Processo de Desenvolvimento

Paulo Trigo Silva

O que é um Processo de Desenvolvimento ?

- Um Processo define
  - quem (*who*) faz o quê (*what*), quando (*when*) e como (*how*)
  - de modo a atingir determinado objectivo
- ... no Desenvolvimento de *Software*, o objectivo é,
  - construir um produto de *software* ou evoluir um já existente
- Um Processo de Desenvolvimento de *Software*,
  - deve ser capaz de evoluir ao longo de vários anos
  - e a sua evolução deve acompanhar os movimentos a nível das
    - tecnologias
    - ferramentas
    - pessoas
    - padrões organizacionais

Processo de Desenvolvimento – evolução

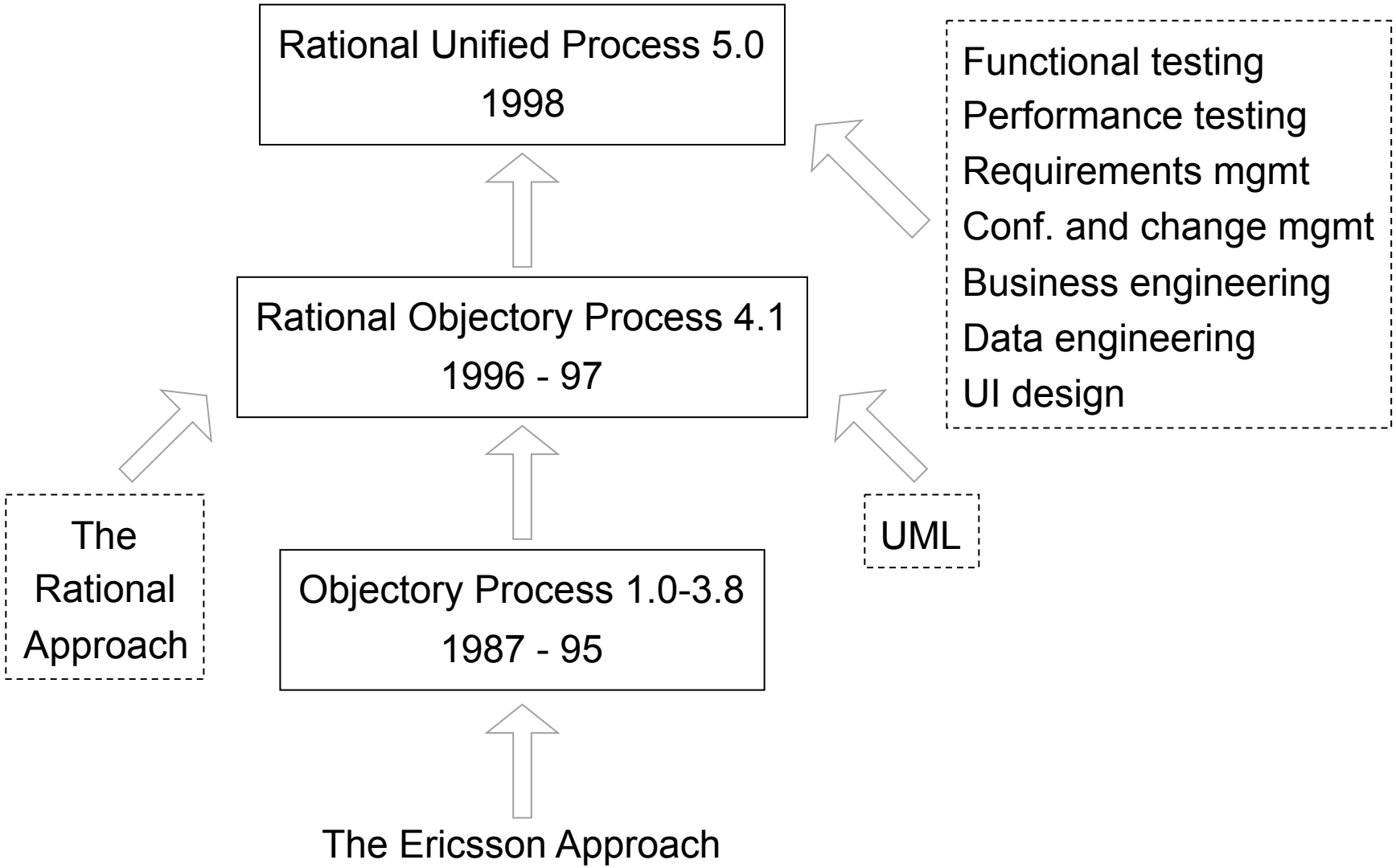
- Tecnologias – Processo tem que ser suportado por tecnologias
  - ambientes de desenvolvimento, sistemas operativos, redes, ...
  - p.e. há 10 anos atrás o suporte à modelação visual era muito caro; os diagramas eram desenhados à mão – isso reduz o grau de integração dessas técnicas de modelação no processo de desenvolvimento
- Ferramentas – Processo e Ferramentas avançam em paralelo
  - processo justifica investimento na construção de Ferramenta de suporte
- Pessoas – Processo será concretizado através de Pessoas
  - não se pode exigir perícia para além das capacidade das pessoas
  - p.e. verificar consistência de um modelo visual, exige da pessoa, uma enorme perícia – possível de embeber numa ferramenta computacional
- Padrões Organizacionais – Processo adequado à realidade actual
  - trabalho à distância, falta de programadores, subcontratação, ...

Processo – para quê ?

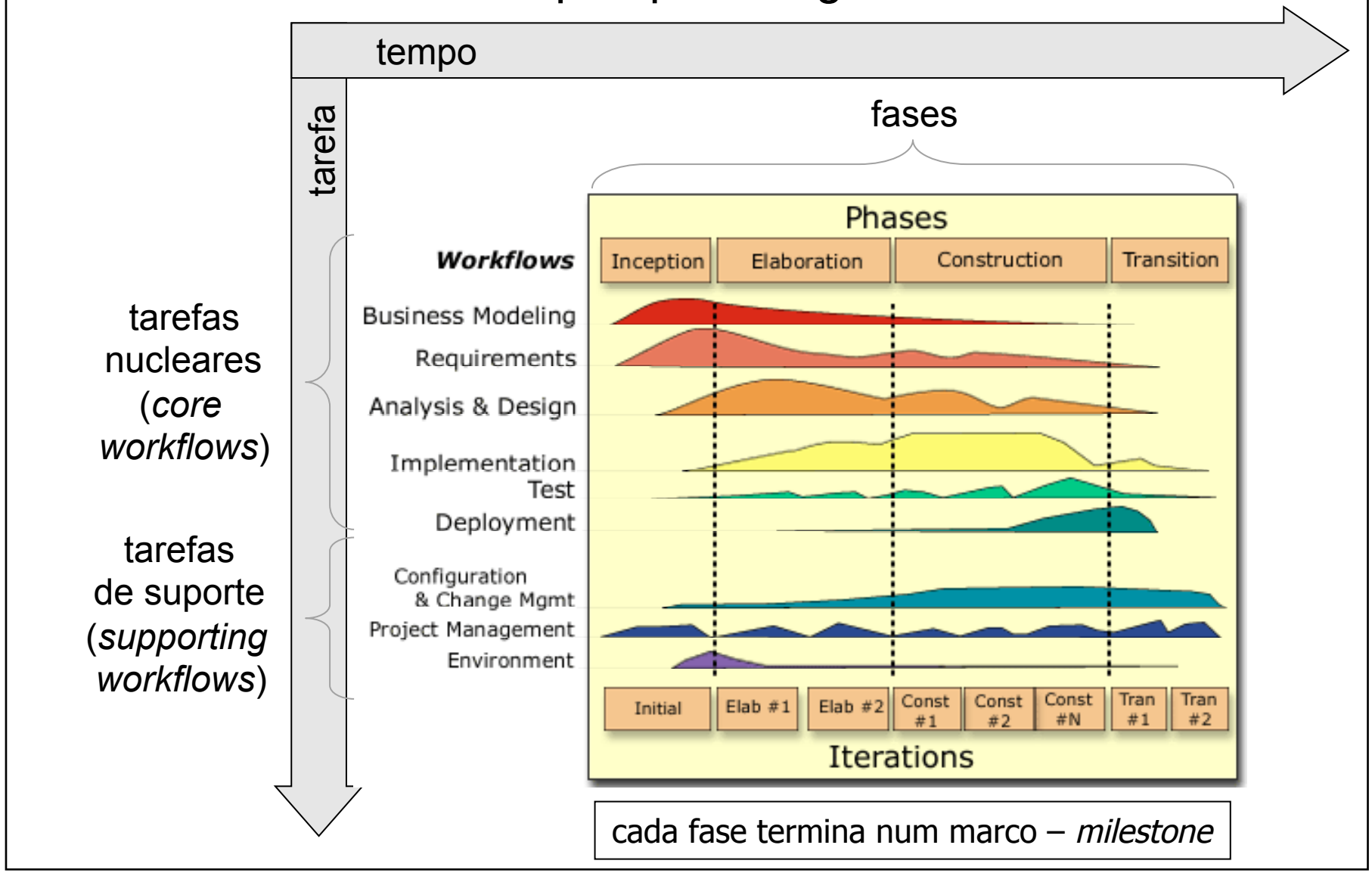
- Desenvolvimento de *Software* necessita de
  - guia de ordenação das actividades de cada equipa de trabalho
  - indicação de tarefas de cada pessoa e cada equipa como um todo
  - especificação dos artefactos a desenvolver
  - critérios de verificação da evolução das actividades e artefactos
- ... esses elementos devem estar presentes num Processo
  - pode marcar a diferença entre o seu sucesso ou insucesso
  - e representa a experiência acumulada ao longo de vários anos
- UP – *Unified Process* (também conhecido por RUP – *Rational* UP)
  - "Processo Unificado" de Desenvolvimento de *Software*
  - resulta de mais de 30 anos de experiência
  - coloca ênfase na utilização da UML (linguagem *standard* de modelação)
  - explora actuais tecnologias, ferramentas, pessoas e modos de trabalho

Paulo Trigo Silva

Creating the Unified Process

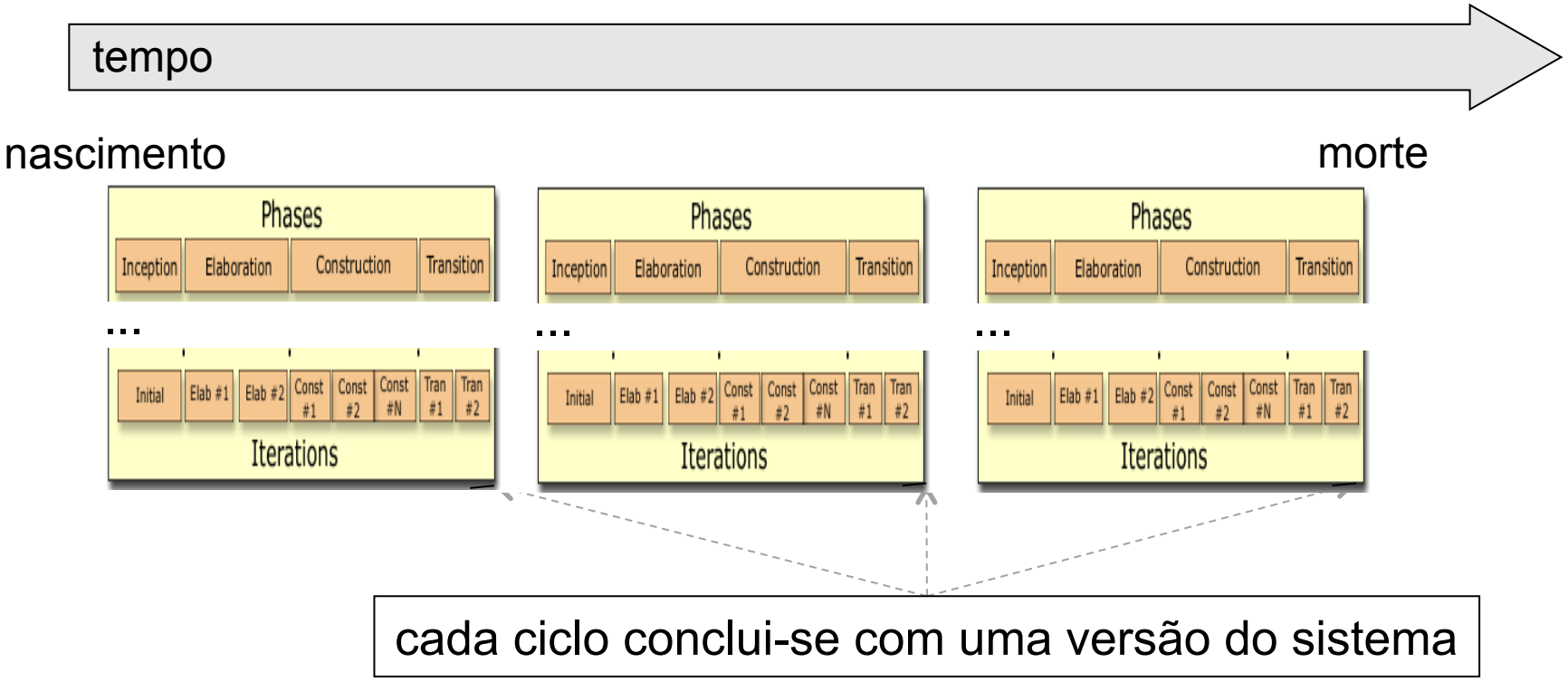


Processo Unificado – perspectiva geral

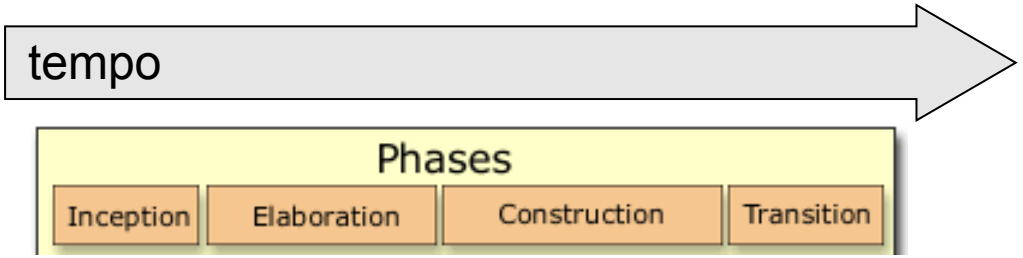


A vida do *Unified Process*

- O *Unified Process* repete-se numa série de ciclos
  - em cada ciclo são percorridas todas as fases (*phases*) do processo
  - cada ciclo conclui-se com uma versão do sistema (*system release*)
  - o conjunto de todos esses ciclos constitui a vida do sistema (*system life*)



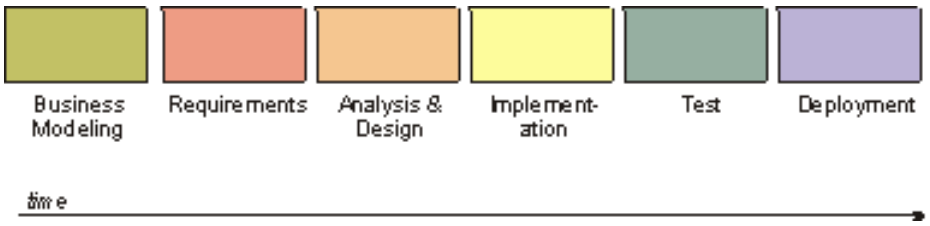
As fases de cada ciclo



- Avaliação Preliminar – *Inception*
  - definição do contexto do projecto; construção de casos de utilização
- Elaboração – *Elaboration*
  - planeamento; especificação dos requisitos funcionais e não funcionais
  - definição das linhas mestras da arquitectura (*baseline architecture*)
- Construção (*Construction*)
  - fabrico de todo o sistema
- Transição (*Transition*)
  - passagem do sistema aos seus utilizadores

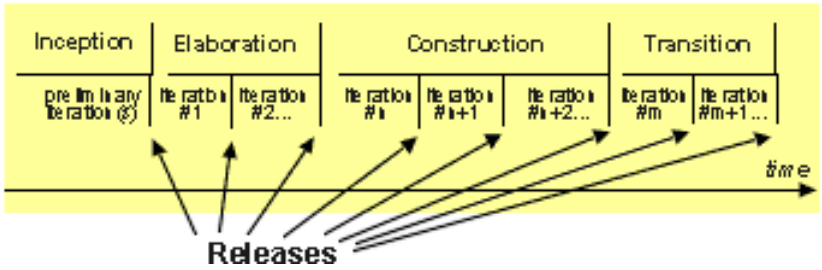
Fase e Iteração

- Cada fase pode ainda ser decomposta
  - em diversas iterações
- Cada iteração consiste
  - na realização das tarefas que conduzem a uma versão do sistema
    - *requirements*, *analysis & design*, *implementation*, *test*, *deployment*
  - num incremento de uma anterior versão do sistema
    - um subconjunto da versão que representa o sistema final
- Cada iteração pode ser vista como
  - um pequeno projecto
  - concretizado num processo em cascata (*waterfall*)



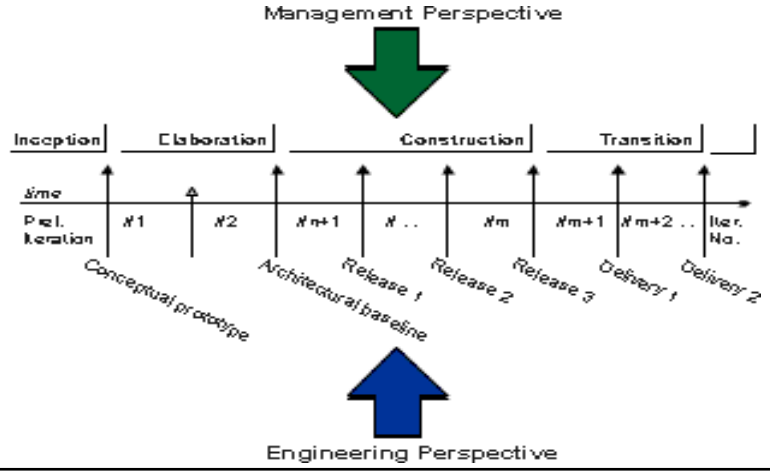
Versão

- Versão (ou entrega – *release*) do sistema, consiste numa
  - realização executável e estável do produto
  - incluindo também todos os elementos necessários à sua utilização
- Cada versão pode ser
  - interna
    - usada pelas equipas de desenvolvimento como parte de um marco
    - usada para efeitos de demonstração a clientes ou utilizadores
  - externa
    - enviada aos utilizadores finais
    - não é necessariamente o produto final, mas um passo nesse sentido
- Versão – "empurra" para o fim
  - combate síndroma,
  - 90% *done*; 90% *remaining* !



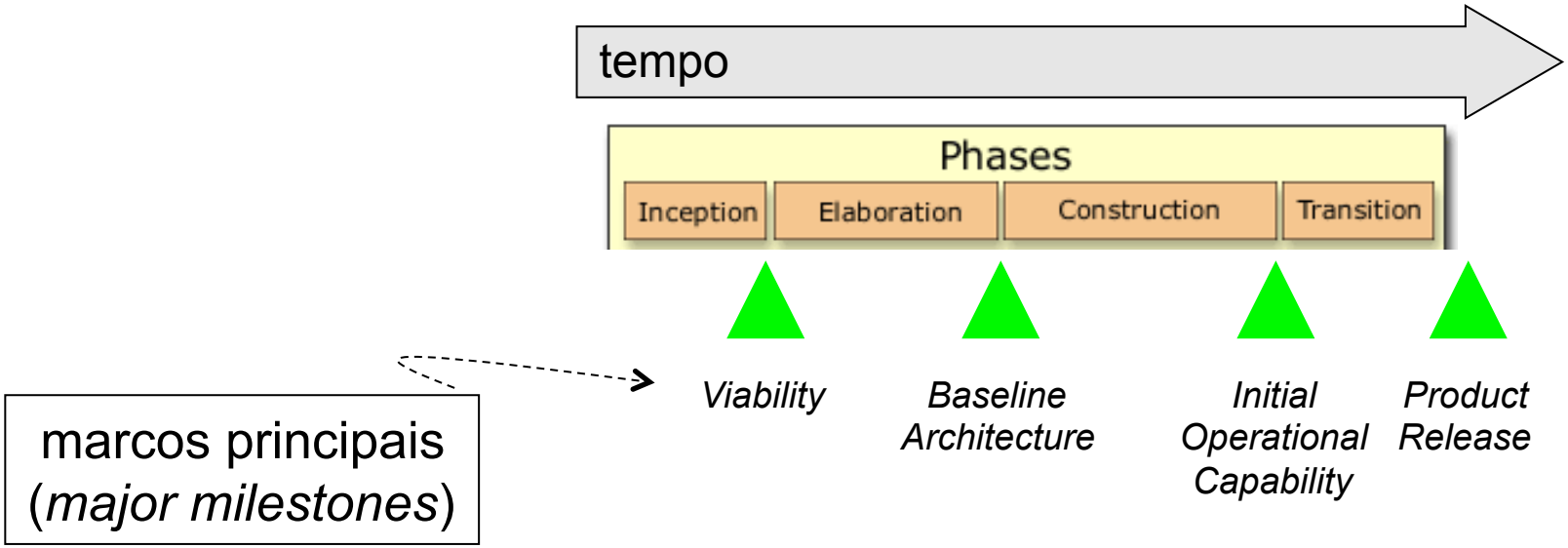
Fase, Iteração, Versão e Marco

- O Processo Unificado pode ser visto de duas perspectivas
  - gestão
    - existem 4 fases cada uma com um marco principal (*major milestone*)
    - de cada marco principal resulta uma versão
    - do último marco resulta o sistema final
  - engenharia
    - até cada marco principal existem marcos parciais (*minor milestone*)
    - atingir cada marco implica passar pelas diferentes tarefas
    - em cada tarefa, a ênfase técnica, depende da fase em que se está

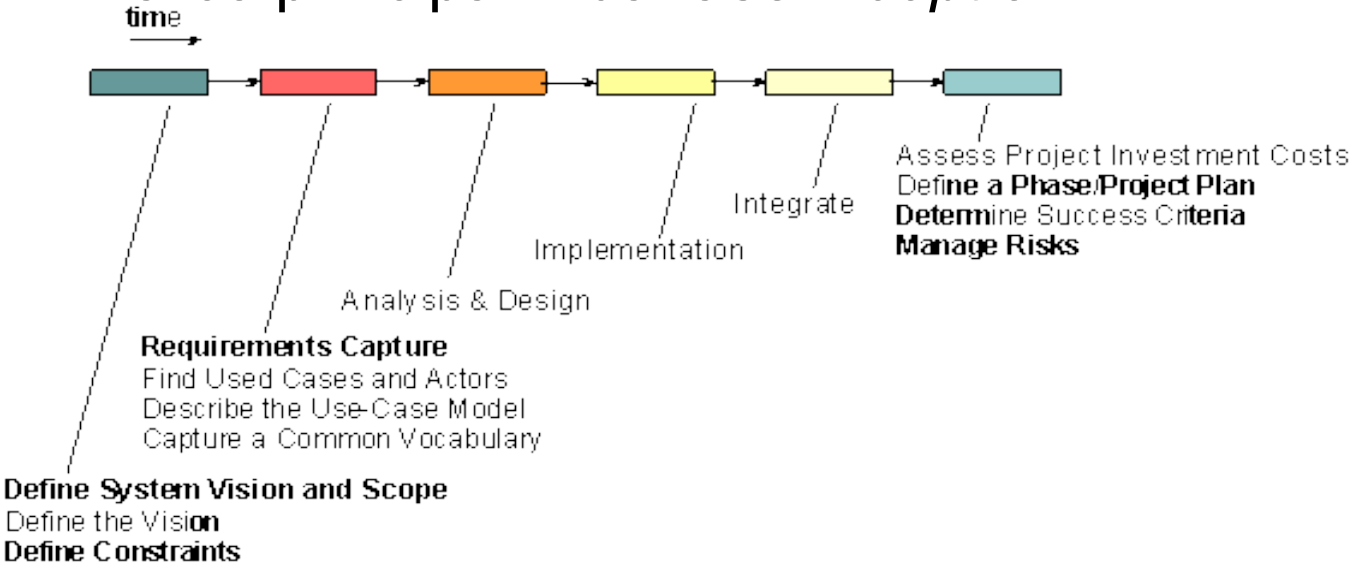


Marcos

- Um marco (*milestone*) num Processo
  - representa um ponto onde gestores e equipas avaliam o progresso
  - define os critérios de passagem para a próxima fase do Processo
- Cada fase pode ser visto como um mini-projecto
  - de cada mini-projecto resulta um incremento
  - ... no sentido do crescimento até se atingir uma versão do sistema

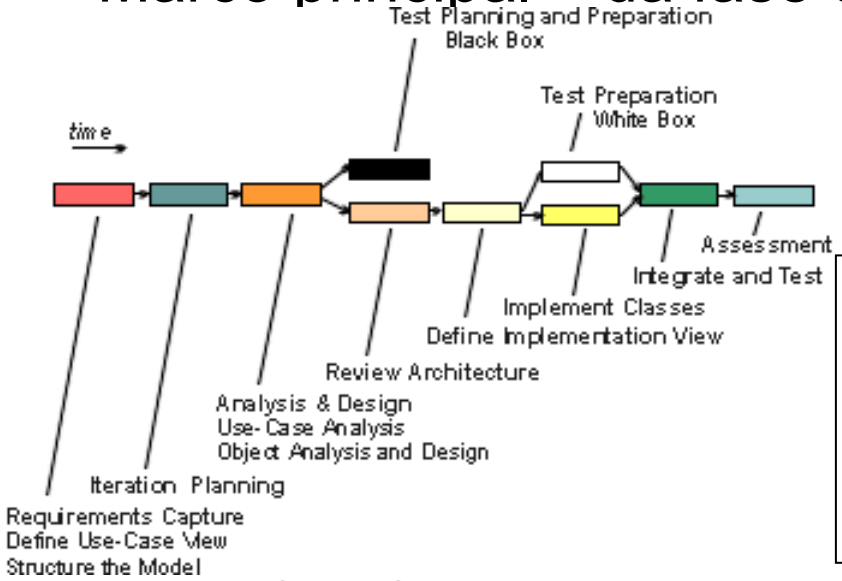


Marco principal – da fase *inception*



- Identificar <u>riscos</u> e construir <u>estimativas</u>
  - definir medidas para reduzir riscos à viabilidade do sistema
  - construir estimativa inicial de custo, esforço, escalonamento e qualidade
- Identificar <u>arquitectura candidata</u>
  - partindo dos casos de utilização críticos (5 a 10% do total)
  - efectuar as tarefas – análise, desenho, implementação
  - com o objectivo de encontrar e avaliar viabilidade de uma arquitectura

Marco principal – da fase *elaboration*

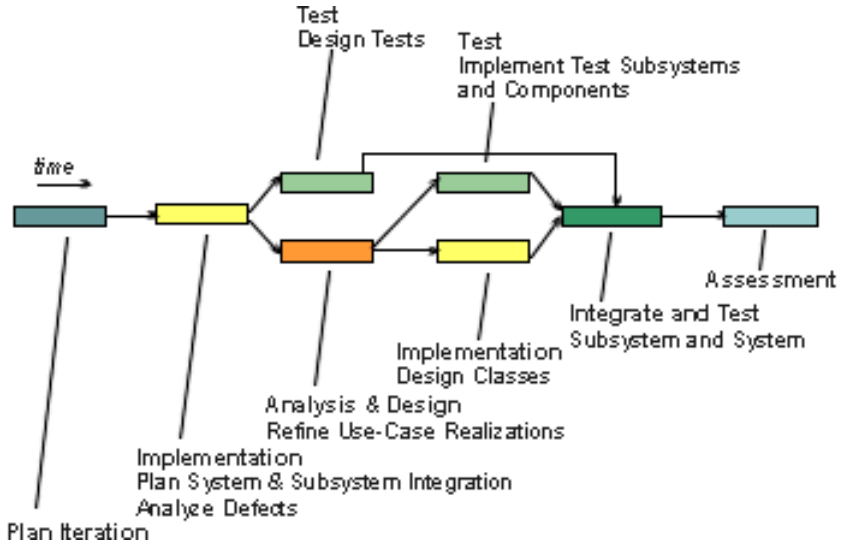


... the architecture baseline is a "small, skinny" system

... an executable architecture prototype is built in one or more iterations, depending on the scope, size, risk, and novelty of the project

- Especificar funcionalidade e detalhar estimativas
  - aprofundar casos de utilização até compreensão dos vários cenários
  - definir compromissos de orçamento – plano fino de custos
  - preparar plano de projecto para fase de *construction*
- Definir arquitectura
  - estender arquitectura candidata ao nível da estrutura de implementação
  - ter em conta os casos de utilização críticos, identificados na fase anterior

Marco principal – da fase *construction*

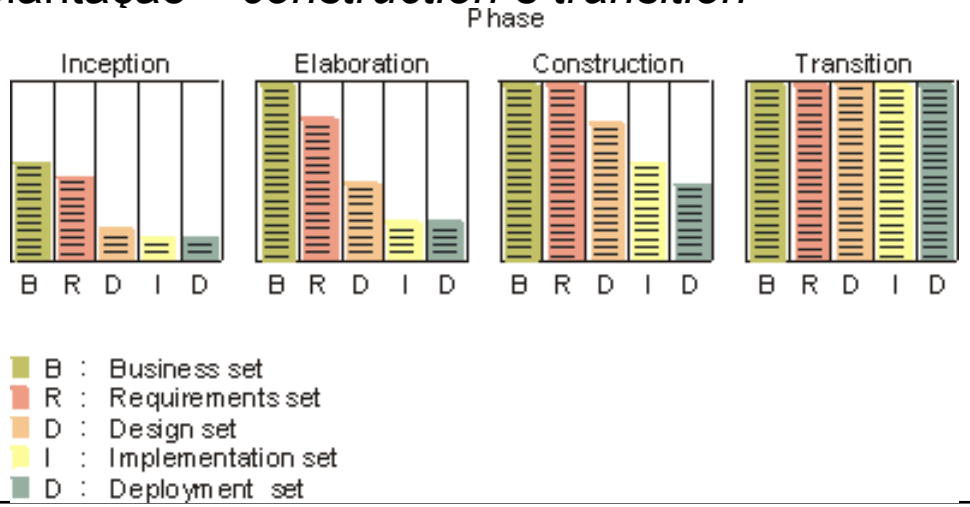


- Gerir recursos, controlar e optimizar trabalho
  - concretizar cada iteração como mais um incremento à anterior
  - no fim deve aparecer uma versão *beta* do sistema
    - versão operacional mas que não deve expor demasiado o projecto
- ... toda a construção assenta na arquitectura já definida
  - em princípio não devem "aparecer novos" subsistemas
  - poderá ser actualizada no detalhe para reflectir a experiência desta fase

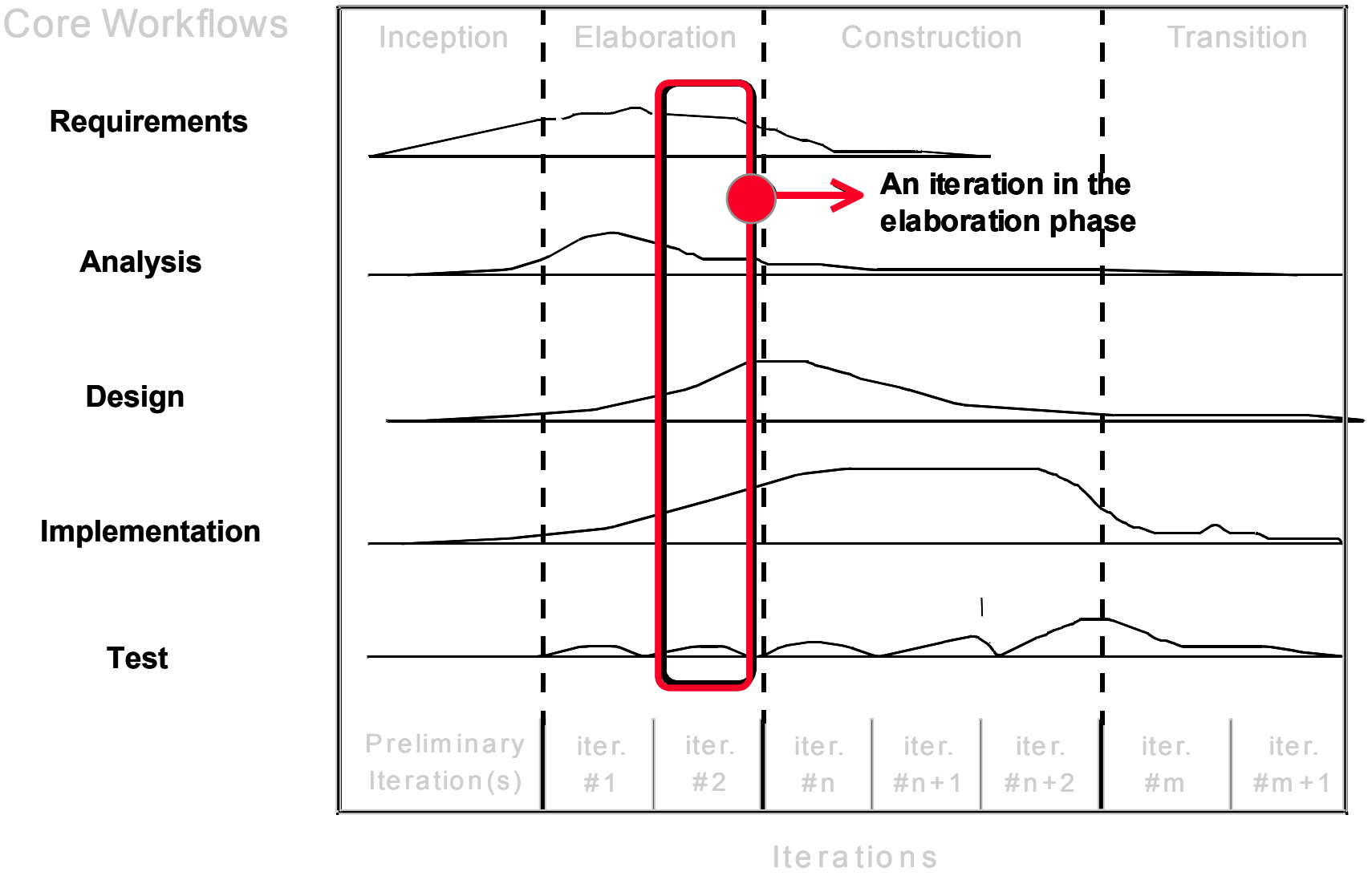
Paulo Trigo Silva

Iterações, Fases e Modelos

- Cada iteração acrescenta algo a cada modelo
  - ao "varrer" as diversas tarefas
    - identificação de requisitos; análise; desenho; implementação; teste
- Cada fase dedica diferente ênfase a cada modelo
  - modelo de casos de utilização – *inception* e *elaboration*
  - modelo de análise e desenho – *elaboration* e *construction*
  - modelo de implementação e teste – *construction*
  - modelo de implantação – *construction* e *transition*

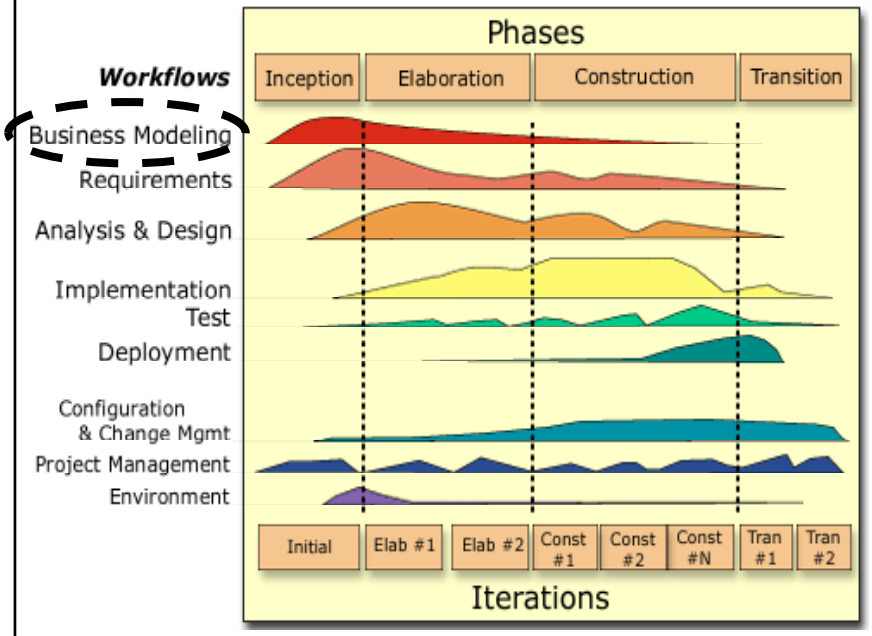


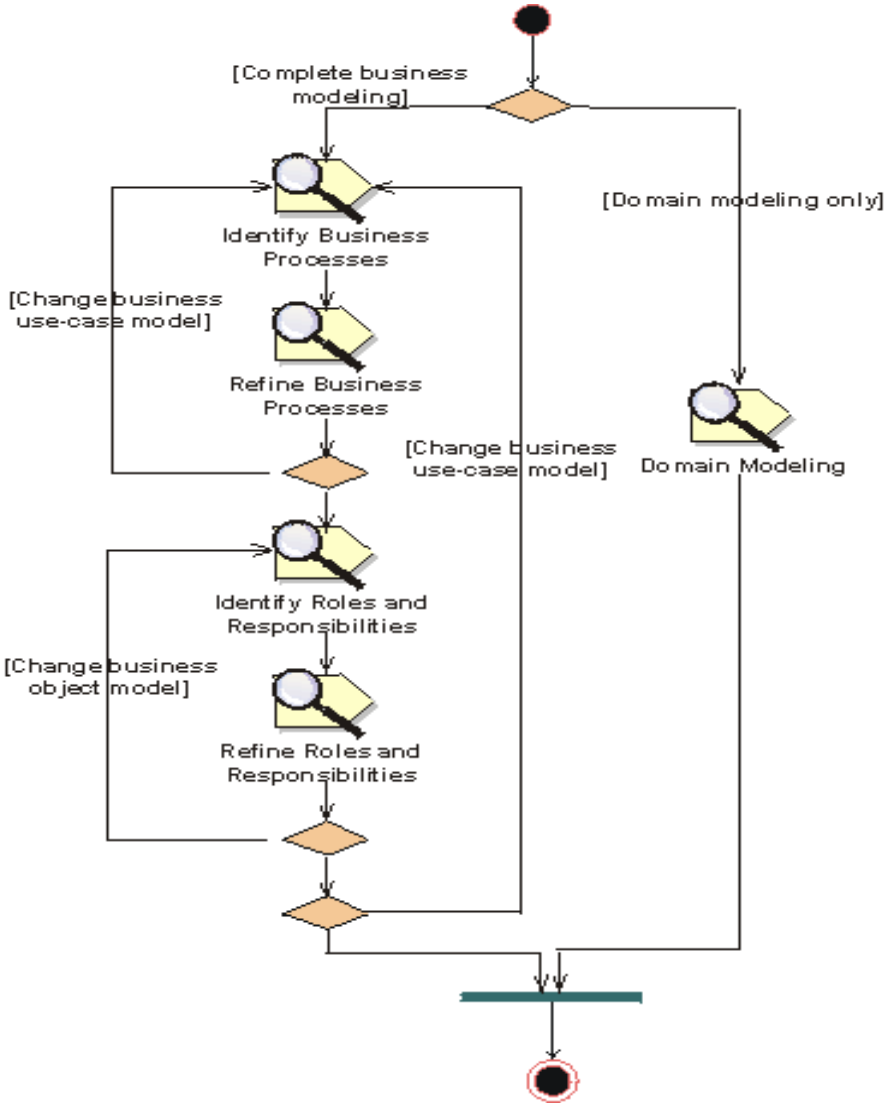
Iterations and Workflows



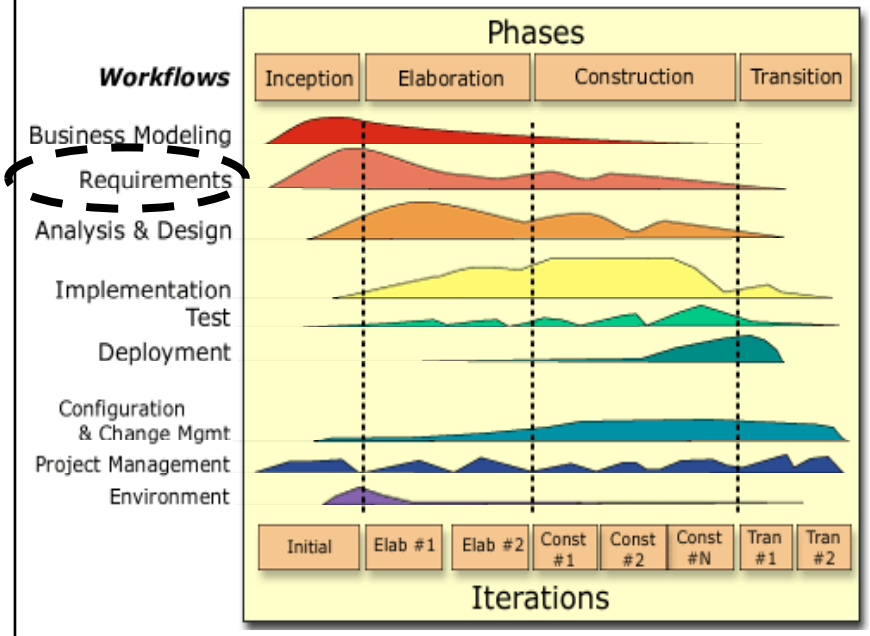
Paulo Trigo Silva

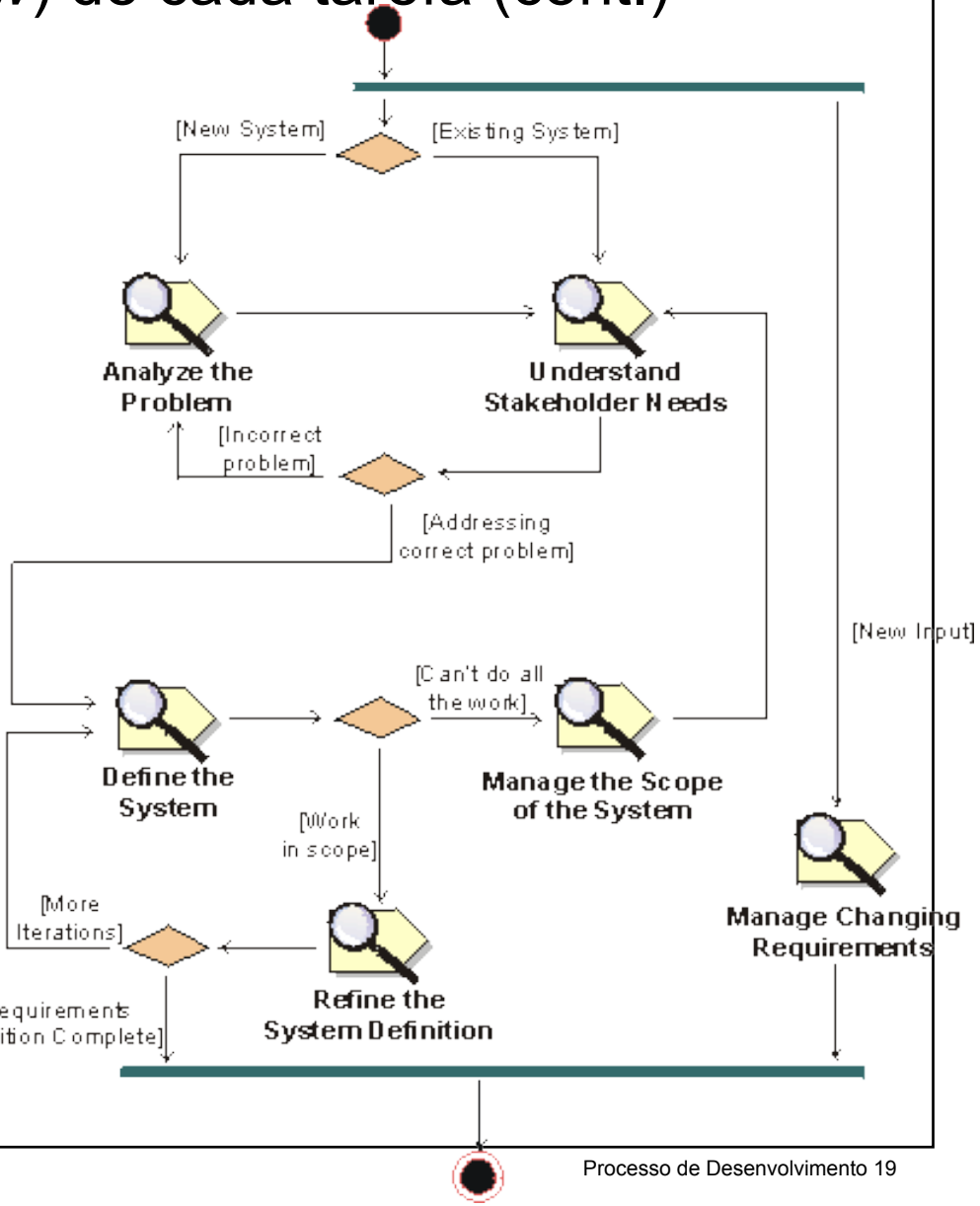
Fluxo de trabalho (*workflow*) de cada tarefa



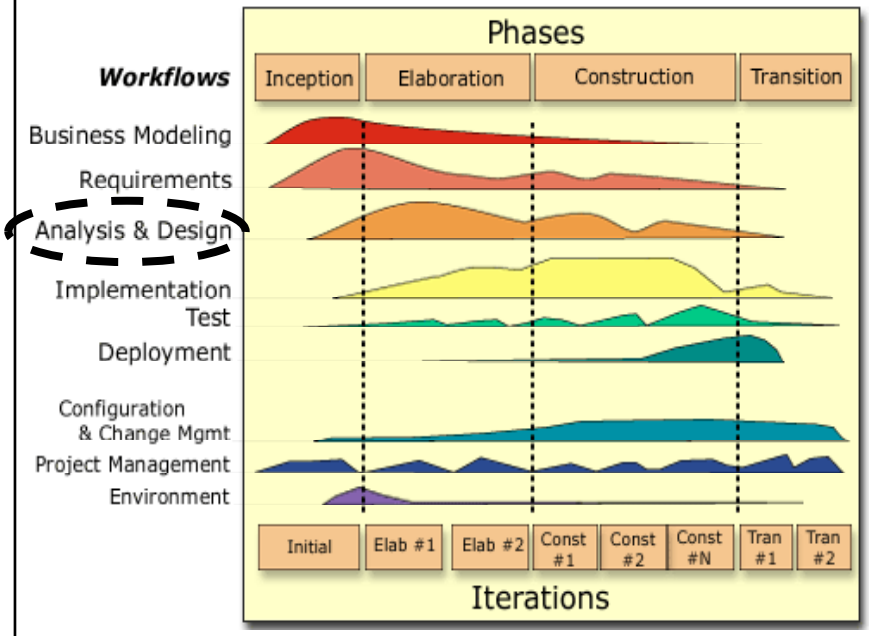


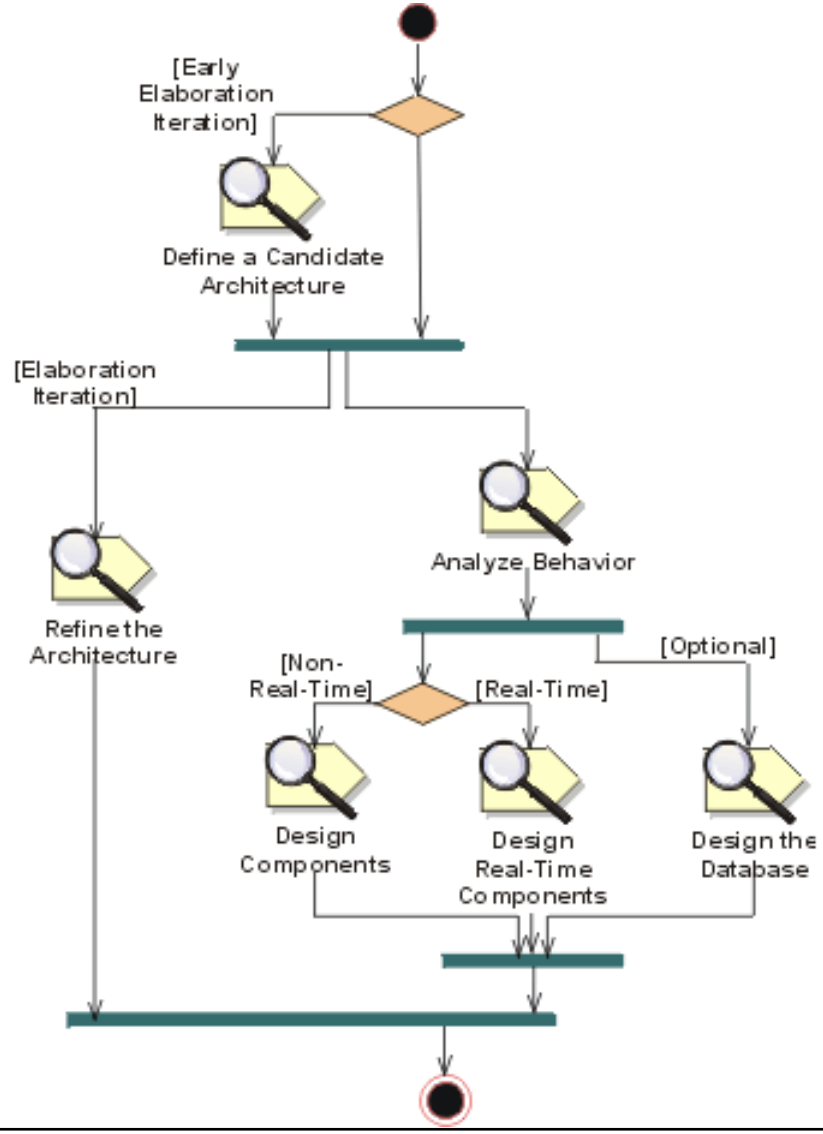
Fluxo de trabalho (*workflow*) de cada tarefa (cont.)



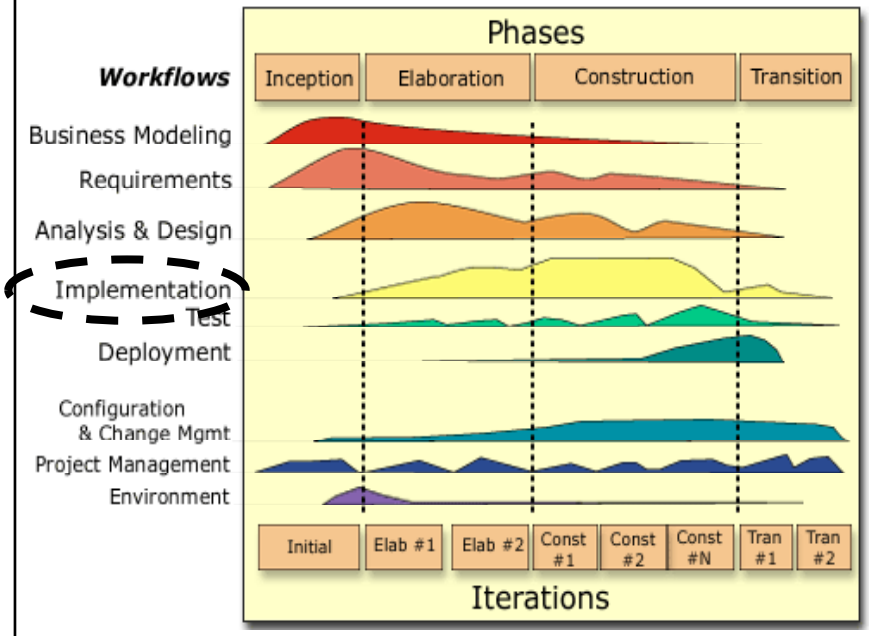


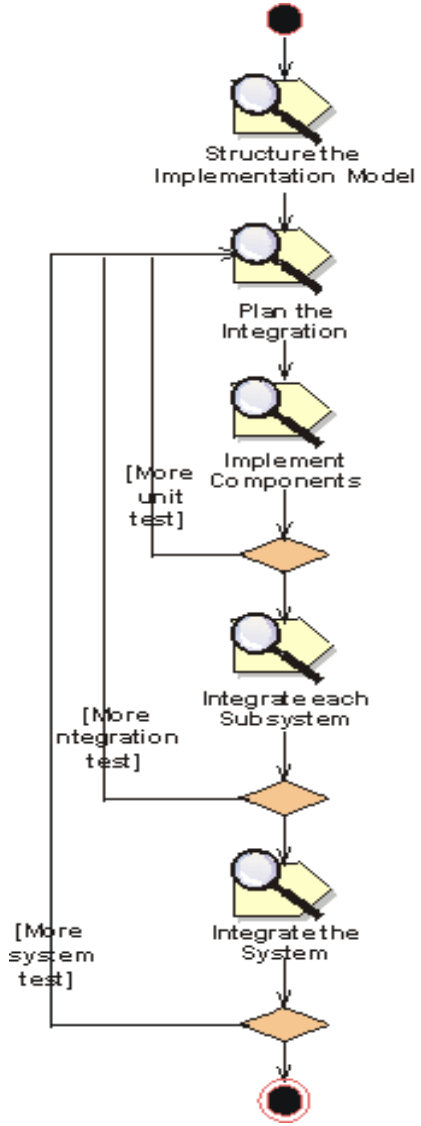
Fluxo de trabalho (*workflow*) de cada tarefa (cont. 1)



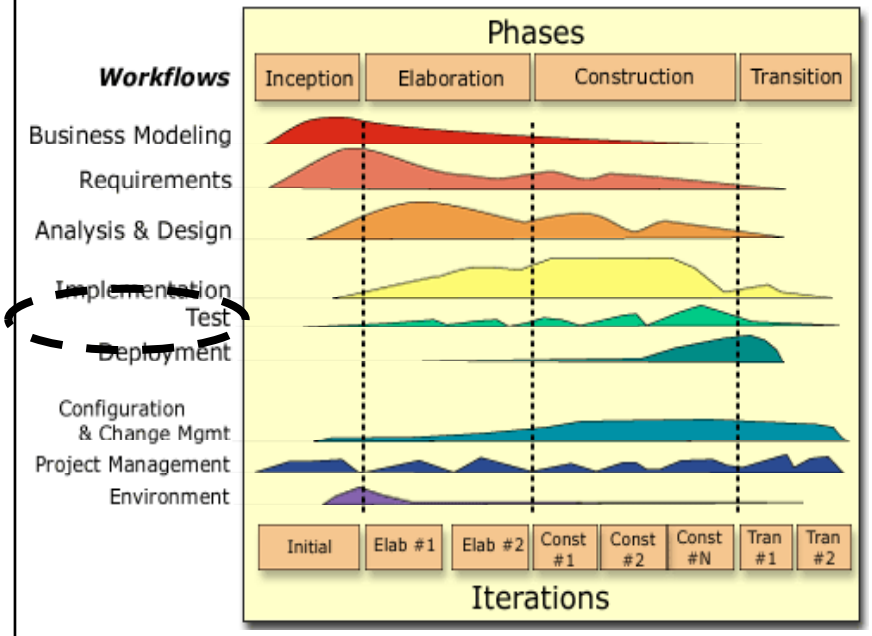


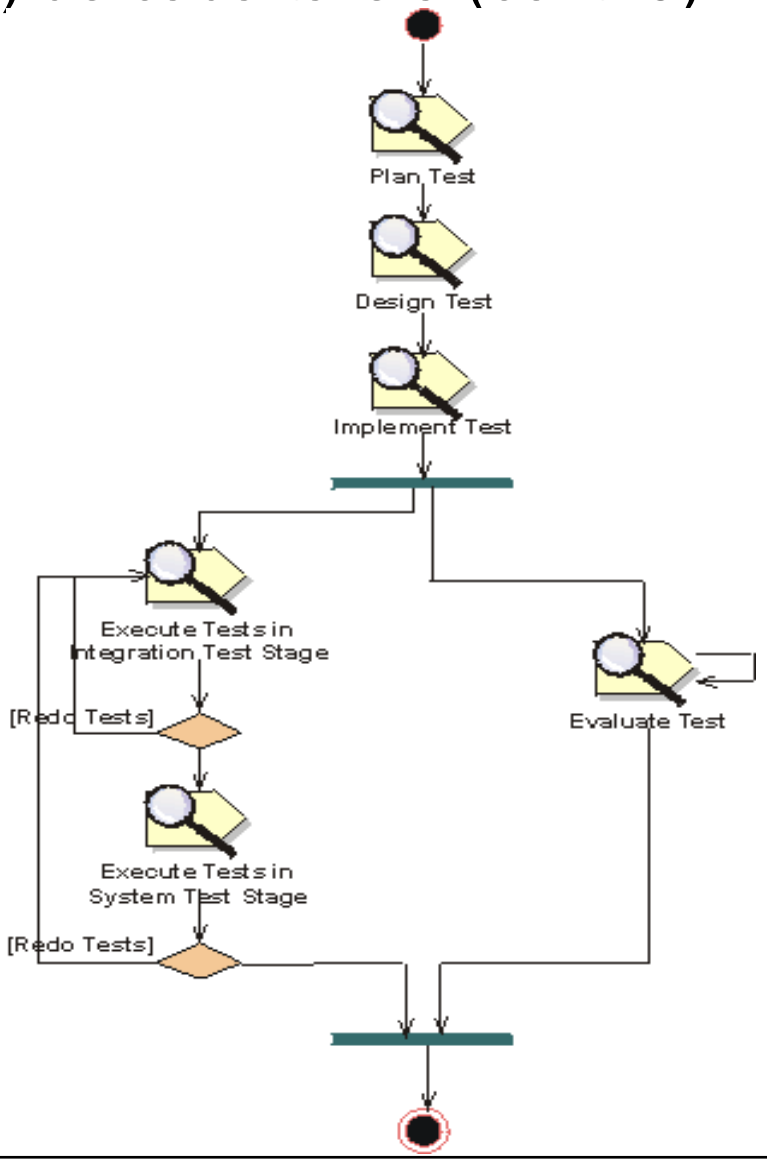
Fluxo de trabalho (*workflow*) de cada tarefa (cont. 2)



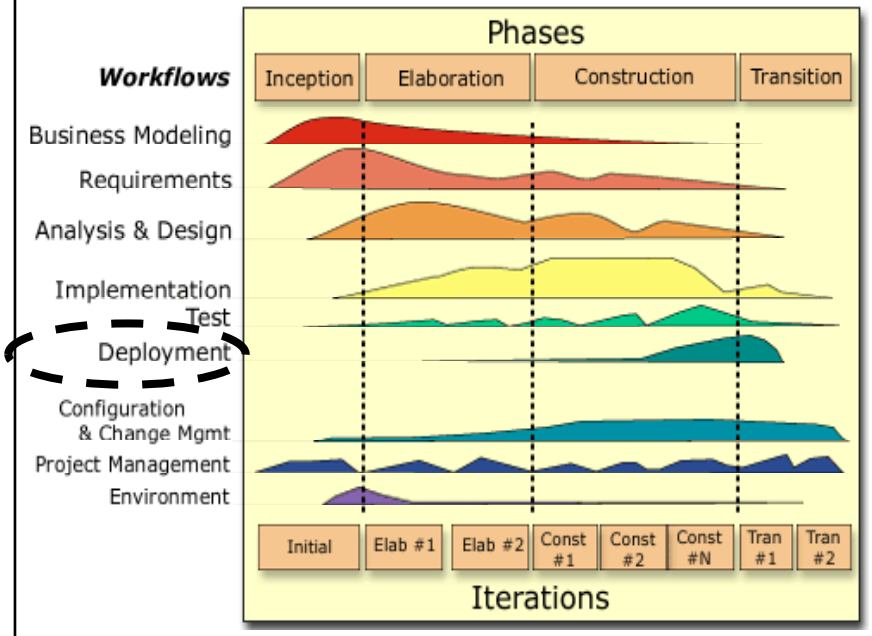


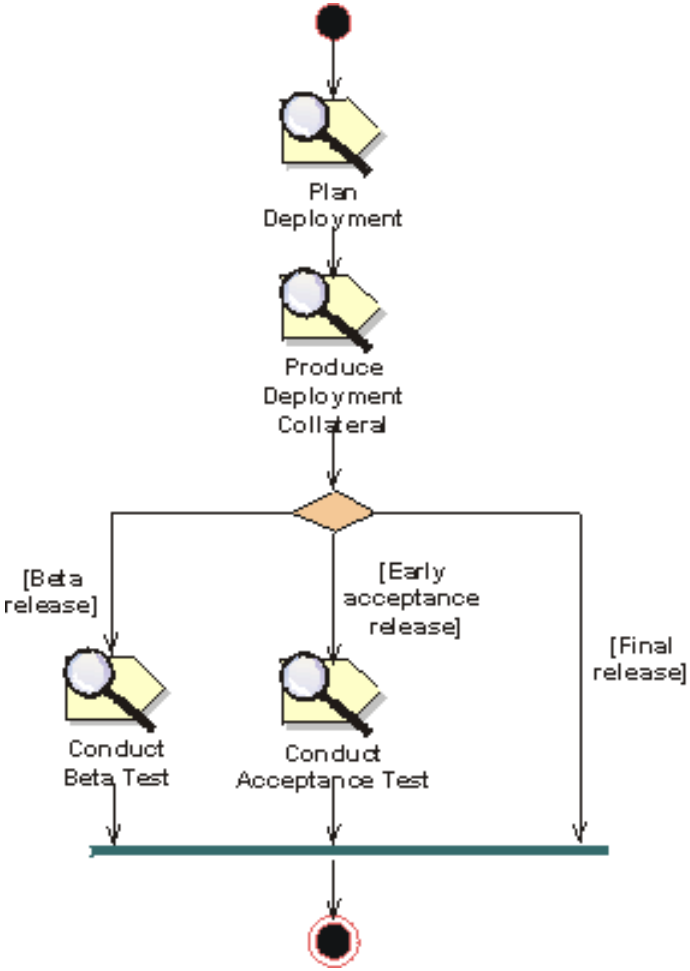
Fluxo de trabalho (*workflow*) de cada tarefa (cont. 3)



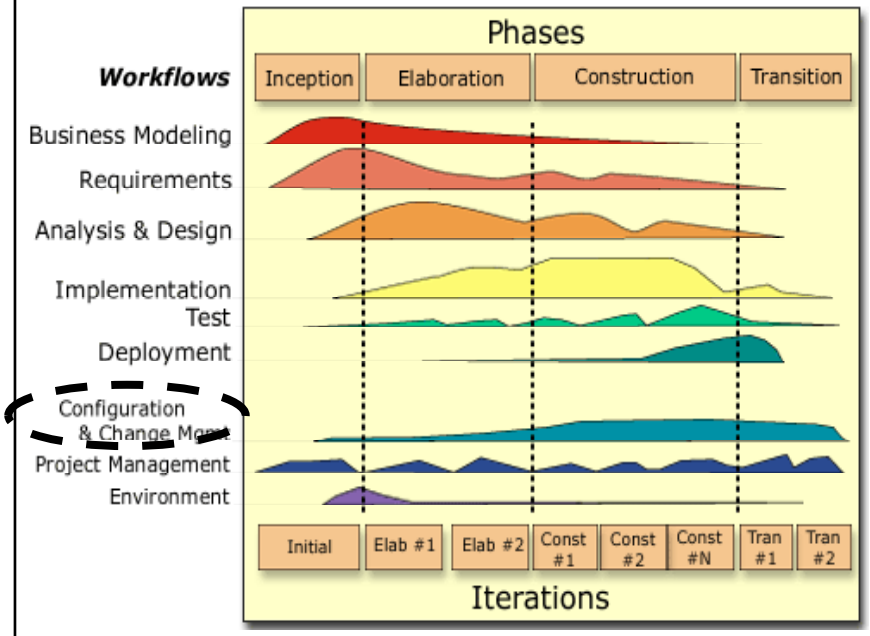


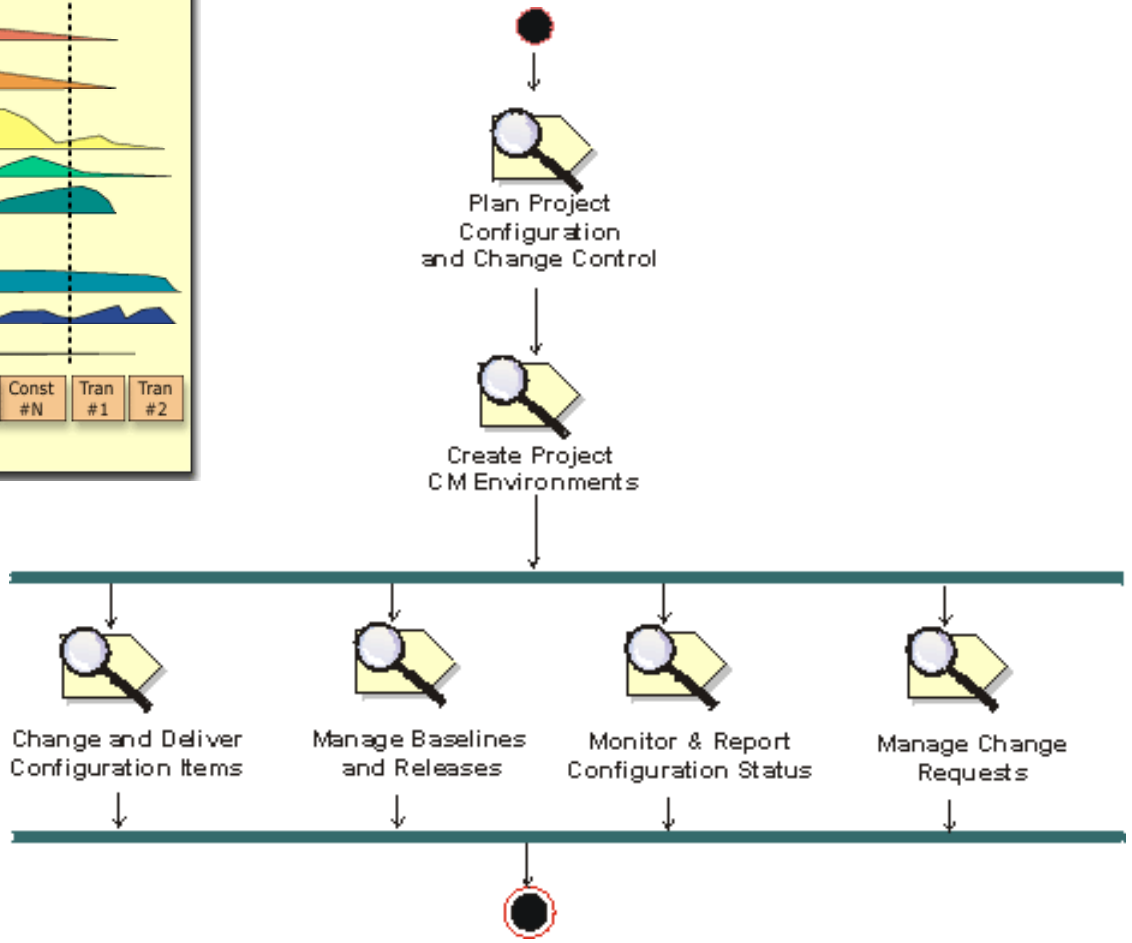
Fluxo de trabalho (*workflow*) de cada tarefa (cont. 4)



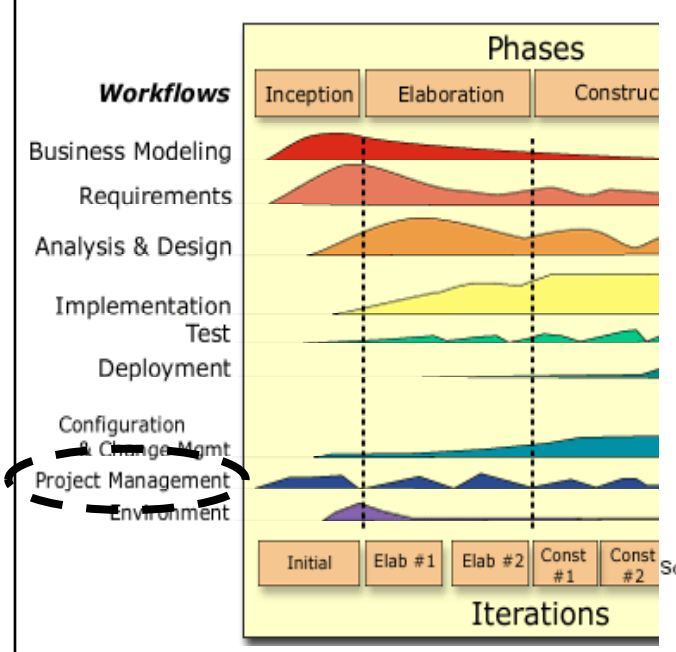


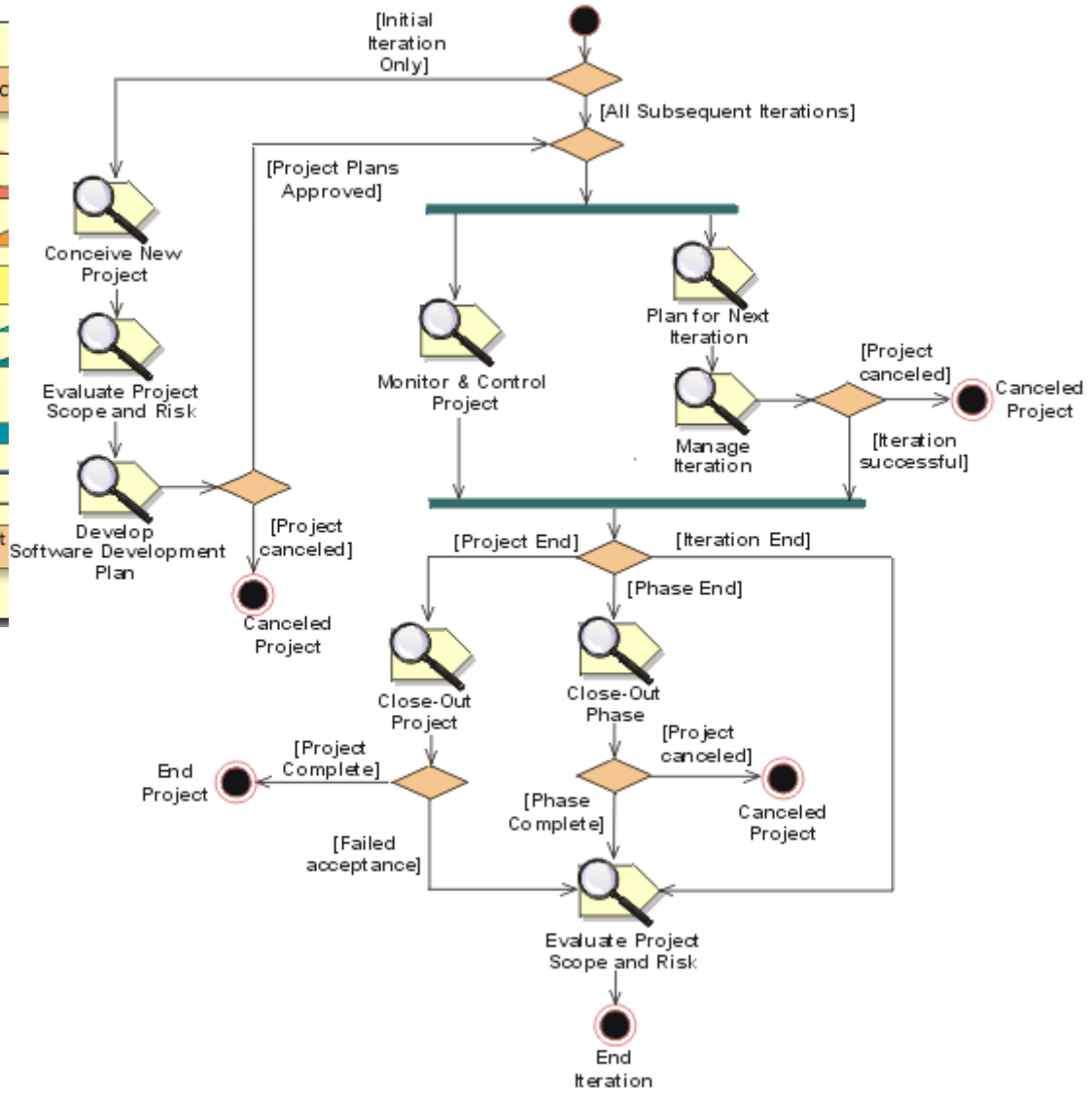
Fluxo de trabalho (*workflow*) de cada tarefa (cont. 5)



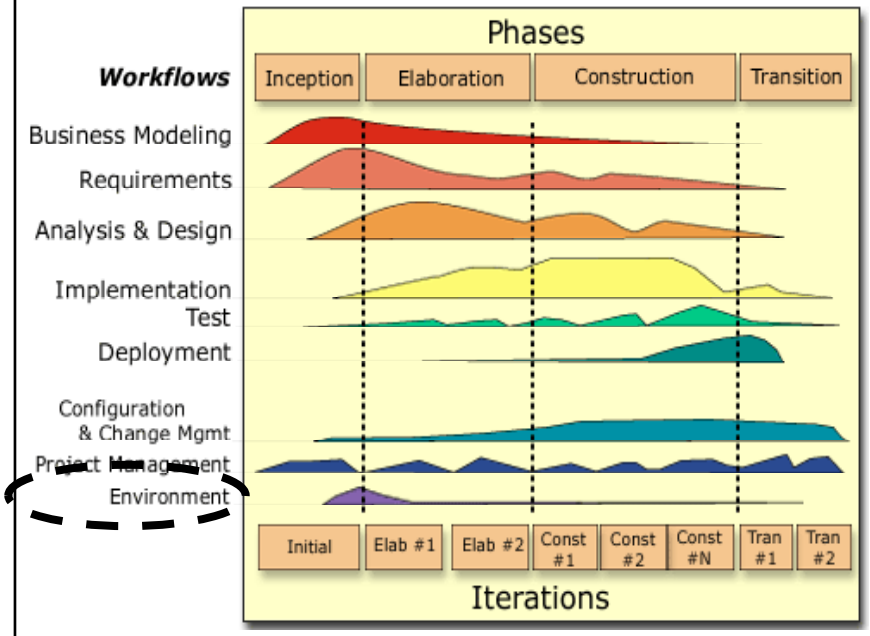


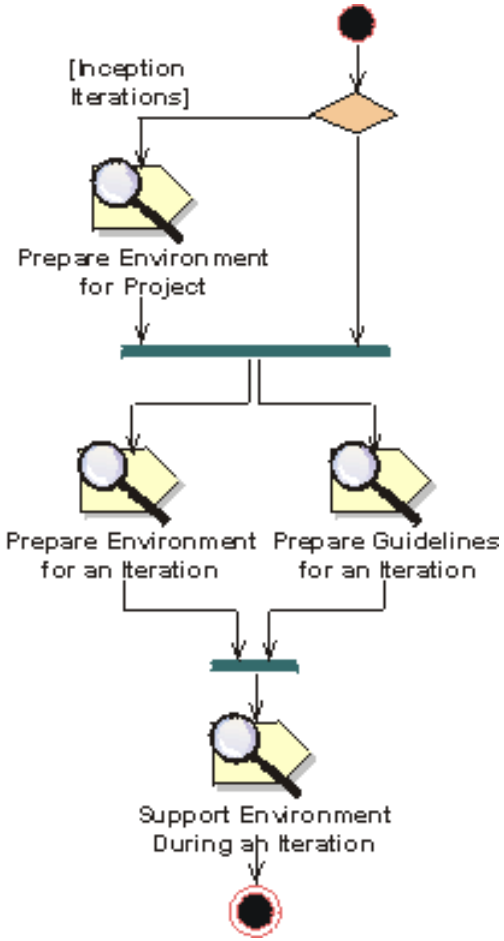
Fluxo de trabalho (*workflow*) de cada tarefa (cont. 6)



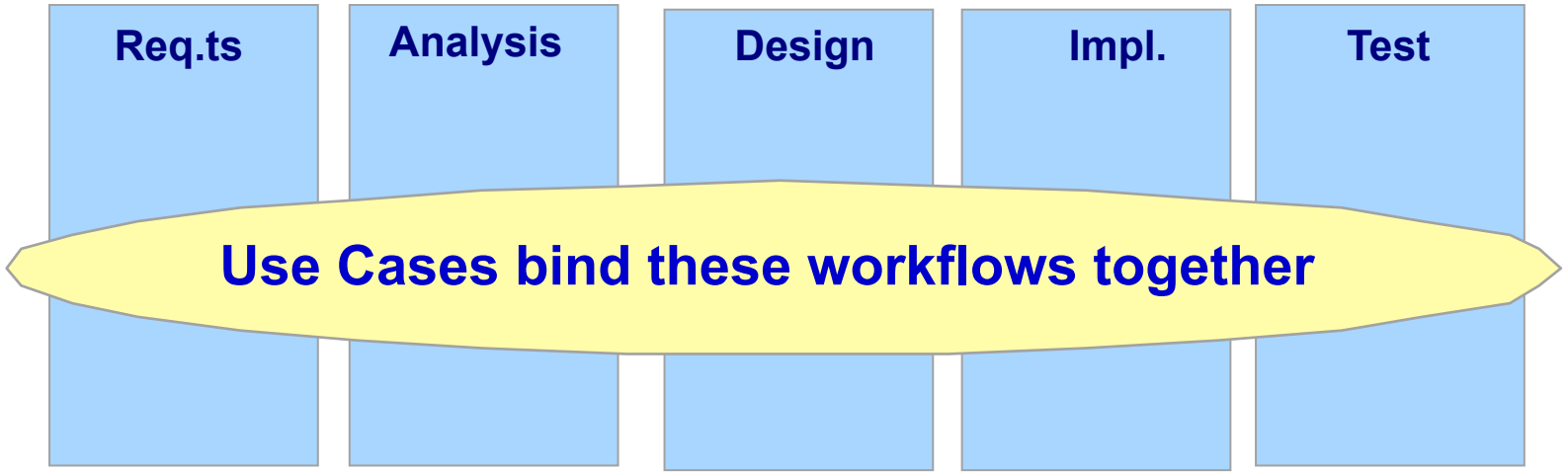


Fluxo de trabalho (*workflow*) de cada tarefa (cont. 6)





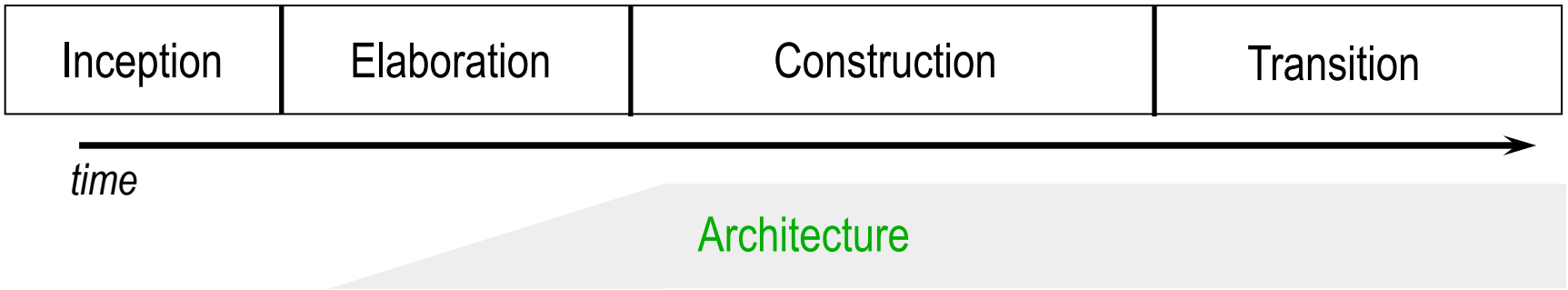
Processo Unificado – é conduzido por casos de utilização



- *Drive a number of development activities*
  - *Creation and validation of the system's architecture*
  - *Definition of test cases and procedures*
  - *Planning of iterations*
  - *Creation of user documentation*
  - *Deployment of system*
- *Synchronize the content of different models*

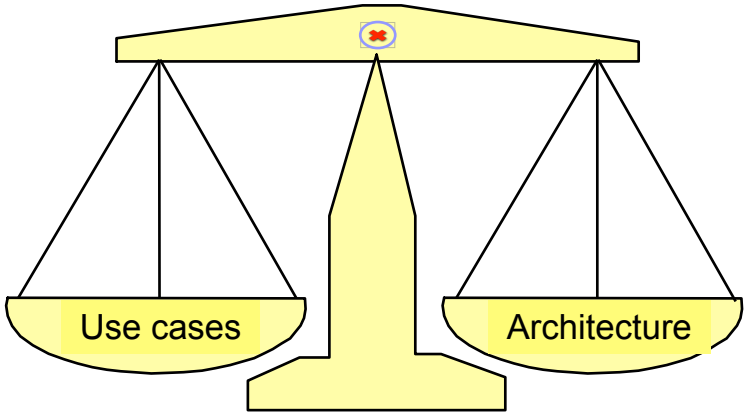
Paulo Trigo Silva

Processo Unificado – é centrado na arquitectura



- *Models are vehicles for*
  - *visualizing, specifying, constructing, and documenting architecture*
- *Synchronize the content of different models*

Function versus Form



- *use case specify function*
- *architecture specifies form*

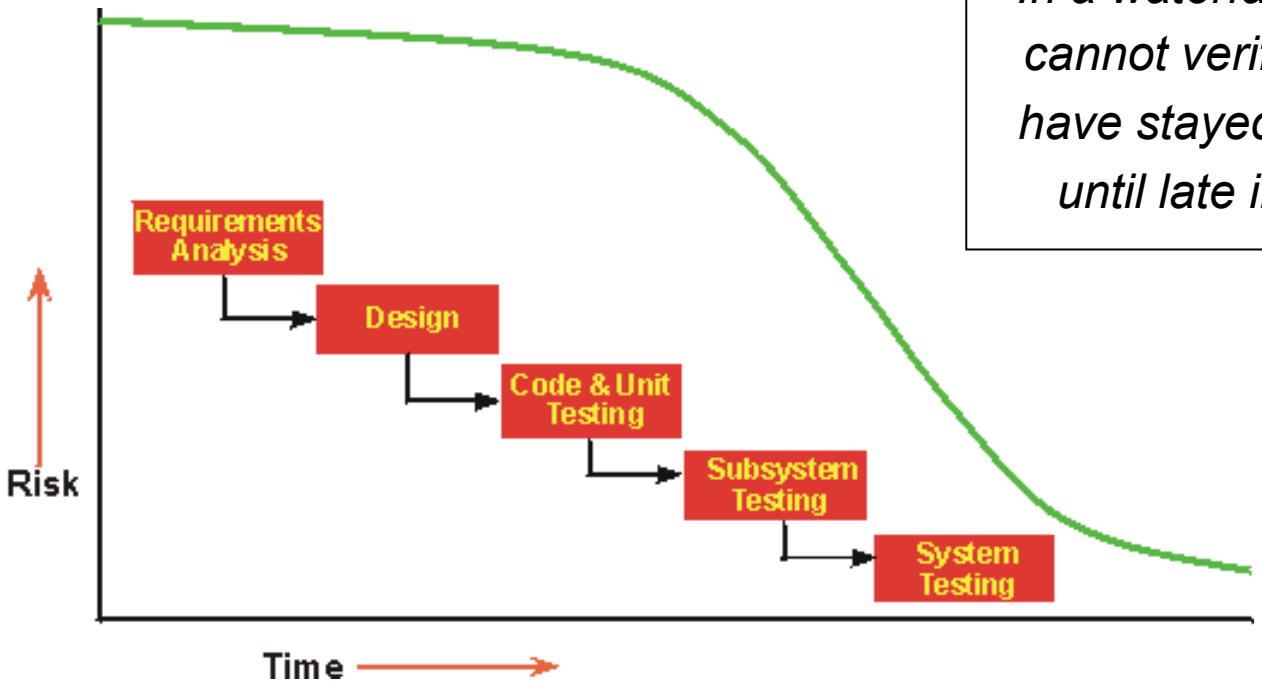
- *use cases and architecture must be balanced*

Paulo Trigo Silva

Porquê desenvolver em iterações ?

- Para identificar e controlar o risco o mais cedo possível
  - quanto mais cedo for identificado mais cedo se podem construir planos
  - riscos não previstos podem ocorrer apenas da integração do sistema

In a waterfall lifecycle, you cannot verify whether you have stayed clear of a risk until late in the lifecycle

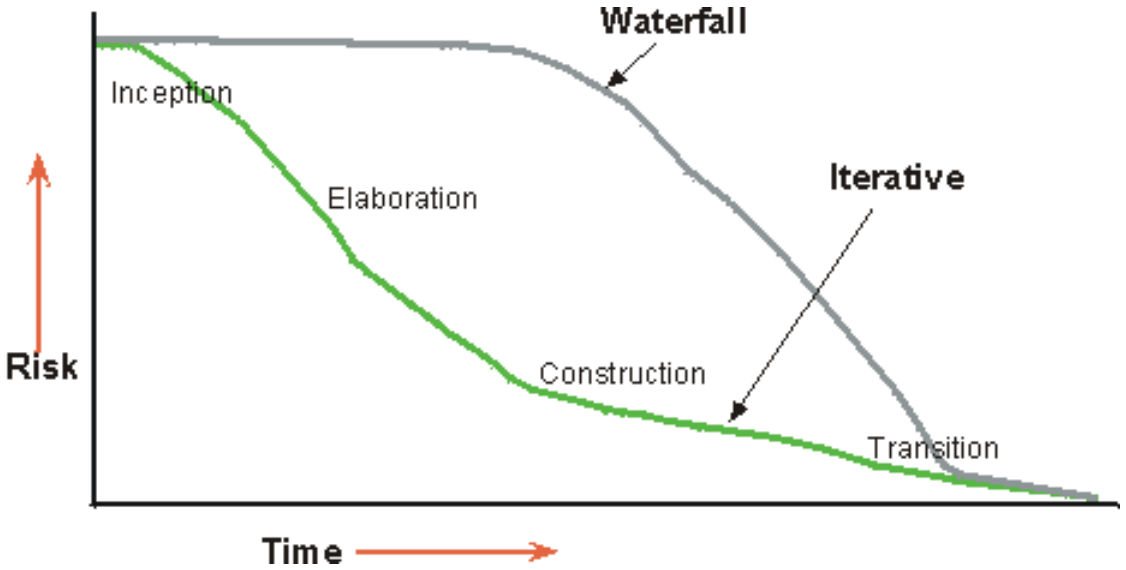


Controlo do risco num processo iterativo

- Por mais experiente que a equipa seja, nunca será possível,
  - prever, logo de início todos os potenciais riscos do projecto ...

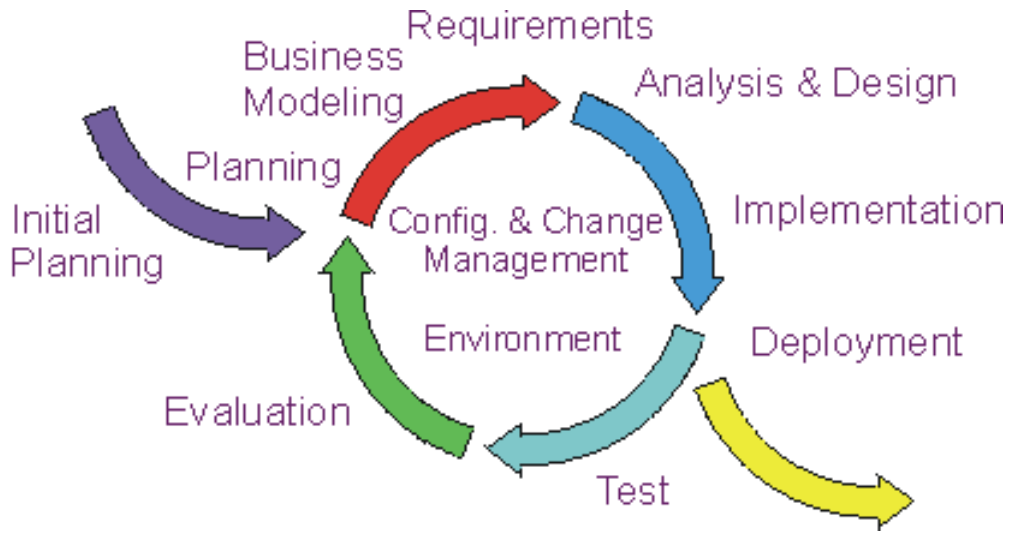
In an iterative lifecycle, you will select what increment to develop in an iteration based on a list of key risks.

Since the iteration produces a tested executable, you will be able to verify whether you have mitigated the targeted risks or not



Boas práticas de engenharia de *software*

- Processo Unificado congrega algumas das
  - boas práticas (*best practices*) da engenharia de *software*
- Desenvolver iterativamente (*develop iteratively*)

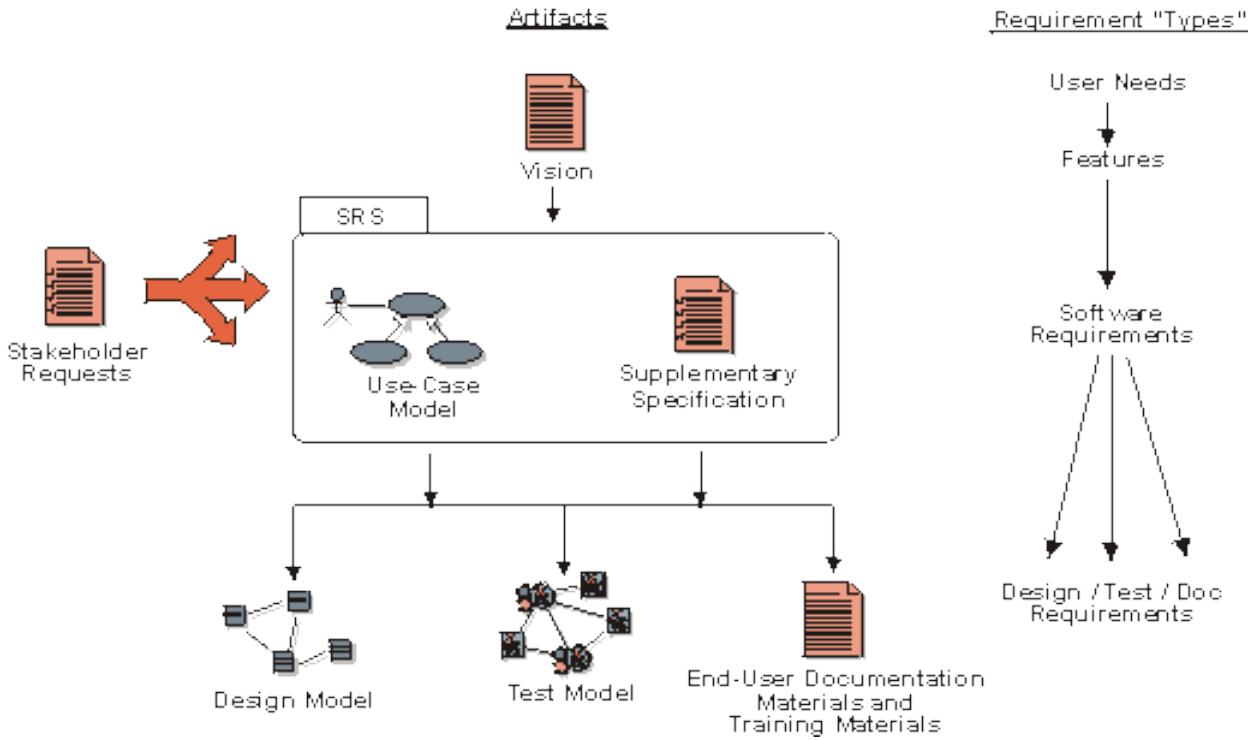


To mitigate risks, develop incrementally in an iterative fashion.

Each iteration results in an executable release

Boas práticas de engenharia de *software* (cont.)

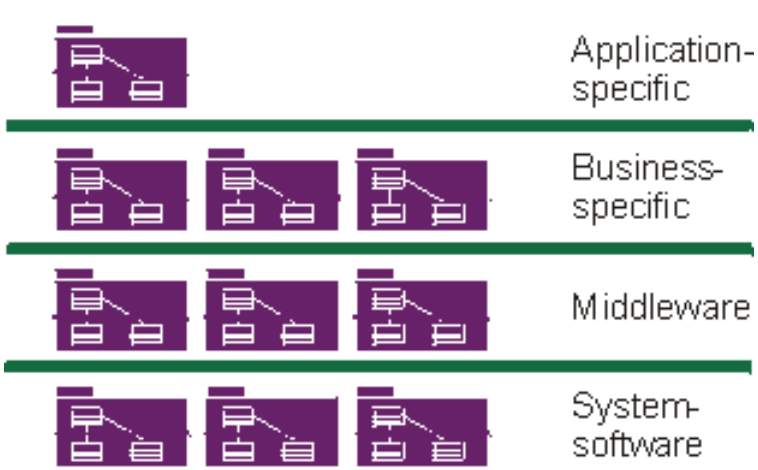
- Gerir Requisitos (*manage requirements*)



We may want to keep track of ambiguous "wishes", as well as formal requests, from our stakeholders to make sure we know how they are taken care of.

Boas práticas de engenharia de *software* (cont. 1)

- Arquitectura de componentes (*component architecture*)



subsistemas específicos para a aplicação

subsistemas reutilizáveis para diferentes aplicações do mesmo domínio

frameworks / class libraries

Component Subsystem:

realizes one or more interfaces,

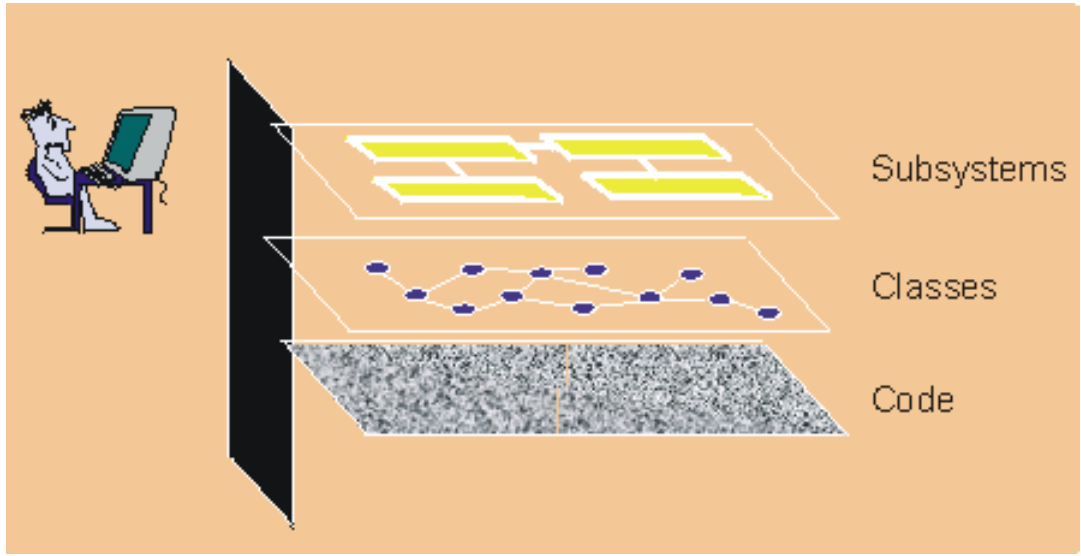
and may be dependent on one or more interfaces.

It may enclose zero or more classes, packages or other component subsystems,

none of which are visible externally (only interfaces are visible)

Boas práticas de engenharia de *software* (cont. 2)

- Modelação Visual (UML) (*model visually*)



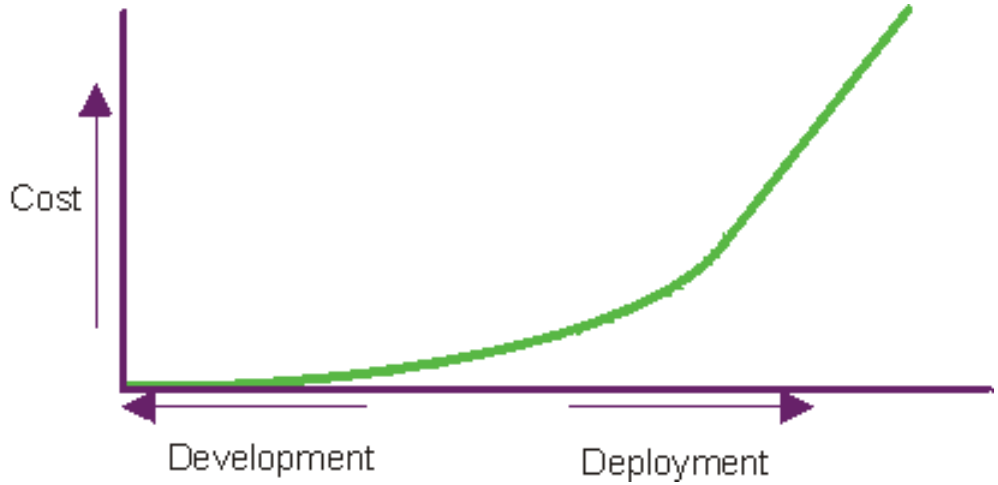
A model, correctly designed using object technology is:

Easy to understand. It clearly corresponds to reality.

Easy to modify. Changes in a particular phenomenon concern only the object that represents that phenomenon.

Boas práticas de engenharia de *software* (cont. 3)

- Verificação contínua de Qualidade (*verify quality*)

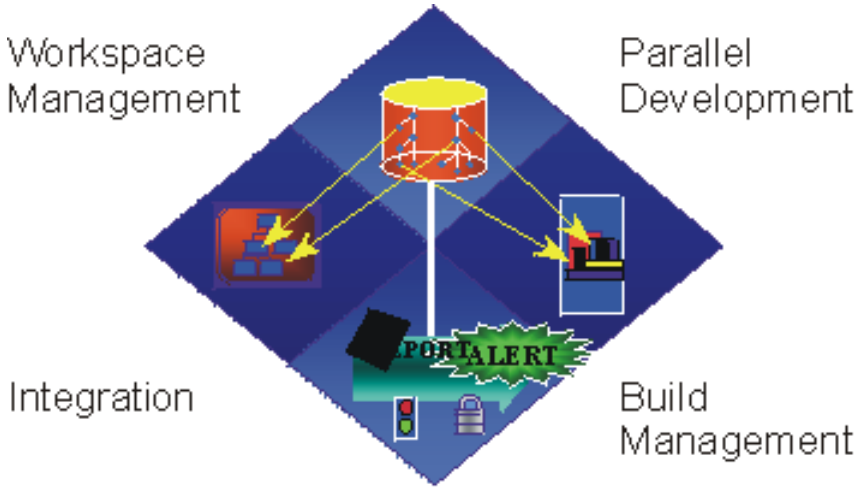


Software problems are 100 to 1000 times more costly to find and repair after deployment.

Verifying and managing quality throughout the project lifecycle is essential to achieving the right objectives at the right time.

Boas práticas de engenharia de *software* (cont. 4)

- Controlar Alterações (*control changes*)

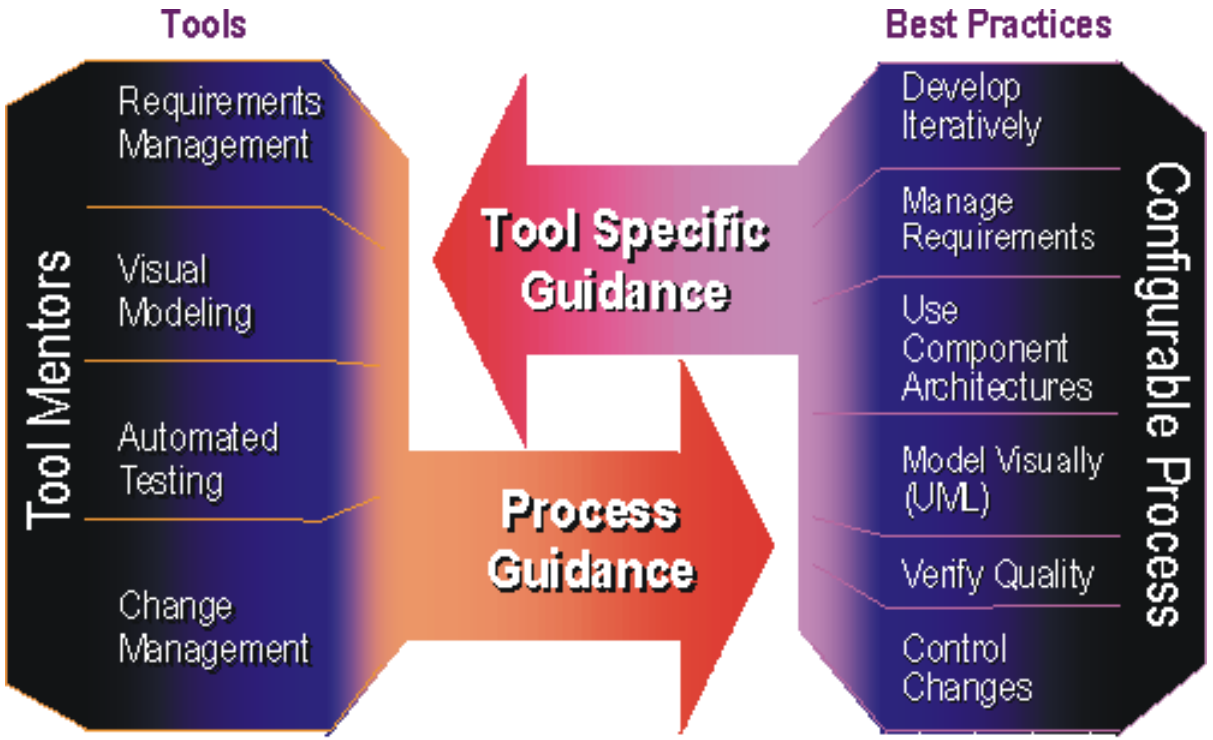


Controlling changes is more than just check-in and check-out of files.

It includes management of workspaces, parallel development, integration, as well as builds.

A key challenge when you are developing software-intensive systems is that you must cope with multiple developers organized into different teams, possibly at different sites, working together on multiple iterations, releases, products, and platforms.

Boas práticas de engenharia de *software* ... e o *RUP*



The Rational Unified Process shows how you can apply best practices of software engineering, and how you can use tools to automate your software engineering process