Engenharia de Software II

Engenharia de Software II

Carga Horária: 80 horas

Competência:

- Definir e construir a arquitetura de um sistema computacional baseado em padrões.

Engenharia de Sofware II

Indicadores:

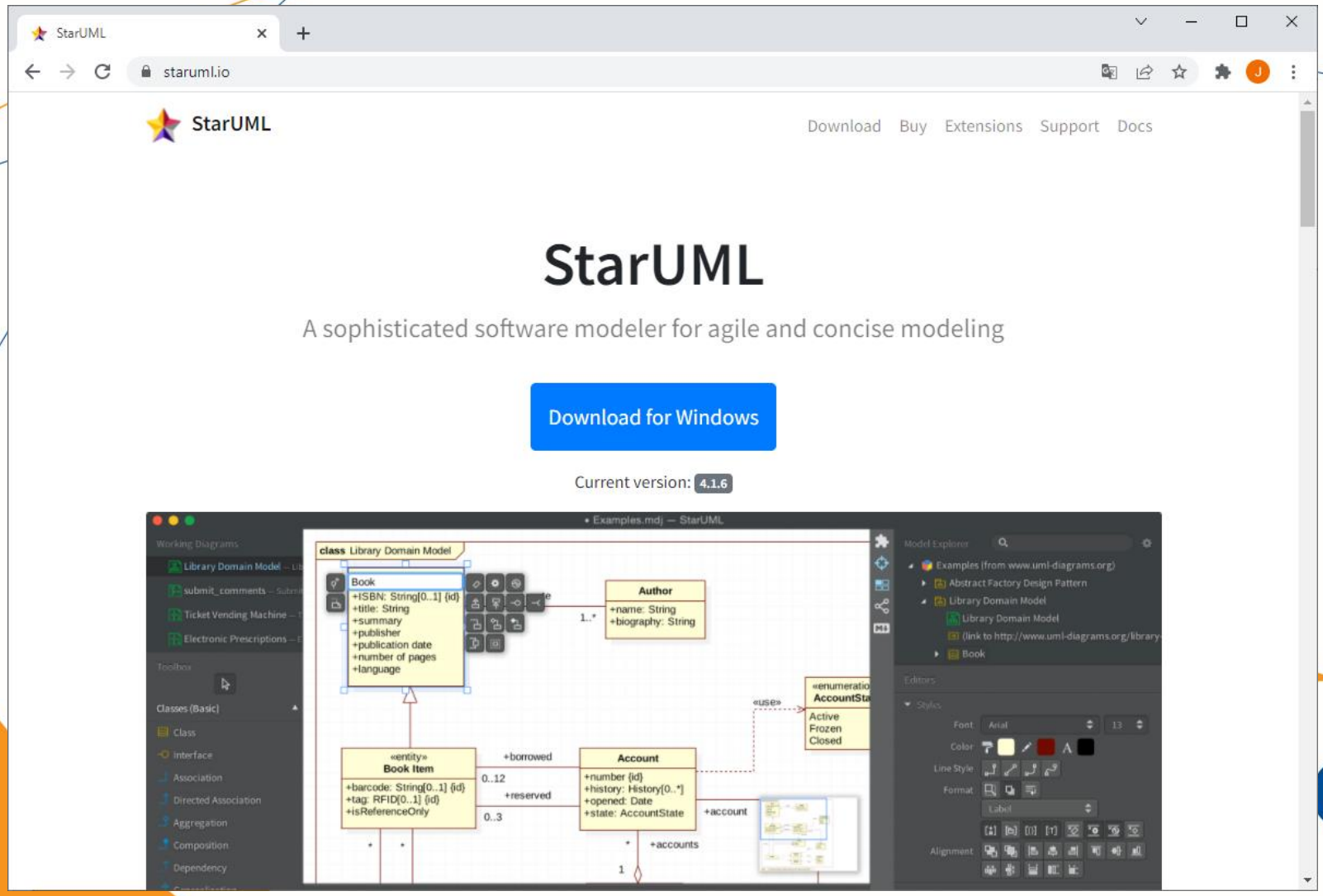
- Constrói modelos orientados a objetos utilizando notações e diagramas da UML.
- Define processo de gerência de configuração e mudança de softwares.
- Utiliza sistema de gerência de configuração executando suas principais funções de manipulação de código fonte.
- Identifica as principais diferenças/padrões entre os ambientes produtivos e não produtivos de software.
- Transforma o modelo de classes em um modelo relacional.

Engenharia de Software II

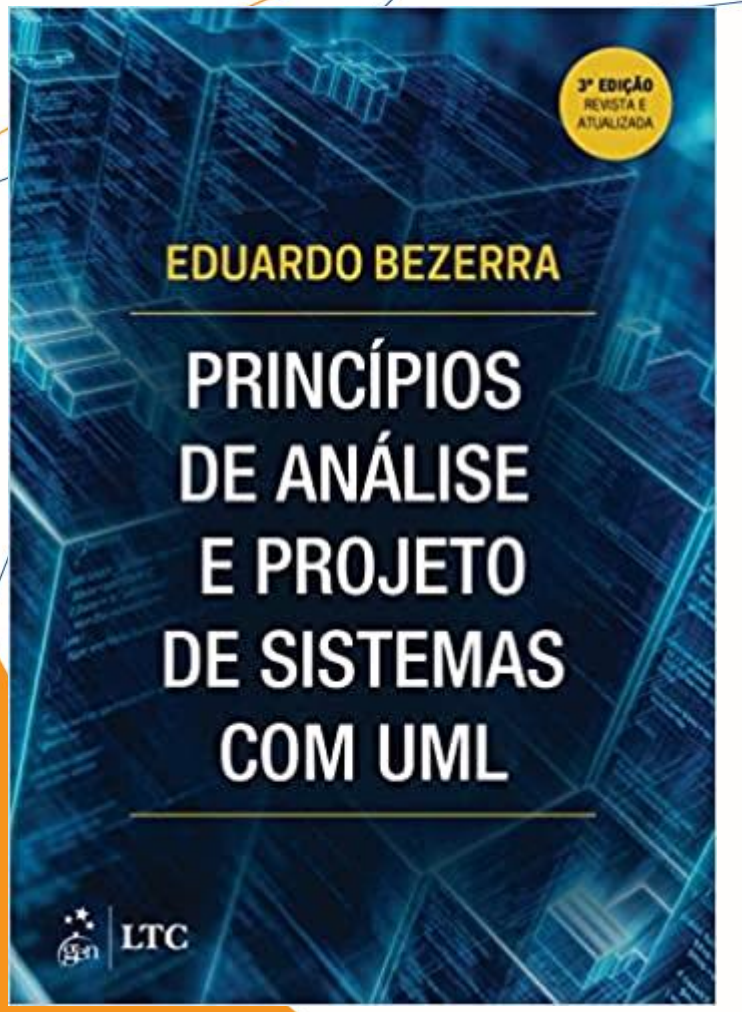
Bases Tecnológicas:

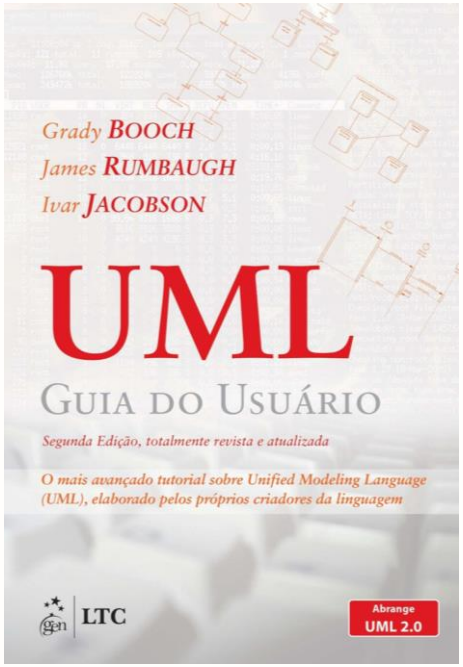
- Modelagem de sistemas;
- Projeto de arquitetura;
- Projeto de implementação;
- UML: diagrama de atividades;
- UML: diagrama de classes e pacotes;
- UML: diagrama de sequência;
- UML: diagrama de estado;
- UML: diagrama de componentes;
- UML: diagrama de implantação;
- Introdução a testes de software;
- Evolução do software;
- Gerência de Configuração de Software;
- Mapeamento objeto relacional. (ORM)

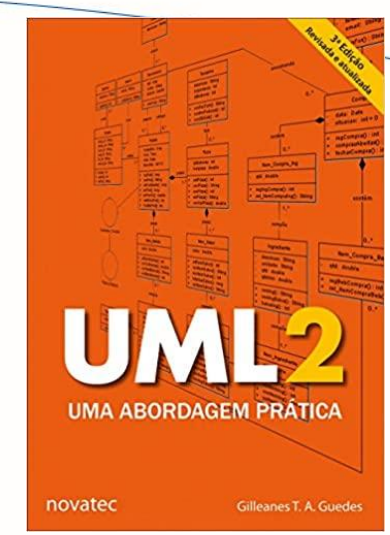
Software utilizado em sala de aula



Bibliografia







Senac

Site OMG com especificação da UML v 2.5

https://www.omg.org/spec/UML/

O que é Análise???

- Análise modela o problema e consiste das atividades necessárias para entender o domínio do problema

O que deve ser feito?

- Trata-se de uma atividade de investigação

O que é Análise???

- E/ou: a análise consiste de atividades feitas com e para o cliente (análise de requisitos)
- A informação produzida na análise deve ser discutida e aprovada pelo cliente
  - Invade-se um pouco o espaço da solução...
  - Interface do usuário, por exemplo

O que deve ser feito, de acordo com o cliente?

O que é Projeto???

- O projeto modela a solução e consiste das atividades de criação

Como pode ser feito?

- Trata-se de uma atividade de resolução

O que é Projeto???

- O Projeto inclui as atividades que resultam em informação que interessa apenas ao Programador.
- A atividade de projeto serve como base para a atividade de programação (construção)

Programador, veja como deve ser feito!

Análise e Projeto

Análise (Modelagem do problema)	Projeto (Modelagem da Solução)

Entender

Criar

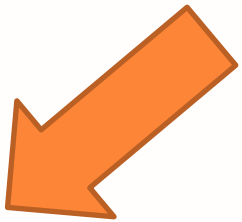
Análise e Projeto

Análise

(Informação importante para o cliente discutir e aprovar)

Projeto

(Informação importante para o programador)



Cliente

Programador

- Análise e projeto criam modelos

Análise Orientada a Objetos

A perspectiva empregada é de ***objetos***

- Coisas, conceitos, entidades... com *estado* e *responsabilidades*
- Ênfase em identificar descrever objetos ou conceitos do domínio do problema
- Num sistema para uma biblioteca, os conceitos são livro, biblioteca, usuário...

Representando modelos na análise

- Podemos criar modelos do domínio do problema
- UML é útil aqui:
  - Se os objetos forem do domínio do problema
  - Se os objetos não tiverem métodos

Processo de desenvolvimento de software

- Análise e projeto acontecem dentro de um processo
- Não um processo específico... e sim uma visão geral
- Quais as atividades do desenvolvedor em cada uma das fases do processo (análise, projeto, implementação e testes)?
- E que são artefatos?

Processo Unificado - UP

- A motivação para o uso da abordagem de Craig Larman ao Processo Unificado deve-se ao fato de que este é um processo bastante conciso e eficiente para análise e projeto de sistemas orientados a objetos.
- Neste método, cada artefato (documento ou diagrama) tem uma razão muito clara para existir e as conexões entre os diferentes artefatos são muito precisas.

UML

- Unified Modeling Language.
- Conhecer uma linguagem não implica na habilidade de saber usá-la para produzir artefatos úteis.
- Escrever bons projetos é como escrever poesia. Não basta conhecer a linguagem. É preciso dominar certas técnicas de escrita.

Software Deselegante

- O *software deselegante* é aquele software feito sem uma estrutura clara.
- O software deselegante é aquele do qual não se consegue reusar partes e que não se consegue entender como funciona sem uma boa carga de documentação (e muitas vezes nem assim).
- É também aquele no qual uma pequena modificação em uma de suas características pode causar um não funcionamento generalizado.

Software Deselegante

- O *software elegante* é o software cuja estrutura é intrinsecamente mais fácil de compreender, que é autodocumentado e pode ser compreendido em nível macro ou em detalhes.
- Ele é mais fácil de modificar: quando alguma de suas características é mudada, ele continua funcionando.

Soluções para prover elegância

- Design Patterns - lições aprendidas ao longo dos anos em diferentes projetos.

Atividades do Desenvolvimento

- Análise
- Projeto
- Implementação
- Teste

Análise

- A *análise* enfatiza a investigação do problema.
- O objetivo da análise é levar o analista a investigar e a descobrir.
- Para que esta etapa seja realizada em menos tempo e de forma mais precisa, deve-se ter um bom método de trabalho.

Análise

- Pode-se dizer que o resultado da análise é o enunciado do problema, e que o projeto será a sua resolução.
- Problemas mal enunciados podem até ser resolvidos, mas a solução não corresponderá às expectativas.

Análise

- A qualidade do processo de análise é importante porque um erro de concepção resolvido na fase de análise tem um custo; na fase de projeto tem um custo maior; na fase de implementação maior ainda, e na fase de implantação do sistema tem um custo relativamente astronômico.

Projeto

- A fase de *projeto* enfatiza a proposta de uma solução que atenda os requisitos da análise.
- Então, se a analise é uma investigação para tentar descobrir o que o cliente quer, o projeto consiste em propor uma solução com base no conhecimento adquirido na análise.

Implementação

- A utilização de técnicas sistemáticas nas fases de análise e projeto faz com que o processo de geração de código possa ser automatizado.
- Neste caso, cabe ao programador dominar as características específicas das linguagens, ferramentas, *frameworks* e estruturas de dados para adaptar o código gerado aos requisitos indicados quando necessário.

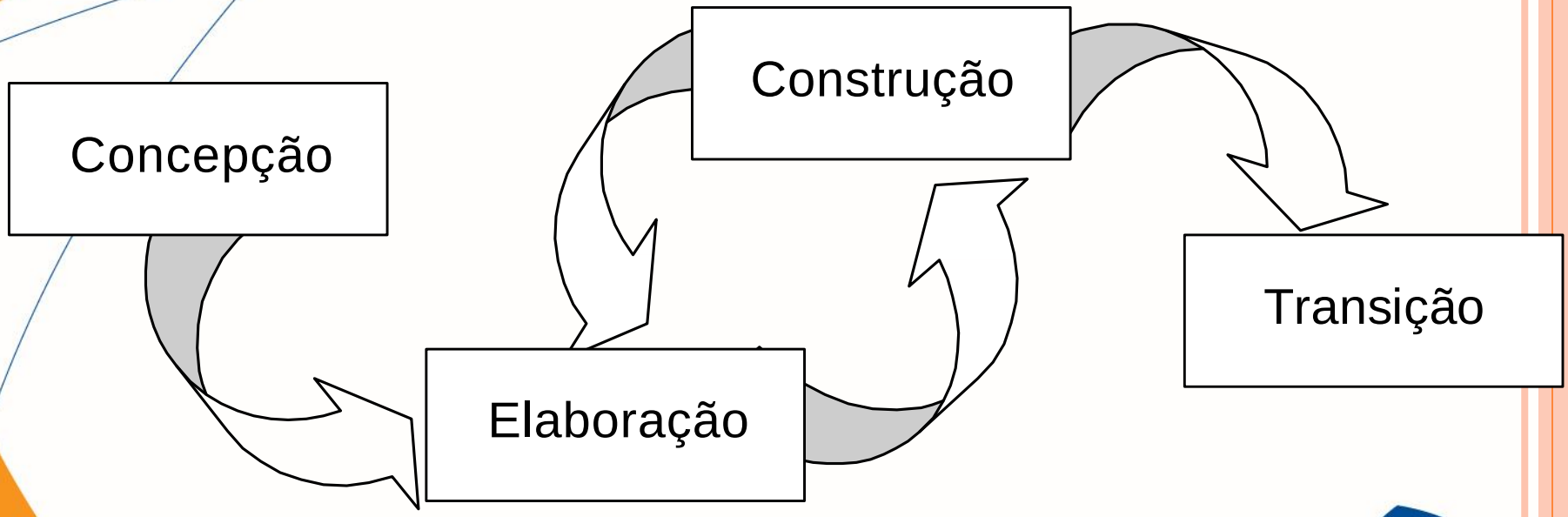
Testes

- A fase de *testes* envolve os testes de unidade, feitos pelo programador, para verificar se os componentes gerados atendem à especificação do projetista, e aos testes de caso de uso, normalmente efetuados por um analista experiente, que visam verificar a adequação do sistema aos requisitos inicialmente levantados.

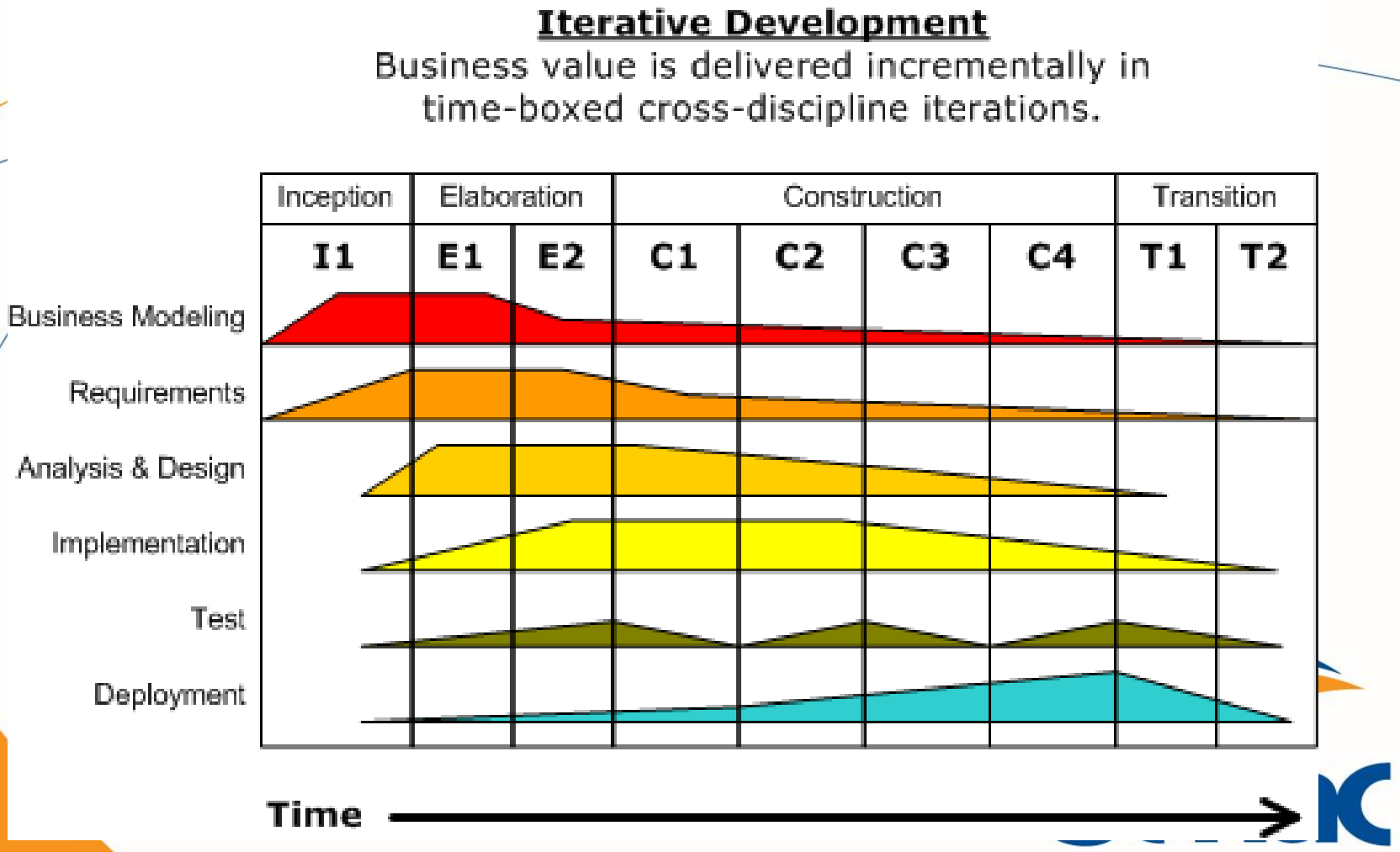
As quatro Fases do Processo Unificado

- A fase de concepção incorpora o estudo de viabilidade e uma parte da análise de requisitos.
- A fase de elaboração incorpora a maior parte da análise de requisitos, a análise de domínio e o projeto.
- A fase de construção corresponde à programação e testes.
- A fase de transição consiste na instalação e manutenção do sistema.

Ciclo de vida



Ciclo de vida



Análise de Requisitos

- A *análise de requisitos* é fundamental para o desenvolvimento de sistemas, pois trata justamente de descobrir o que o cliente quer com o sistema.
- A análise de requisitos está associada ao processo de descobrir quais são as operações que o sistema deve realizar e quais são as restrições que existem sobre estas operações.

Requisitos

- Funcionais – o que o sistema deve fazer
- Não-funcionais – restrições sobre como o sistema deve desempenhar suas funções

Erro comum

- Deve ficar claro ao analista que requisitos são coisas que o cliente ou usuário *solicitam*, e não coisas que ele, como analista, *planejou*.

Dúvidas?