Projeto Integrador II

Princípios de Análise e Projeto de Sistemas com UML (livro de Eduardo Bezerra)

Prof. Arliones Hoeller e Eraldo Silveira e Silva

arliones.hoeller@ifsc.edu.br

31 de julho de 2014

Cap.1 – Visão Geral

Sistema de Informação

- Combinação de pessoas, dados, processos, interfaces, redes de comunicação que interagem com objetivo de dar suporte e/ou melhorar um processo de negócio de empresa.

- Um dos componentes de um sistema de informação é o sistema de software.

Modelagem do Sistema de Software

- Para construir um sistema de software complexo deve-se elaborar um ou mais modelos.
- Um modelo é uma visão idealizada de um sistema. Modelo de um circuito elétrico, uma maquete etc.

Razões para uso de modelos

- Gerenciamento de complexidade
- Comunicação entre as pessoas envolvidas
- Redução de custos no desenvolvimento
- Predição do comportamento futuro do sistema.

No caso de sistemas de software,
a modelagem é realizada com notações gráficas e textuais que representam as partes essenciais do sistema

O Paradigma de Orientação a Objetos

- Paradigma: uma forma de abordar um problema.
- Nas décadas que se seguiram a criação dos primeiros computadores o mundo passou por diferentes paradigmas para o desenvolvimento do software. Exemplo: paradigma de programação e projeto estruturado na década de 80.

O Paradigma de Orientação a Objetos

- Princípios:
  - Qualquer coisa é um objeto
  - Objetos realizam tarefas através de requisição de serviços a outros objetos (e serviços internamente implementados)
  - Cada objeto pertence a uma determinada classe;
  - Classes são organizadas em hierarquias

No paradigma orientado a objetos, o objeto é uma unidade autônoma que contém os seus próprios dados que são manipulados por processos (métodos) do próprio objeto e que integragem com outros objetos para atingir um objetivo

O paradigma orientado a objetos visualiza um sistema de software como uma coleção de agentes (objetos) interconectados. Cada objeto realiza uma tarefa específica. A interação entre estes objetos permite a execução de uma tarefa computacional maior.

Classes e Objetos

- Na terminologia da orientação a objetos, as coisa do mundo real são denominadas objetos.
- Em modelagem de sistemas, uma classe é uma abstração das características relevantes de um grupo de coisas do mundo real.
- O princípio da abstração é importante para gerenciar a complexidade. Ele depende da perspectiva ou do contexto usado.

Mensagens

- Para que um objeto realize uma operação ele deve receber um estímulo: uma mensagem.
- Um objeto pode requisitar uma operação de outro objeto enviando uma mensagem para este objeto.

Encapsulamento e

- Objetos possuem comportamento. A classe do objeto define o seu comportamento.
- O mecanismo de encapsulamento restringe o acesso comportamento interno do objeto;
- A interface de um objeto define os serviços que ele pode realizar e, portanto, as mensagens que ele pode receber.
- Note que uma interface pode ter diferentes implementações, mas estas não são vistas pelo objeto requisitante.

Polimorfismo

- Capacidade de abstrair várias implementações diferentes em uma mesma interface.
- Um objeto pode enviar a mesma mensagem para objetos semelhantes, mas que implementam a sua interface de forma diferente.

Herança

- Classes semelhantes são agrupadas em hierarquias.
- Cada nível de uma hierarquia pode ser visto como um nível de abstração.
- Cada classe em um nível de hierarquia herda as características das classes de um nível acima.
- Este me mecanismo facilita o compartilhamento de comportamento e organiza as diferenças/variações entre classes semelhantes.

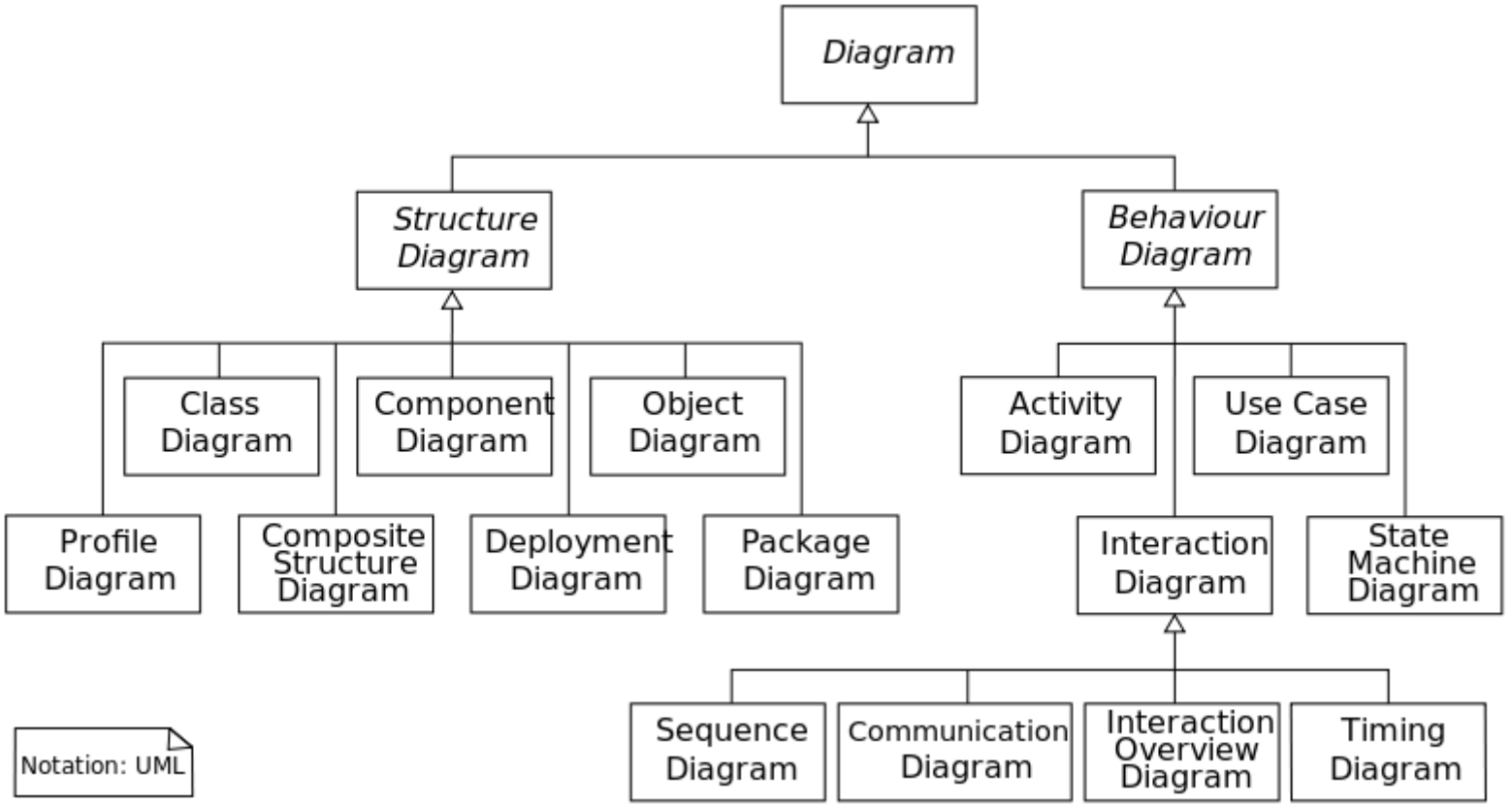
A Linguagem de Modelagem Unificada (UML)

- A UML é uma linguagem visual para modelar sistemas orientados a objetos. Foi especificada pela OMG e baseada em três contribuições (Booch, OMT e OOSE).
- Ela é constituída de elementos gráficos que permitem construir diagramas que representam o sistema sobre diferentes perspectivas.
- Cada elemento gráfico possui uma sintaxe (que define a sua forma) e uma semântica (significado). Estes elementos podem ser extensíveis.
- A UML independe da linguagem de programação a ser utilizada.

Visões de um Sistema

- Os autores da UML sugerem as seguintes visões de sistemas:
  - Visão de Caso de Uso;
  - Visão de Projeto;
  - Visão de Implementação;
  - Visão de Implantação;
  - Visão de Processo.

Diagramas UML



Exercício

- Suponha que você está trabalhando na concepção de um sistema de controle de empréstimo de livros de uma biblioteca. Liste possíveis objetos e classes deste sistema.

CAP.2 – O Processo de Desenvolvimento de Software

O desenvolvimento de software é uma atividade complexa e o NÃO USO de uma metodologia adequada pode levar a resultados desastrosos

O processo de desenvolvimento de software compreende todas atividades necessárias para definir, desenvolver , testar e manter um produto de software.

NOTA: UML, por si só, não é uma metodologia!

Atividades Típicas de um Processo de Desenvolvimento

- Levantamento de Requisitos
- Análise de Requisitos
- Projeto
- Implementação
- Testes
- Implantação

Levantamento de Requisitos

- Requisito: “uma condição ou capacidade que deve ser alcançada por um sistema ou componente deste, para satisfazer um contrato, especificação ou outros documentos formalmente impostos”
- Requisitos são identificados a partir de um domínio de negócio (área de conhecimento específica em que o sistema se insere);
- O levantamento de requisitos corresponde a etapa de compreensão do problema aplicada ao desenvolvimento do software.

Levantamento de Requisitos

- Durante o levantamento de requisitos a equipe de desenvolvimento tenta entender o domínio do negócio

Documento de Requisitos

- Escrito de maneira informal mas deve conter:
  - Requisitos funcionais do tipo:

    “O sistema deve permitir que um funcionário registre o empréstimo de um livro”
  - Requisitos não funcionais: questões relativas ao desempenho, segurança, usabilidade, confiabilidade etc.
    - Desempenho: “Um registro de empréstimo de livro não deve demorar mais que 5 segundos a partir do momento de inserção”
    - Confiabilidade: O tempo médio entre falhas do sistema é de 1 ano.
  - Restrições: adequação de custos, prazos, plataforma tecnológica, aspectos legais, limitações sobre interfaces etc.

Medição de Qualidade do Software baseada nos requisitos

- Pode-se medir a qualidade do sistema pela sua capacidade de atender aos requisitos definidos.
- Neste sentido os requisitos devem ser expressos de forma a atender ao público técnico e não técnico.

- NOTA: em particular, clientes devem entender o documento de requisitos de forma a priorizar o desenvolvimento dos mesmos.

O QUE o documento de requisitos não deveria ter

- O foco do levantamento de requisitos é responder a pergunta O QUE SE QUER FAZER
- Neste sentido, o documento de requisitos não "deveria" conter informações sobre soluções técnicas que devem ser adotadas...

O Documento de Requisitos serve como um termo de consenso entre cliente e equipe técnica. Ele deveria servir como base contratual...

O escopo do sistema deve estar claramente definido no Documento de Requisitos. Qualquer alteração implica em alterações de custo, prazo etc.

Requisitos Voláteis

- Alguns requisitos podem sofrer alterações ao longo do projeto.
- O impacto financeiro, de prazo destas alterações devem ser avaliados.

NOTA: a existência de requisitos voláteis corresponde mais a regra do que a exceção.

Prioridade de Requisitos

- É desejável ordenar os requisitos por ordem de prioridade.
- A priorização permite que o gerente de projeto possa tomar decisões acerca do momento no qual cada requisito deve ser considerado durante o desenvolvimento do sistema.

Análise de Requisitos

- Etapa na qual os analistas analisam com detalhes os requisitos levantados na etapa anterior.
- Este estudo deve permitir gerar modelos a partir dos quais o sistema será construído.
- Como resultado da análise tem-se:
  - Modelos que especificam as funcionalidades do sistema de software;
  - Modelos (um primeiro modelo) que representam as estruturas das classes de objetos que compõem o sistema

Projeto

- Esta fase do desenvolvimento se preocupa em COMO o sistema será construído de forma a atender os requisitos levantados.
- Esta fase deve se preocupar com a arquitetura do sistema, o padrão de interface gráfica, o gerenciador de banco de dados a ser usado, entre outros.
- Dois focos principas do projeto:
  - Projeto da Arquitetura
  - Projeto Detalhado

Projeto da Arquitetura

- Distribuição de classes de objetos em subsistemas e componentes;
- Distribuição de componentes em recursos de hardware;
- O projeto de arquitetura do software “deveria” ser realizado por um arquiteto de software

Projeto Detalhado

- São modeladas as colaborações entre os objetos de cada módulo com o objetivo de realizar as funcionalidades do módulo
- São projetadas as interfaces com os usuários;
- O banco de dados é projetado
- Nesta fase, os diagramas UML normalmente usados são os diagramas de classe, de casos de uso, de interação, de estados e de atividades.

NOTA: Deve-se observar que as fase de análise de requisitos e projeto não são claramente delimitadas.

Implementação

- Fase de codificação do sistema
- Mapeamento de classes definidas no projeto em classes da linguagem de programação
- Reutilização de classes e componentes de bibliotecas.

Testes

- Atividades de verificação do sistema levando em conta principalmente as especificações realizadas no início do desenvolvimento;
- Deve ser gerado um relatório de testes nesta fase

Implantação

- O sistema é empacotado, distribuído e instalado no ambiente do usuário.
- Manuais são escritos
- Dados são importados
- Usuários são treinados.

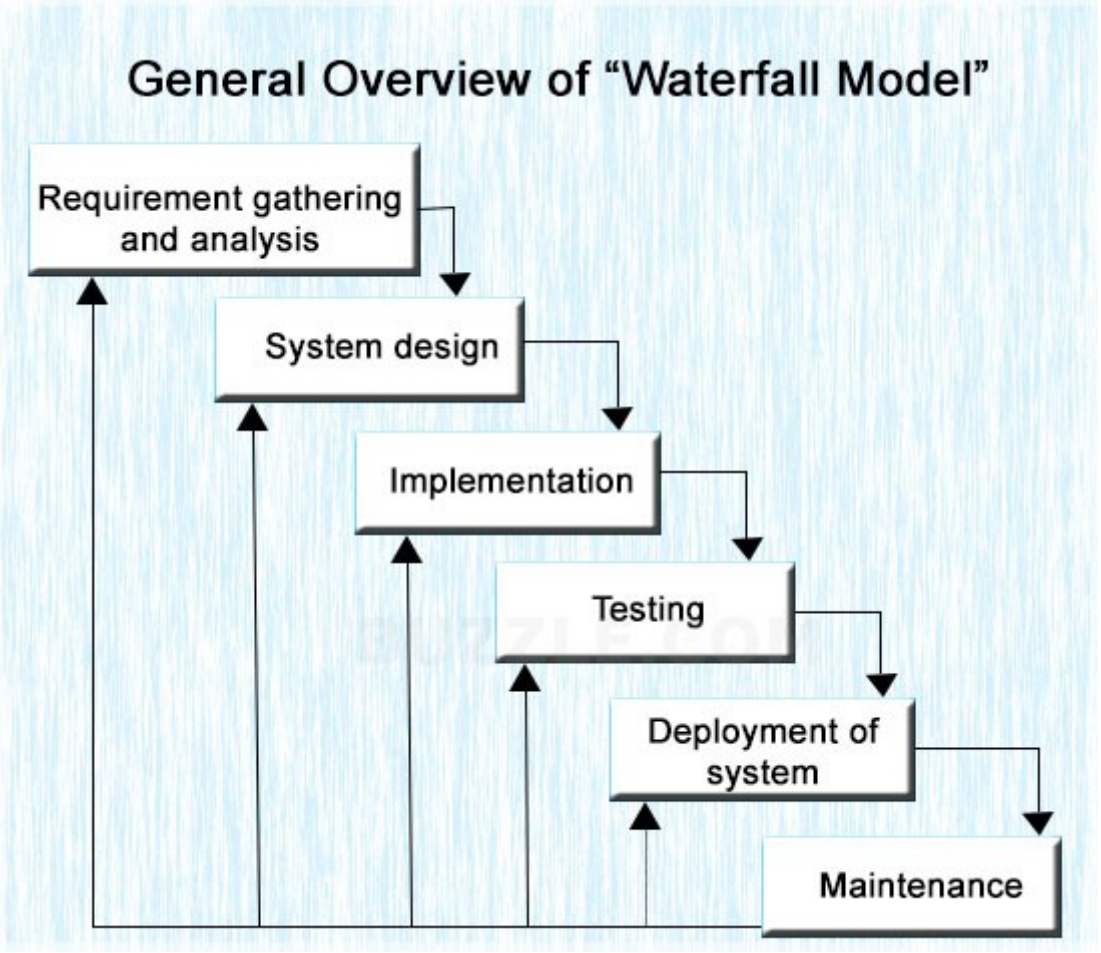
Componentes Humanos Envolvidos

- Gerentes de projeto
- Analistas
- Projetistas
- Arquitetos de Software
- Programadores
- Clientes
- Avaliadores de Qualidade

Modelos de Ciclo de Vida

- Modelo em Cascata (Waterfall): usado no PI1
- Modelo de ciclo de vida iterativo e incremental

Modelo em Cascata (Waterfall): usado no PI1



Modelo de ciclo de vida iterativo e incremental

- Divide o desenvolvimento em ciclos
- A cada ciclo corresponde um desenvolvimento similar ao modelo Waterfall
- Cada ciclo corresponde a um subconjunto dos requisitos levantados.
- Desta forma, o sistema incorpora gradativamente novas funcionalidades

Modelo de ciclo de vida iterativo e incremental