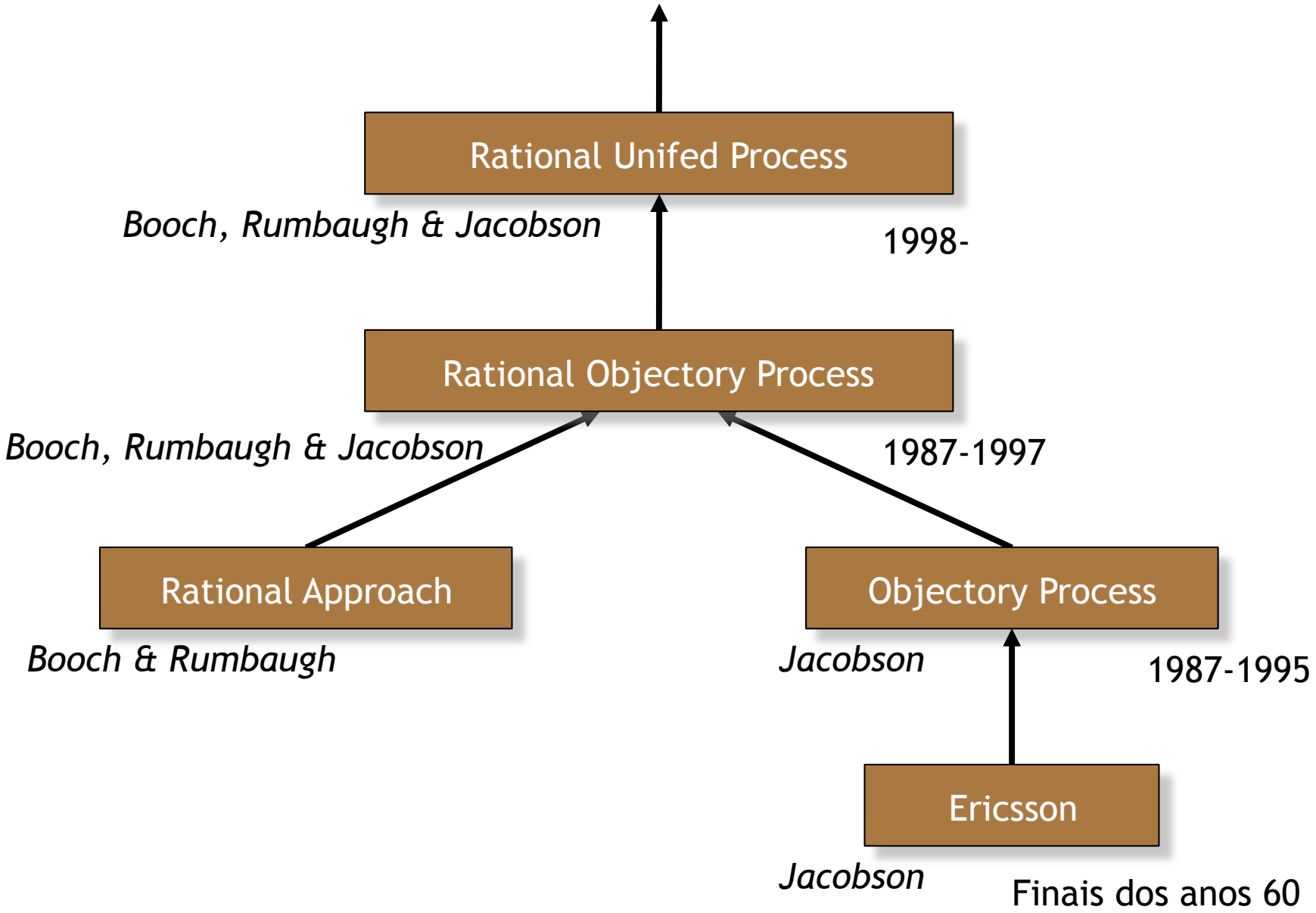
- Analisar o problema
  - Definir o *quê* ➔ Análise de requisitos
- Conceber a solução
  - Definir o *como* ➔ Desenho conceptual; Especificação
- Implementar a solução
  - Fazer ➔ Implementação; Integração
- Testar a solução
  - Verificar e validar ➔ Teste(; análise de modelos)

Abordagem em DSS

A nossa abordagem de Engenharia ao Desenvolvimento de Sistemas Software, passa por algumas ideias fundamentais, a saber:

- Adoptar o Unified Process (UP) como processo de base para o desenvolvimento;
- Seguindo o UP, apostar na Modelação Orientada aos Objectos;
- Seguindo o UP, usar UML (standard da OMG), como notação de modelação;
- Definir alguma **metodologia** na utilização dos modelos UML.
- Realizar o desenvolvimento integrado e coerente de todas as camadas do sistema software, desde a camada de dados até à camada interactiva.

Breve História do Unified Process



O Unified Process

O Unified Process é:

- Uma *framework* para o processo de desenvolvimento, genérica e adaptável.

O Unifed Process fomenta:

- O desenvolvimento baseado em componentes.

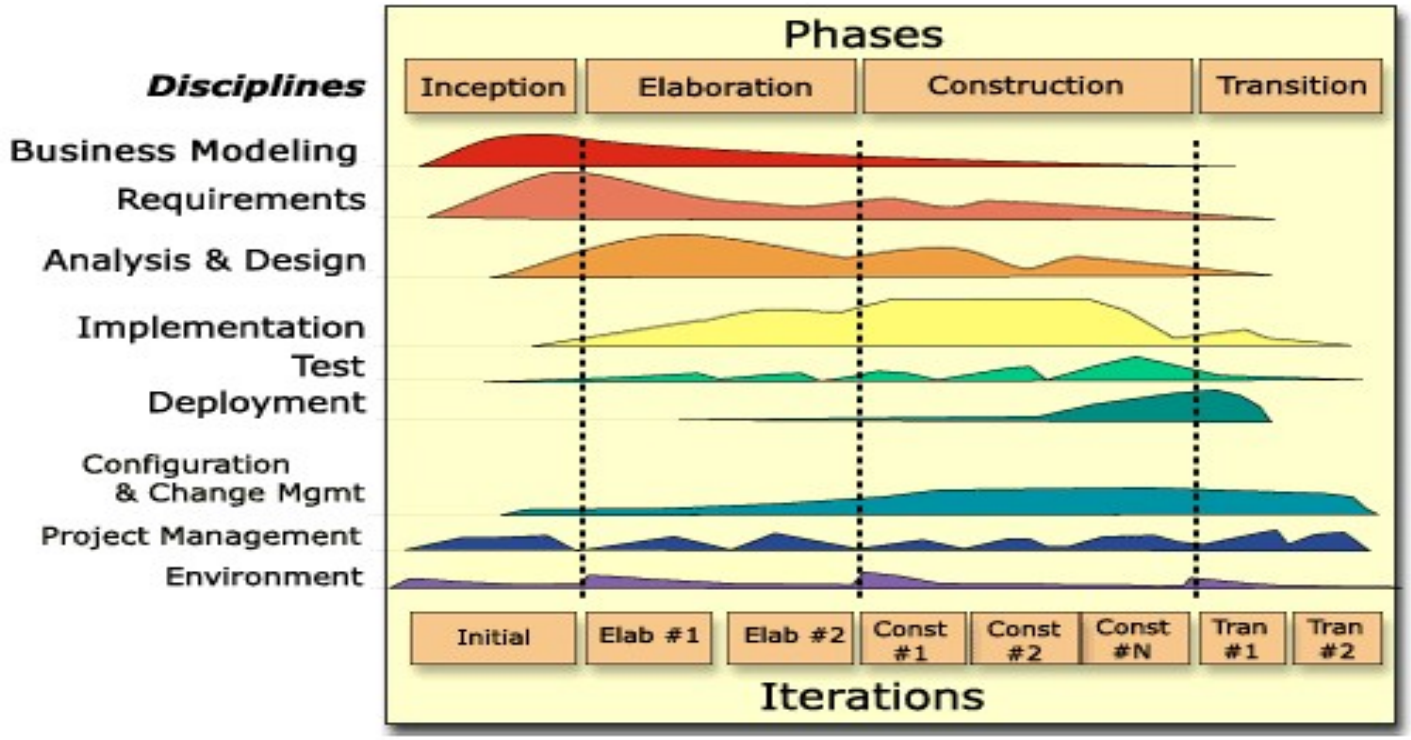
O Unified Process utiliza:

- A UML como ferramenta de modelação durante todas as fases do processo de desenvolvimento.

(Rational) Unified (software development) Process

Três ideias chave:

- guiado por casos de uso (use cases);
- centrado na arquitectura do sistema a desenvolver;
- iterativo e incremental:
  - Início;
  - Elaboração;
  - Construção;
  - Transição.



Desenvolvimento guidado por Use Cases

- Um *use case* representa uma interacção entre o sistema e um humano ou outro sistema;
- O modelo de *use cases* é utilizado para:
  - guiar a concepção do sistema (captura de requisitos funcionais);
  - guiar a implementação do sistema (implementação de um sistema que satisfaça os requisitos);
  - guiar o processo de testes (testar que os requisitos são satisfeitos).

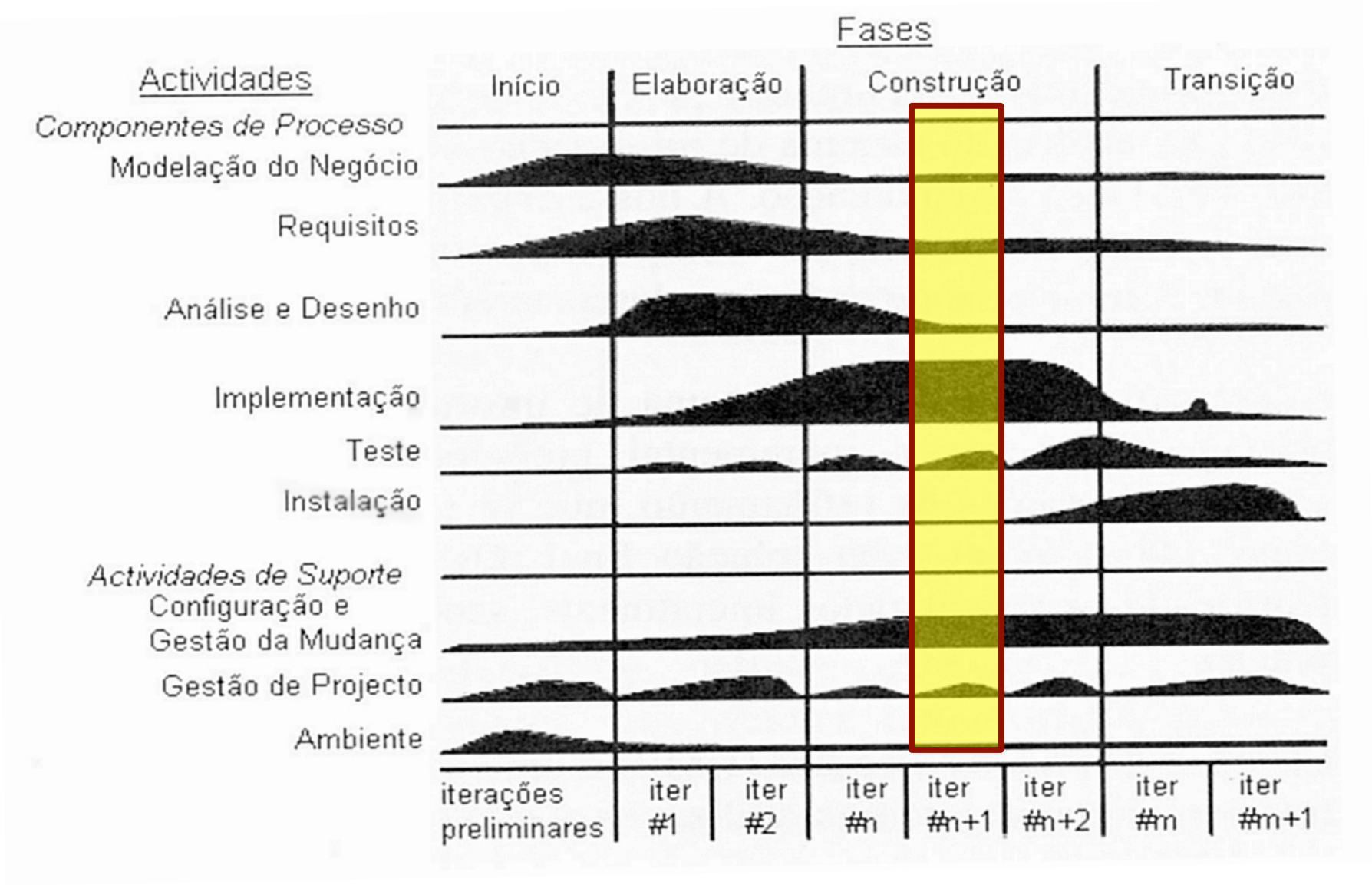
Desenvolvimento centrado na arquitectura

- O modelo de *use cases* descreve a função do sistema, o modelo da arquitectura descreve a forma;
- O modelo arquitectural permite tomar decisões sobre:
  - O estilo de arquitectura a adoptar;
  - Quais os componentes do sistema e quais as suas interfaces;
  - A composição de elementos estruturais e comportamentais.

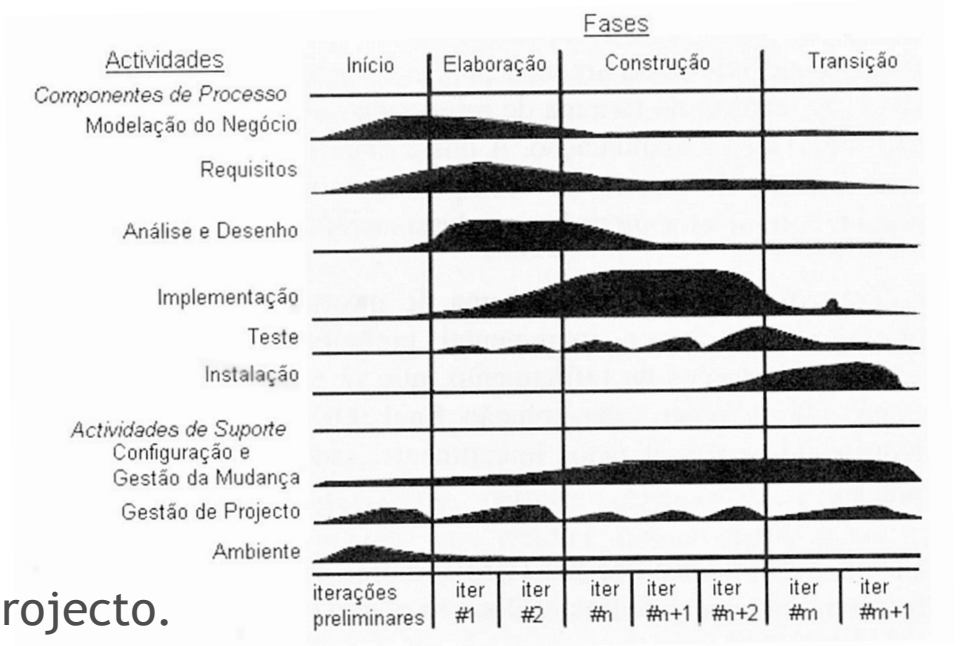
Desenvolvimento iterativo e incremental

- Permite dividir o desenvolvimento em “pedaços geríveis";
- Em cada iteração:
  - identificar e especificar *use cases* relevantes;
  - criar uma arquitectura que os suporte;
  - implementar a arquitectura utilizando componentes;
  - verificar que os *use cases* são satisfeitos.
- Selecção de uma iteração:
  - grupo de *use cases* que extendam a funcionalidade;
  - aspectos de maior risco.

Ciclo de vida do Unified Process



Fases do Unified Process (I)



Início

- Identificar o problema.
- Definir âmbito e natureza do projecto.
- Fazer estudo de viabilidade.

Resultado da fase: decisão de avançar com o projecto.

Elaboração (Análise / Concepção Lógica)

- Identificar o que vai ser construído (quais os requisitos?).
- Identificar como vai ser construído (qual a arquitectura?).
- Definir tecnologia a utilizar.

Resultado da fase: uma arquitectura geral (conceptual) do sistema.

Fases do Unifed Process (II)

Construção (Concepção Física/Implementação)

- Processo iterativo e incremental.
- Em cada iteração tratar um (conjunto de) *Use Case*:

  análise / especificação / codificação / teste / integração

Resultado da fase: um sistema de informação!

Transição

- Realização dos acertos finais na instalação do sistema.
- Optimização, formação.

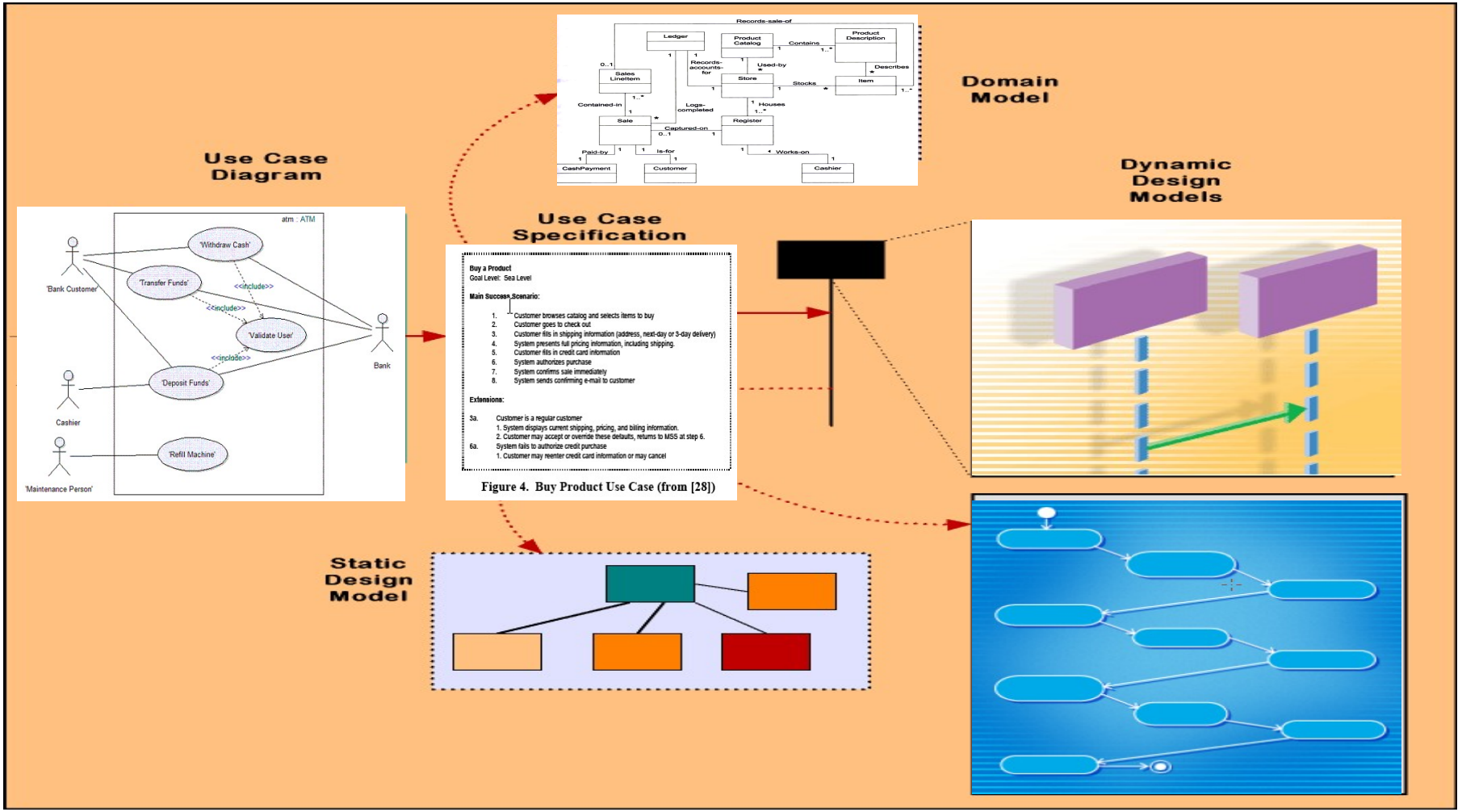
Resultado da fase: um sistema instalado e 100% funcional (espera-se!).

O RUP (Rational Unifed Process)

- O RUP é uma concretização do Unified Process.
- O RUP fornece:
  - ferramentas de gestão do processo de desenvolvimento segundo a *framework* definida pelo Unified Process (ver página 51);
  - ferramentas para a modelação e desenvolvimento baseadas no UML;
  - uma base de conhecimento (knowledge base).

- Na verdade as coisas passaram-se um pouco ao contrário...

Abordagem em DSS



Domain Model + Use Case Model são refinados sistematicamente nos outros modelos, estruturais e comportamentais, idealmente diferenciando objectos que são de camadas distintas (dados, computacional, negócio e IU).

Unified Software Development Process

Sumário:

- Processos de desenvolvimento de software
  - Sequenciais – modelo em cascata
  - Iterativos – modelo em espiral; *Unified Process*
  - Ágeis – Extreme programming
- Abordagem em DSS
  - O Unified Process
  - O ciclo de vida do Unified Process
  - O RUP (Rational Unifed Process)