INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

RUP – Unified Process

Profª Jocelma Rios

Nov/2012

O que pretendemos:

- Reforçar os aspectos que caracterizam o processo iterativo e incremental
- Identificar como atingir os objetivos dos projetos de *software* orientado a objetos, gerenciando tempo e recursos
- Refletir sobre a importância dos processos definidos e controlados para o sucesso do desenvolvimento de *software*
- Refletir sobre as possibilidades de customização do RUP para adequação aos vários tipos de *software*

Motivação para a OO


Como o cliente explicou


Como o lider de projeto entendeu

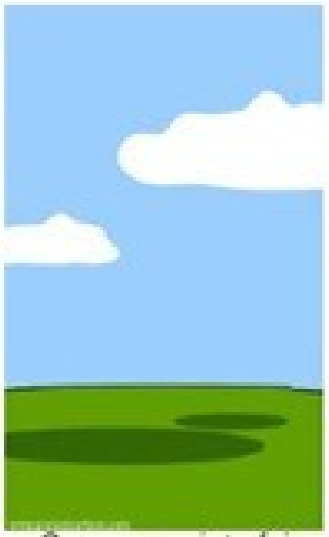

Como o analista planejou


Como o programador codificou

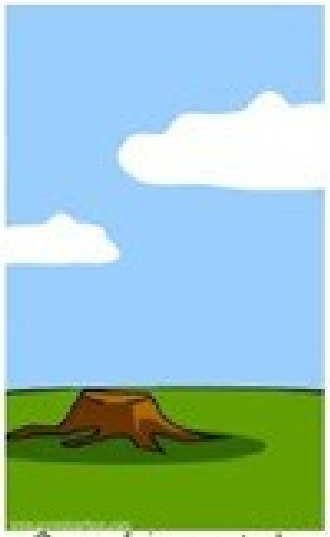

O que os beta testers receberam


Como o consultor de negocios descreveu


Valor que o cliente pagou


Como o projeto foi documentado


O que a assistencia tecnica instalou


Como foi suportado


Quando foi entregue


O que o cliente realmente necessitava

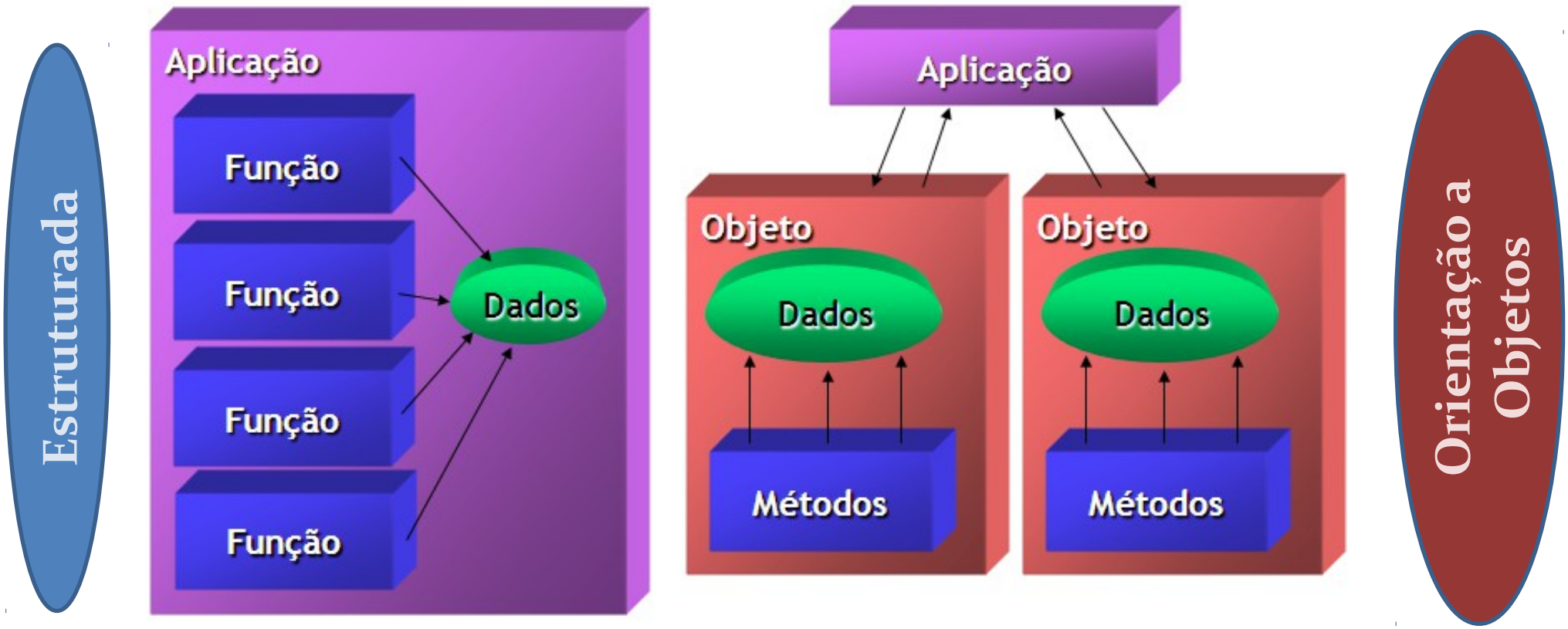
Surgimento da OO → Cenário

- Crescente demanda de soluções computacionais
- Evolução acentuada do *hardware*
- Demanda muito superior à capacidade de desenvolvimento
- Distância semântica entre os modelos projetados e a realidade analisada
- Qualidade insuficiente dos produtos
- Estimativas de custo e tempo raramente cumpridas nos projetos

O problema da abordagem de SW orientado a funções

- Ênfase nas funções leva a sistemas com muita redundância
- Inconsistentes e difíceis de serem integrados
- Dados possuem existência própria nas organizações independentemente dos processos que os manipulam
- Dados são muito mais estáveis que as funções/processos em uma organização

Análise Estruturada *vs.* OO

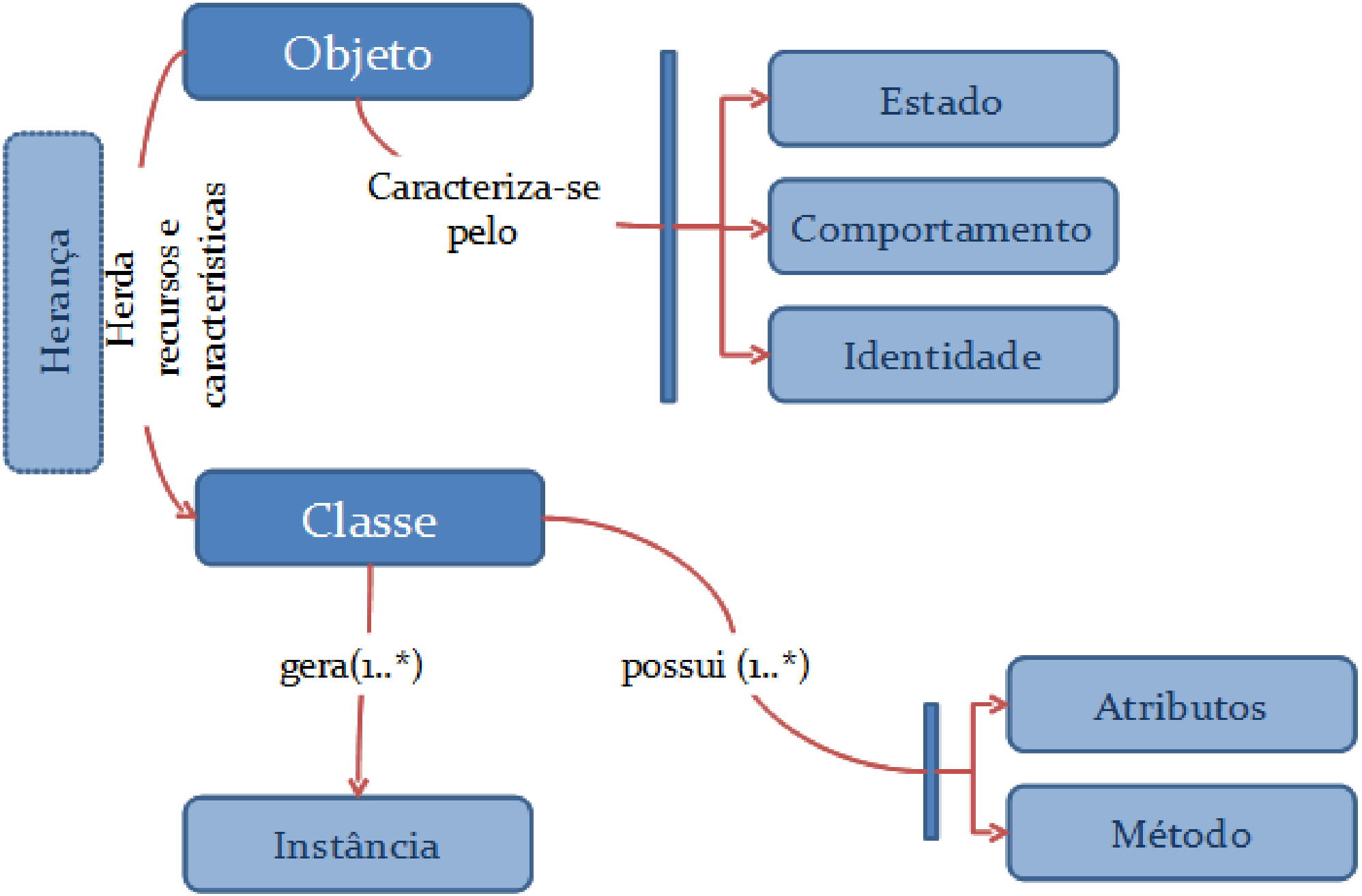


POO -> Estratégia de projeto em que o *software* é pensado em termo de coisas em vez de funções

Conceitos basilares da OO

- ✓Objeto
- ✓Estado
- ✓Comportamento
- ✓Identidade
- ✓Classe
- ✓Instância
- ✓Método
- ✓Herança
- ✓Persistência
- ✓Polimorfismo
- ✓Encapsulamento
- ✓Ligação

Conceitos basilares da OO

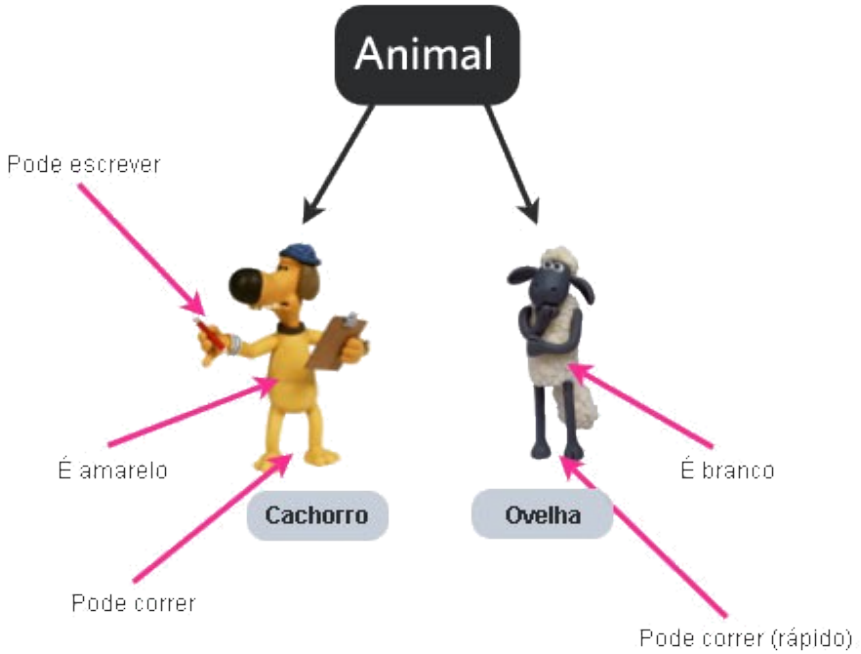


Conceitos basilares da OO

- ✓Objeto
- ✓Estado
- ✓Comportamento
- ✓Identidade
- ✓Classe
- ✓Instância
- ✓Método
- ✓Herança
- ✓Persistência
- ✓Polimorfismo
- ✓Encapsulamento
- ✓Ligação

Conceitos basilares da OO

- ✓Objeto
  - Unidade real ou abstrata
  - Entidade que incorpora uma abstração da realidade
  - Possui estado, exibe um comportamento bem definido e possui identidade própria



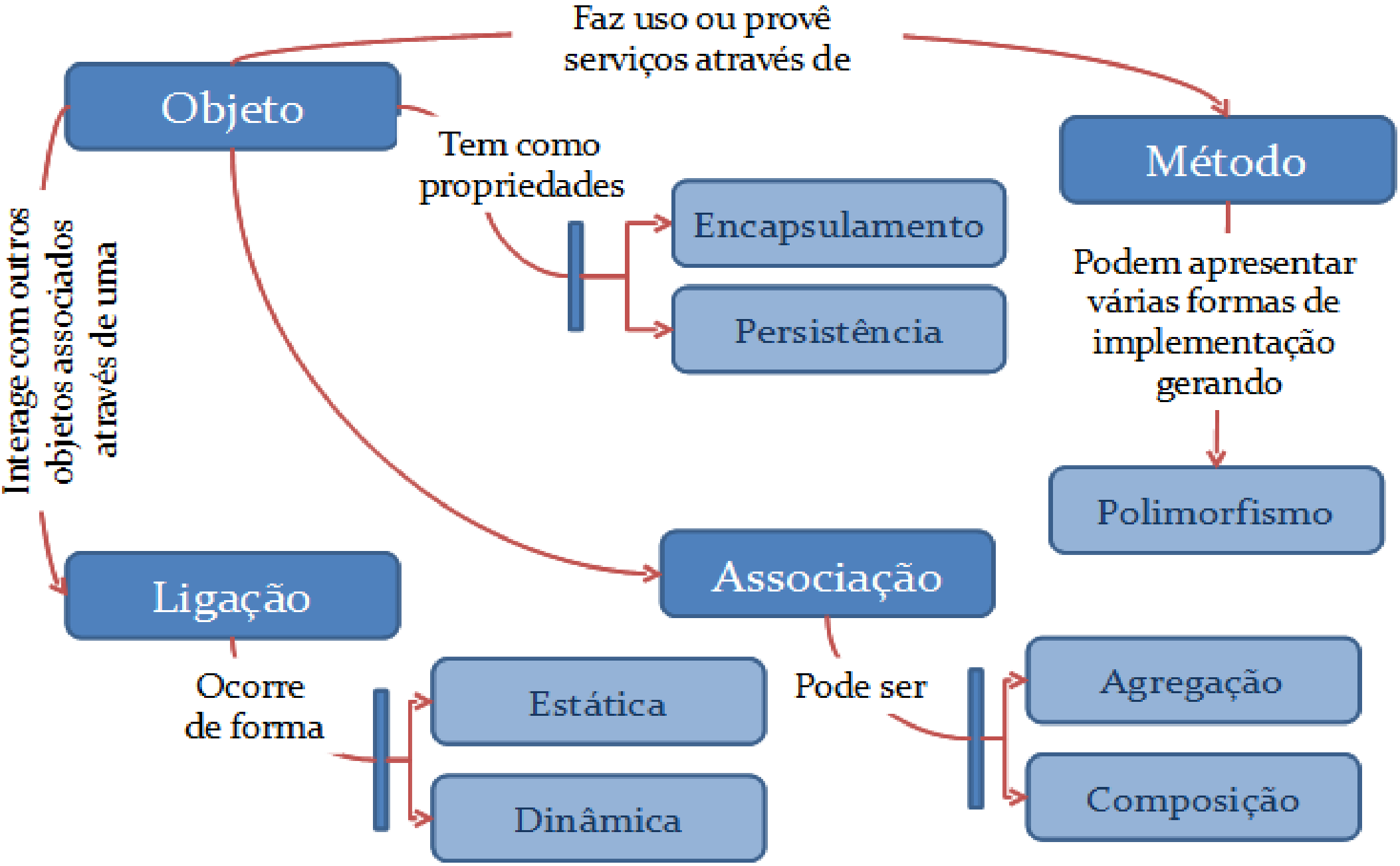
Conceitos basilares da OO

- ✓ Estado
  - Conjunto das propriedades de um objeto
    - Atributos e valores a ele associados
- ✓ Comportamento
  - Conjunto de serviços (operações) que ele próprio ou outros objetos (clientes) podem requisitar
  - Operações
    - Manipulam a informação de estado de um objeto
    - Método de uma classe
- ✓ Identidade
  - Identificador único pelo qual pode ser referenciado inequivocadamente

Conceitos basilares da OO

- ✓Classe
  - Modelo que descreve a estrutura e o comportamento de objetos
  - ex: ClasseFuncionário
- ✓Instância
  - Um dado objeto que se comporta da maneira especificada pela classe mãe
  - ex: João Alfredo Ramos
- ✓Método
  - Implementação de uma operação
  - ex: RegistraPonto(), HomologaCronograma()
- ✓Herança
  - Compartilhamento de atributos e operações entre classes baseado numa relação de hierarquia
  - Tipos: Herança simples (de uma única classe) e herança complexa (de mais de uma classe)

Conceitos basilares da OO



Conceitos basilares da OO

- ✓Persistência
  - Propriedade que garante a existência do objeto além do tempo e espaço de sua criação
- ✓Polimorfismo
  - Quando a mesma operação possui métodos (implementação) diferentes em classes distintas
- ✓Encapsulamento
  - Interação entre objetos sem conhecimento do funcionamento interno
- ✓Associação
  - Permite que objetos de uma ou mais classes se relacionem.
  - Pode ser do tipo agregação (menor coesão) ou composição (maior coesão)
- ✓Ligação
  - É estabelecida quando uma mensagem é trocada entre dois objetos visando a execução de um método
  - Pode ser estática (compilação) ou dinâmica (execução)

UML → modelagem na OO

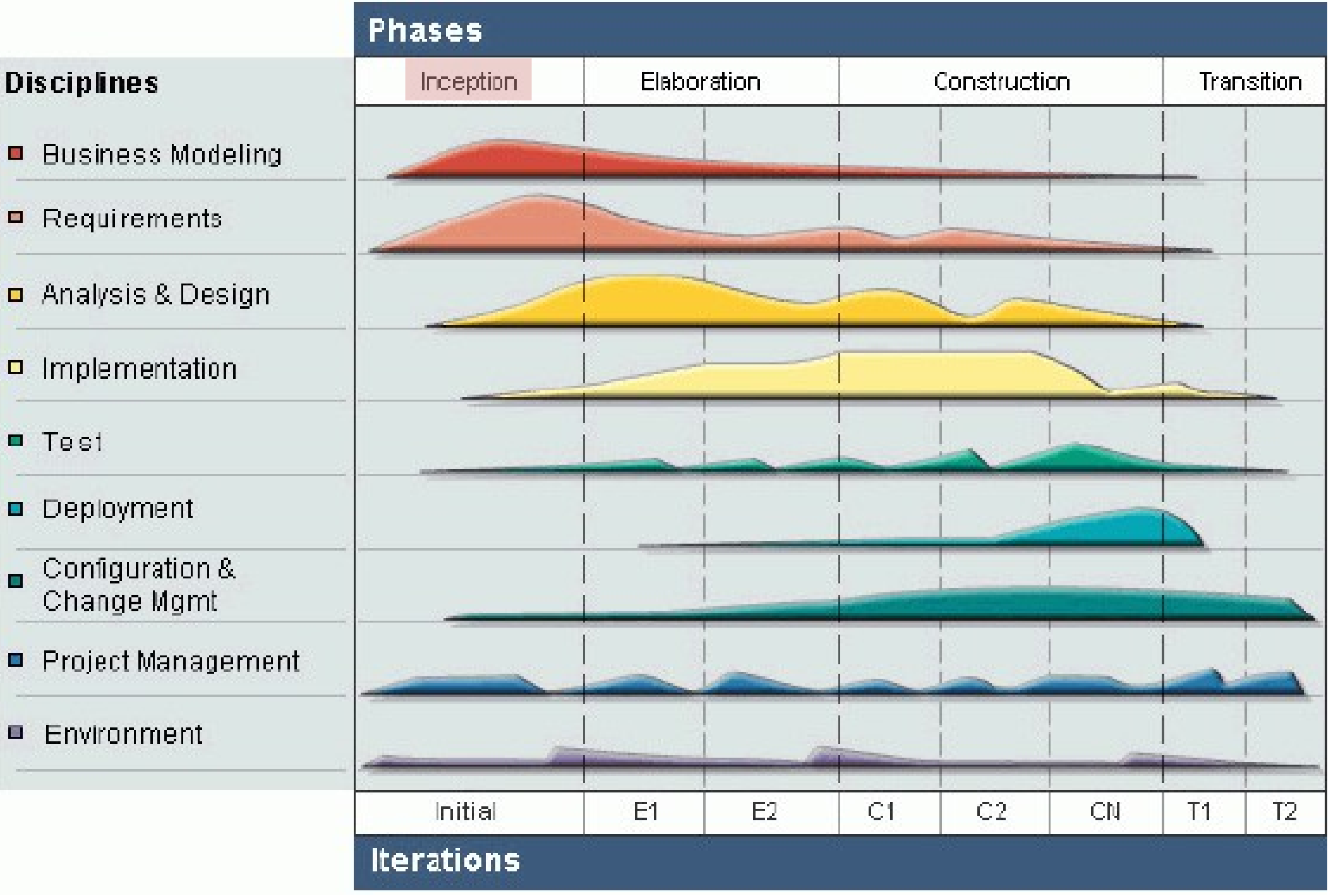
- ✓UML é uma linguagem de modelagem, não proprietária, com notação gráfica
- ✓Destina-se à especificação, construção, visualização e documentação de projetos de *software*
- ✓Não é uma metodologia de desenvolvimento, ou seja, não determina o que deverá ser feito e em qual sequência

- ✓Tipos de diagramas:
  - Comportamentais: casos de uso, transição de estados, atividade
  - Estruturais: classes, objetos, componentes, implementação, pacotes, estrutura
  - Interacionais: sequência, interatividade, colaboração, tempo

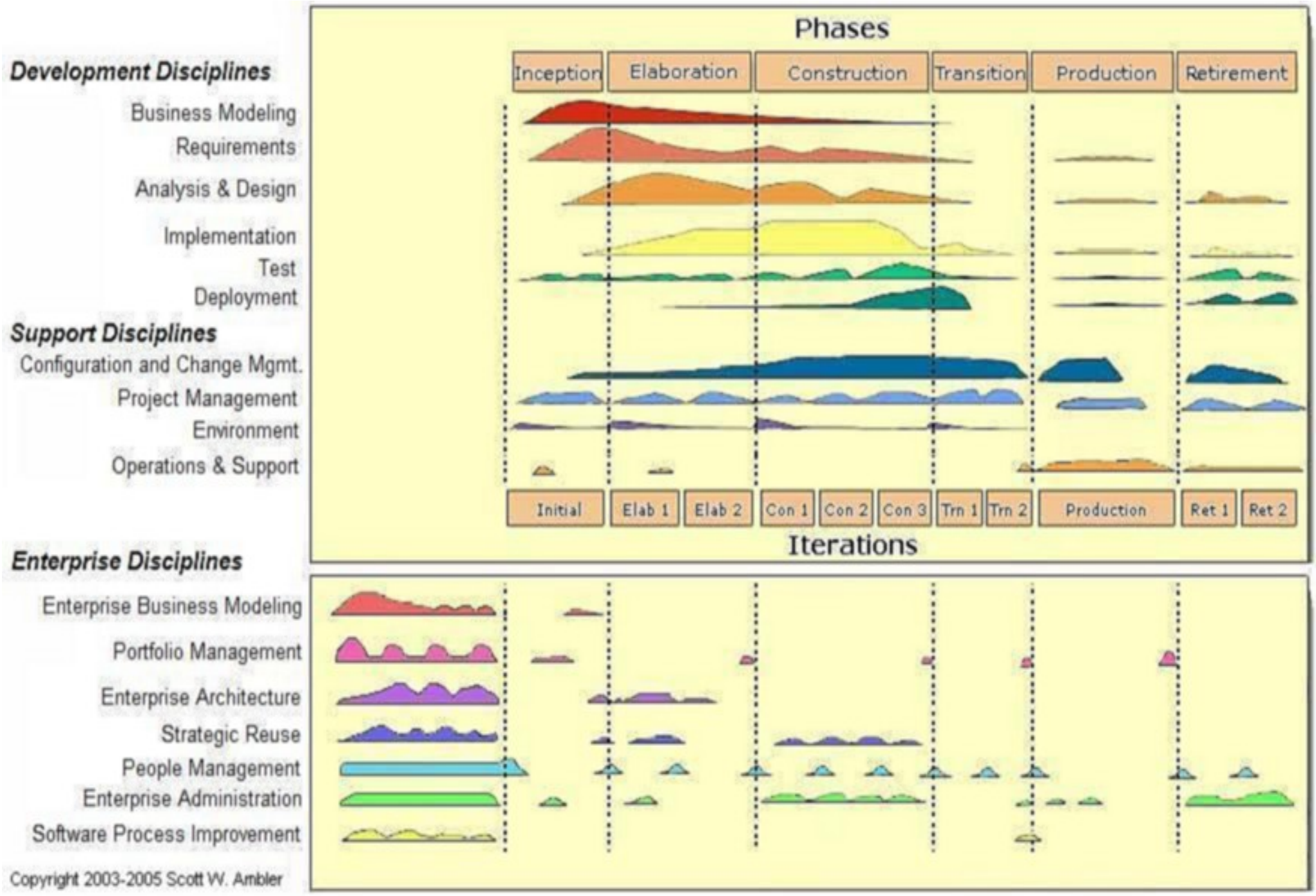
Processo de *software* OO

- ✓O **Processo Unificado – RUP** foi criado pela Rational, adquirida mais tarde pela IBM
  - O RUP utiliza a abordagem da OO em sua concepção
  - Utiliza a UML para modelagem
  - É melhor aplicado em grandes projetos com grandes equipes
  - É passível de customização, ou seja, é possível ser utilizado com equipes pequenas desde que seja feita a devida adequação
    - Sua customização é apoiada por ferramentas da IBM → Rational Suites

Processo Unificado



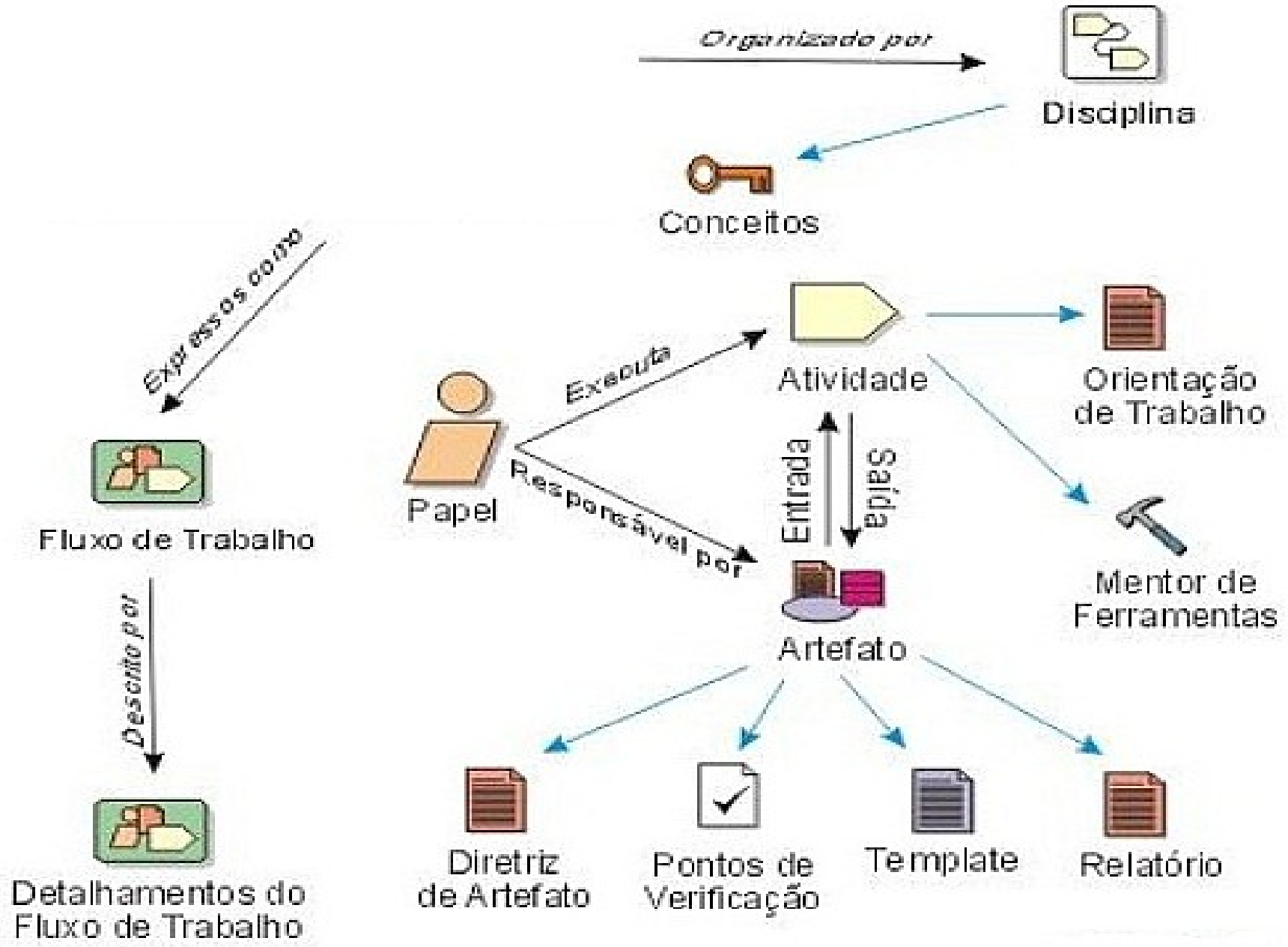
Processo Unificado



Processo Unificado

Saiba mais: Metodologia RUP (espanhol) - www.youtube.com/watch?v=bV8MgbfOO5g

Processo Unificado



Fase 1: Concepção (*Inception*)

✓**Objetivos**

- Definir objetivos do projeto
- Identificar/Modelar a visão da organização para o software
- Identificar e organizar requisitos
- Descobrir se o software é viável
- Planejar o *desenvolvimento*

✓**Disciplinas (especialistas) envolvidas**

- Modelagem de negócios
- Ambiente computacional
- Análise de requisitos
- Gerência de projetos

✓**Diagramas UML**

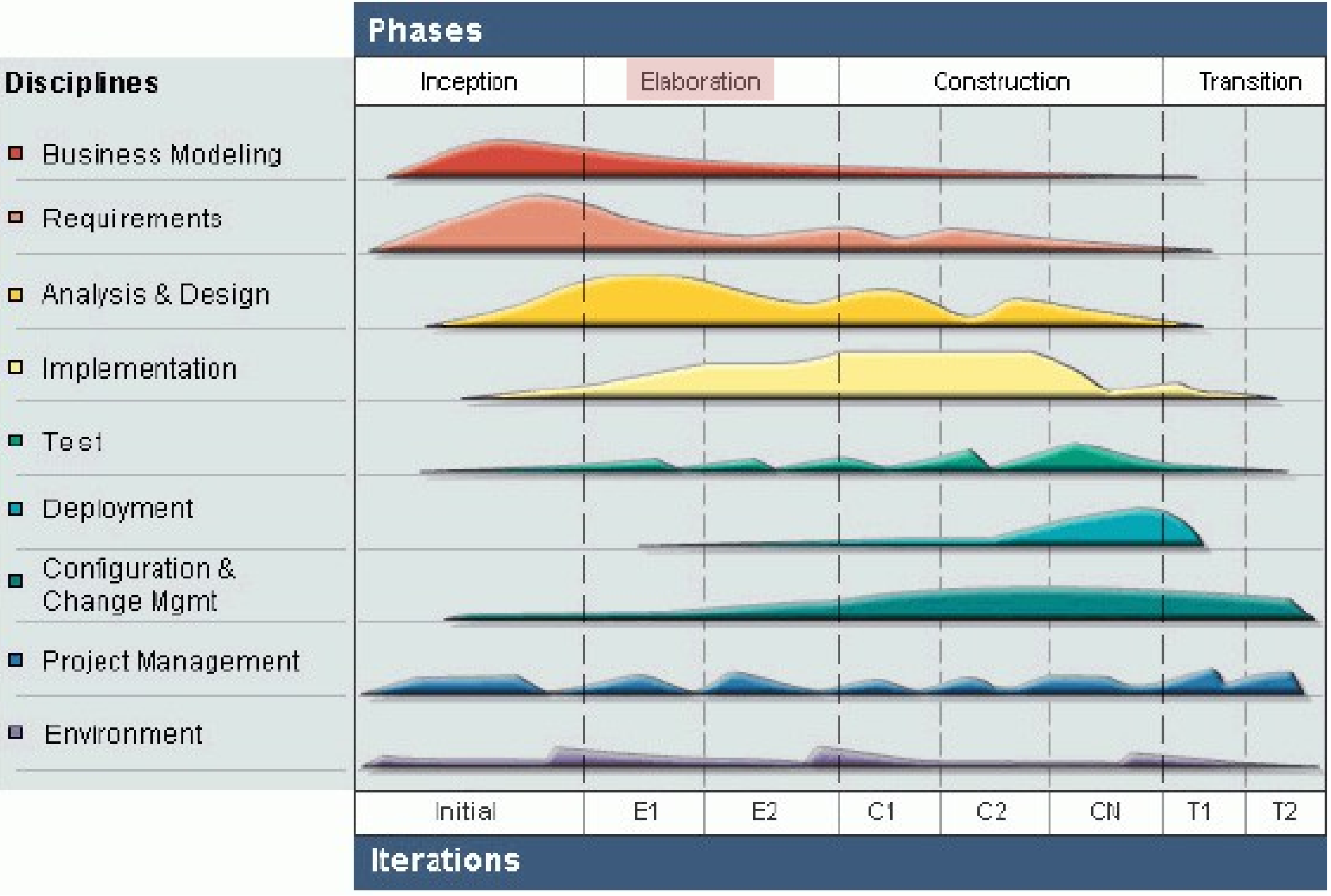
- Comportamentais: Casos de uso, transição de estados e atividade (preliminar)

✓**Artefatos gerados (principais)**

- Documento de Visão (uso de diagramas)
- Proposta da solução computacional (preliminar)
- Plano de Projeto (requer algumas iterações até finalizar)

Fase 1: Concepção (*Inception*)

Processo Unificado

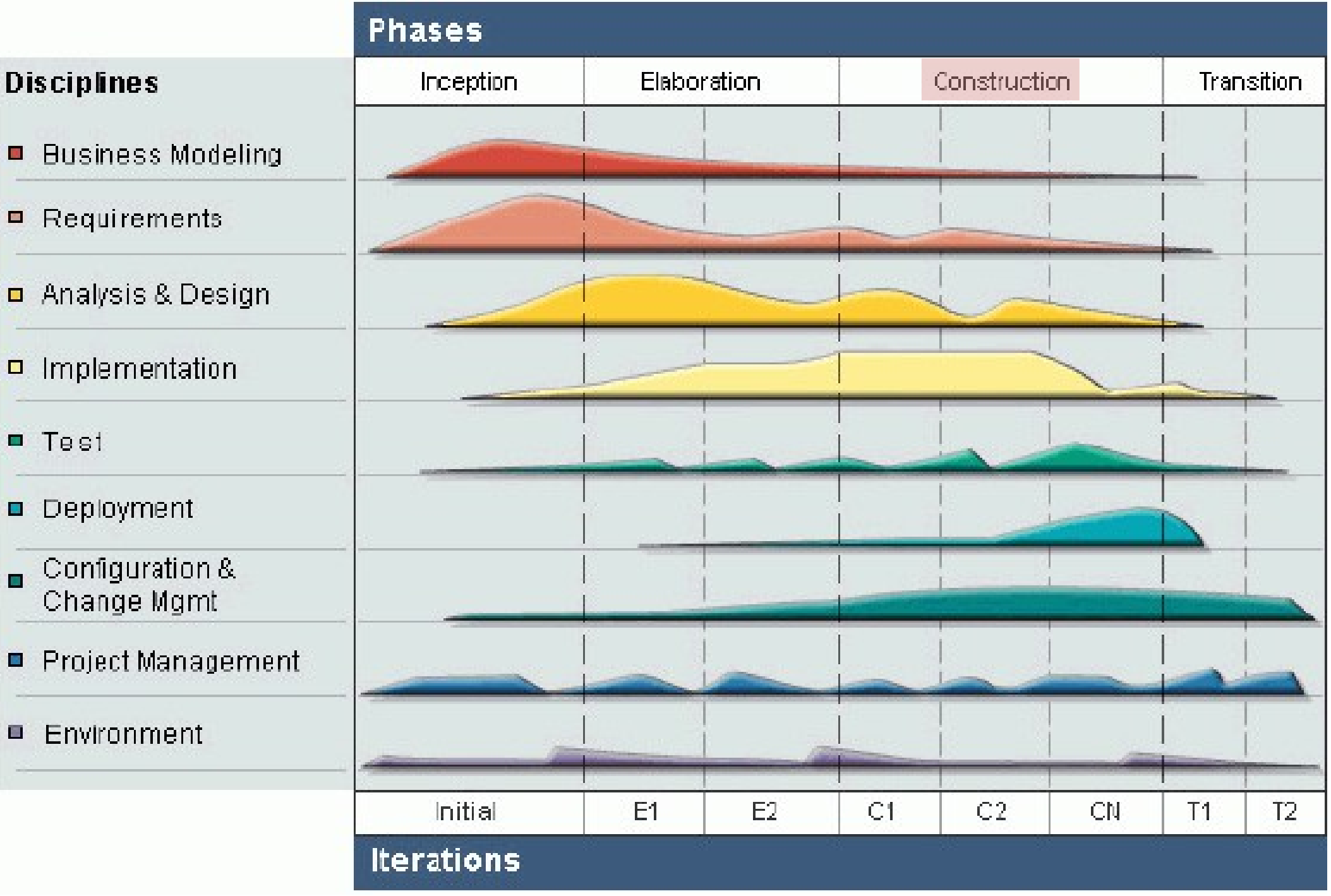


Fase 2: Elaboração (*Elaboration*)

- ✓**Objetivos**
  - Detalhar documentação produzida na fase de Concepção
  - Refinar Plano de Projeto
  - Definir a Solução Computacional
- ✓**Disciplinas (especialistas) envolvidos**
  - Análise de requisitos
  - Análise e design
  - Implementação
  - Gerência de projetos
- ✓**Diagramas UML**
  - Estruturais: classes, objetos, componentes, pacotes, implementação, estrutura
  - Interacionais: sequência, interatividade, colaboração, tempo
- ✓**Artefatos gerados**
  - Especificação funcional da Solução
  - Especificação detalhada da solução
  - Projeto de arquitetura
  - Projeto de implantação
  - Plano de testes e integração

Fase 2: Elaboração (*Elaboration*)

Processo Unificado

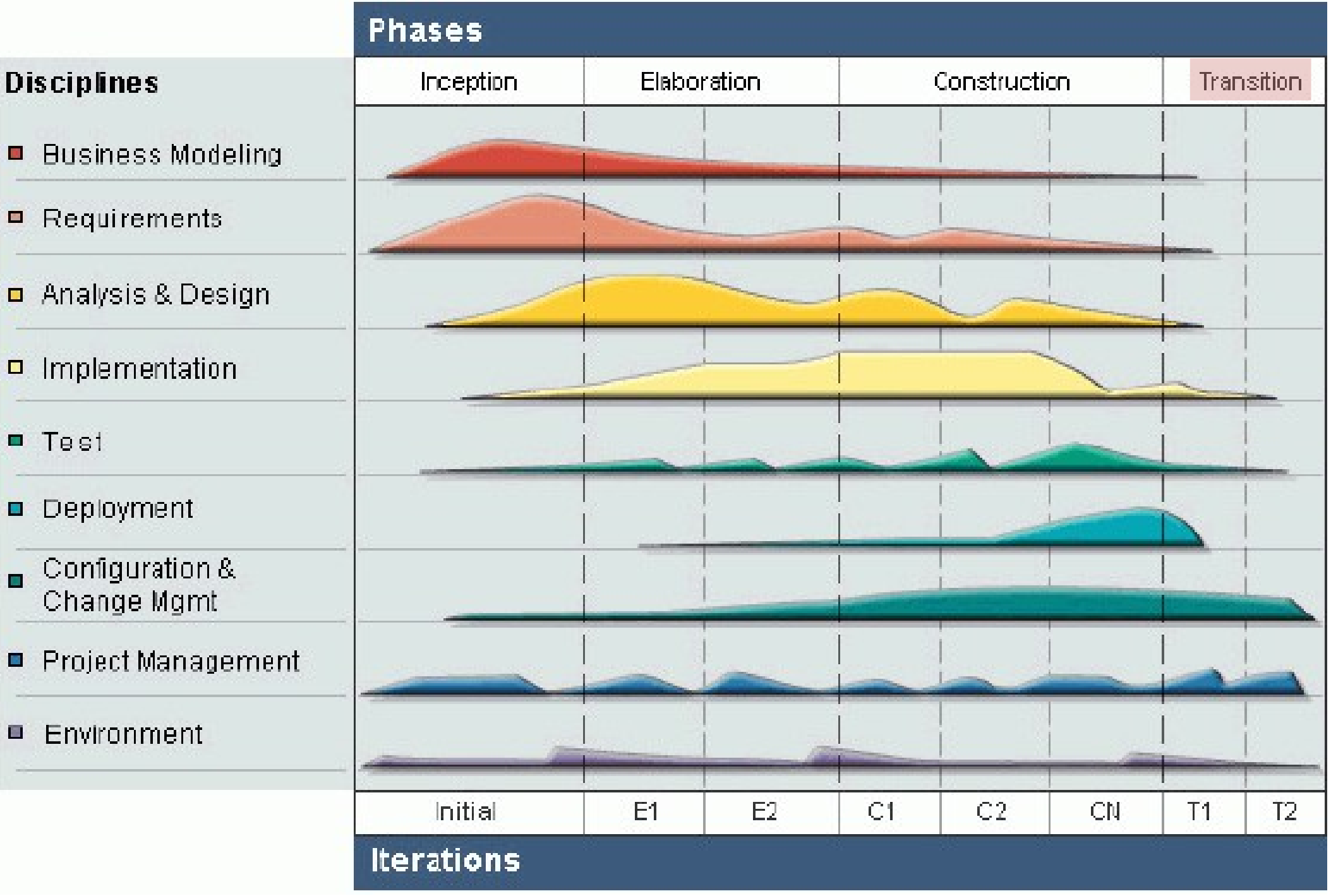


Fase 3: Construção (*Construction*)

- ✓ **Objetivos**
  - Implementar o software
  - Realizar testes
- ✓ **Disciplinas (especialistas) envolvidos**
  - Análise de requisitos
  - Análise e *design*
  - Implementação
  - Testes
  - Implantação
  - Gerência de projetos
- ✓ **Diagramas UML**
  - Estruturais: objetos, classe, componentes, implementação, pacotes e estrutura
- ✓ **Artefatos gerados**
  - Código-fonte
  - Executáveis
  - Relatórios de testes

Fase 3: Construção (*Construction*)

Processo Unificado



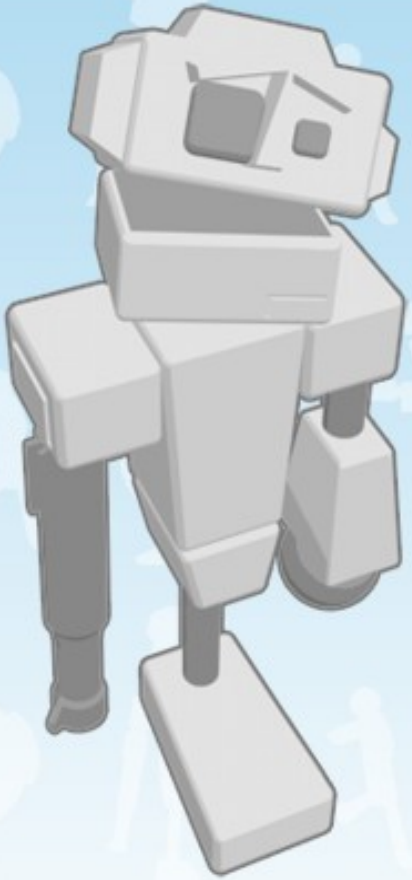
Fase 4: Transição (*Transiction*)

- ✓ **Objetivos**
  - Implantar o *software*
  - Realizar os últimos testes
  - Preparar o ambiente do usuário e também a ele
  - Entregar
- ✓ **Disciplinas (especialistas) envolvidos**
  - Análise de requisitos
  - Análise e design
  - Implementação
  - Gerência de projetos
- ✓ **Diagramas UML**
  - Interacionais (para possíveis consultas): interatividade, colaboração, tempo, sequência
- ✓ **Artefatos gerados**
  - Relatórios de testes
  - Manuais
  - Avaliação do treinamento
  - Homologação do *software*

Fase 4: Transição (*Transiction*)

Importante

RUP Overview



Dica

As fases do jogo RUP **não têm duração fixa.**

Elas vão durar o tempo necessário para eliminar os respectivos riscos (...riscos de negócio, riscos técnicos...lembra?). Entretanto, algumas "forças do mal" se manifestam contra o nosso jogo fazendo com que a fase se estenda por um tempo maior que o previsto.

Já que o assunto, agora, é a fase "Elaboração", vamos **identificar as forças que atuam contra ela, fazendo com que durem mais tempo:**

- Arquitetura não comprovada ou instável
- Requisitos instáveis
- Ambiente de desenvolvimento instável
- Requisitos não-funcionais vagos ou complexos.

Por que?

Importante

RUP Overview

Por que?

Para refletir...

O **RUP** propõe uma série de atividades baseadas nas disciplinas envolvidas na engenharia de *software*, que geram diversos artefatos como produto.

A princípio parece ser uma carga difícil para uma equipe reduzida.

Como seria possível utilizar o **RUP** nessas circunstâncias?

Referências

SiteBOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML, Guia do Usuário**. Rio de Janeiro: Campus,2000.

FOWLER, M. **UML Essencial**. São Paulo: Bookman, 2004. Cap. 2 (*disponível on line*, parcialmente)

KRUCHTEN, P. **The Rational Unified Process**: An Introduction. 2ª. ed. Massachusets: Addison-Wesley, 2000.

IBM. **Rational Unified Process**: best practices for software development teams. TP026B, Rev 11/01. Disponível em: www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf . Acesso em 03 out. 2011.

LARMAN, C. **Utilizando UML e Padrões**: uma introdução à análise e ao projeto orientados a objetos. São Paulo: Bookman, 2005. (disponível *on line*)

PRESMANN, R. **Engenharia de Software**: uma abordagem profissional. 7. ed. Rio de Janeiro: Mc Graw Hill, 2011. Cap. 2

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. Rio de Janeiro: Pearson, 2007. Cap. 4