Unified Modeling Language (UML)

Universidade Federal do Maranhão – UFMA
Pós Graduação de Engenharia de Eletricidade
Grupo de Computação
Assunto: Introdução

Autoria:Aristófanes Corrêa Silva
Adaptação: Alexandre César M de Oliveira

- Bibliografia Utilizada
  - Ariadne Training. UML Applied: Object Oriented Analysis and Design Using the UML. http://www.ariadnetraining.co.uk, 2001.
  - COCKBURN, Alistair. Writing Effective Use Case. Addison-Wesley, 2000.
  - CONALLEN, Jim. Building Web Applications with UML. Addison-Wesley, 2000.
  - FOWLER, Martin, SCOTT, Kendal. UML Essencial. Bookman, 2000.
  - FURLAN, José Davi. Modelagem de Objetos Através da UML – The Unified Modeling Language. Makron Books, 1998.
  - LARMAN, Craig. Applying UML and Patterns: Na Introduction to Object-Oriented Analysis and Design and the Unified Process. Free Book for Everyone
  - MARTIN, Robert Cecil. UML for Java Programmers. Prentice Hall, 2002.
  - MEDEIROS, Ernani. Desenvolvendo Software com UML 2.0. Makron Books, 2004.

1 Introdução

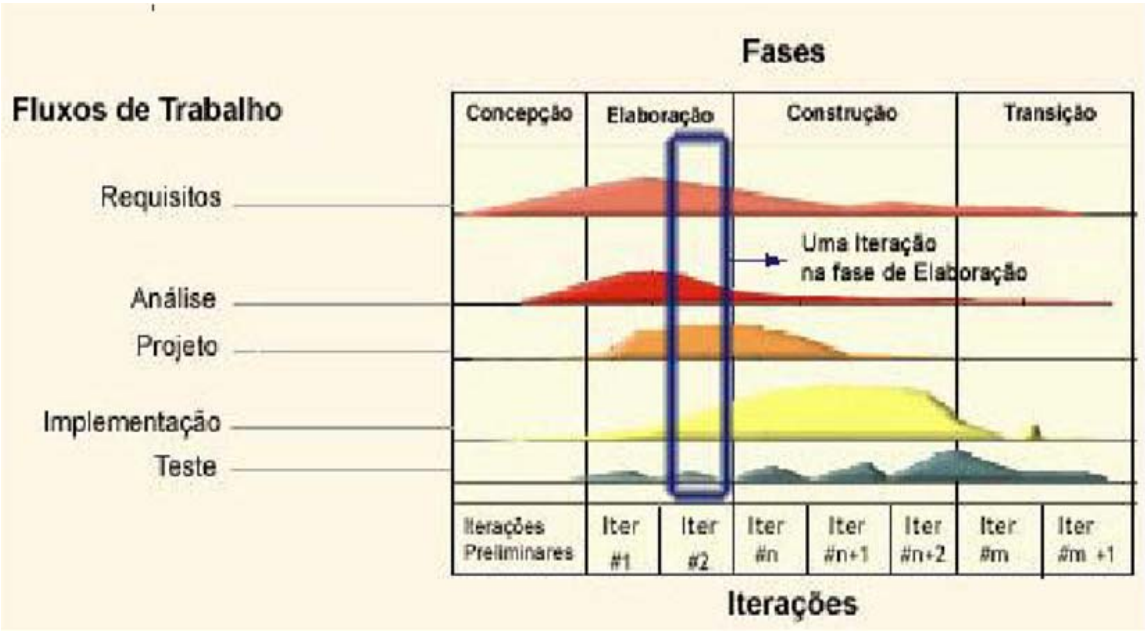
1.1 Motivação

- A modelagem é parte central de todas as atividades que levam à implantação de um bom sistema.
- Somente com o auxílio da modelagem é possível visualizar e controlar o desenvolvimento do sistema de maneira eficaz, identificando e gerenciando riscos, estipulando e cumprindo prazos, dentro das estimativas do custo.
- Métodos de destaque são o Booch de Grady Booch, OOSE (Engenharia de Software Orientada a Objetos) de Ivar Jacobson e OMT (Técnica de Modelagem de Objetos) James Rumbaugh.
- Motivação para criarem, juntos, uma *linguagem unificada de modelagem*, pelo fato de seus métodos estarem evoluindo um em direção ao outro de maneira independente.
- Outro motivo foi que a unificação dos métodos traria estabilidade ao mercado orientado a objetos.
- Além disso, o surgimento de uma linguagem unificada seria um aprimoramento dos métodos já existentes, sendo capaz de solucionar problemas que nenhum método resolveria até então.

- A unificação dos métodos para a criação de um novo padrão.
- A UML (Linguagem de Modelagem Unificada) é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.
- A UML é parte de uma metodologia, uma linguagem de modelagem do sistemas, ela apenas disponibiliza as ferramentas necessárias para se criar e ler modelos, mas não aponta quais os modelos deverão ser criados, nem quando deverão ser criados. Esta tarefa é de responsabilidade de um processo de desenvolvimento de sistemas.
- Mesmo sendo independente o UML precisava interagir com um metodologia especifica que pudesse obter o máximo proveito dos recursos da Linguagem de Modelagem Unificada, foi então criado o Processo Unificado.
- Com a integração com Processo Unificado à UML respostas para *quem* está fazendo *o que*, *quando* e *como* puderam ser definidas e um padrão no desenvolvimento de sistemas orientados a objeto pode ser estabelecido.

1.2 Fases de Desenvolvimento

- Existem cinco fases no desenvolvimento de sistemas de software: análise de requisitos, análise, projeto, programação e testes.
- Estas cinco fases não devem ser executadas na ordem descrita acima, mas concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente de forma que o resultado global gere um produto de alta qualidade e performance.



Análise de Requisitos

- Esta fase captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas "caso de uso (*use-cases*)".
- Através do desenvolvimento de caso de uso, as entidades externas ao sistema ("atores externos") que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem, funções estas chamadas de "caso de usos".
- Os atores externos e os casos de uso são modelados com relacionamentos que possuem comunicação associativa entre eles ou são desmembrados em hierarquia.
- O diagrama de caso de uso mostrará o que os atores externos, ou seja, os usuários do futuro sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade sem importar como esta será implementada.
- A análise de requisitos também pode ser desenvolvida baseada em processos de negócios, e não apenas para sistemas de software.

- Análise
  - A fase de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema.
  - As classes são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de Classe.
  - As colaborações entre classes também são mostradas neste diagrama para desenvolver os caso de usos modelados anteriormente, estas colaborações são criadas através de modelos dinâmicos em UML.
  - Na análise, só serão modeladas classes que pertençam ao domínio principal do problema do software, ou seja, classes técnicas que gerenciem banco de dados, interface, comunicação, concorrência e outros não estarão presentes neste diagrama.

- Design (Projeto)
  - Na fase de design, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros.
  - As classes do domínio do problema modeladas na fase de análise são mescladas nessa nova infra-estrutura técnica tornando possível alterar tanto o domínio do problema quanto a infra-estrutura.
  - O design resulta no detalhamento das especificações para a fase de programação do sistema.

- Programação
  - Na fase de programação, as classes provenientes do design são convertidas para o código da linguagem orientada a objetos escolhida (a utilização de linguagens procedurais é extremamente não recomendada).
  - Dependendo da capacidade da linguagem usada, essa conversão pode ser uma tarefa fácil ou muito complicada.
  - No momento da criação de modelos de análise e design em UML, é melhor evitar traduzi-los mentalmente em código.
  - Nas fases anteriores, os modelos criados são o significado do entendimento e da estrutura do sistema, então no momento da geração do código onde o analista conclua antecipadamente sobre modificações em seu conteúdo, seus modelos não estarão mais demonstrando o real perfil do sistema.
  - A programação é uma fase separada e distinta, onde os modelos criados são convertidos em código.

- Testes
  - Um sistema normalmente é rodado em testes de unidade, integração, e aceitação.
  - Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador.
  - Os testes de integração são aplicados já usando as classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos.
  - Os testes de aceitação observam o sistema como uma “ caixa preta” e verificam se o sistema está funcionando como o especificado nos primeiros diagramas de “use-cases”.
  - O sistema será testado pelo usuário final e verificará se os resultados mostrados estão realmente de acordo com as intenções do usuário final.

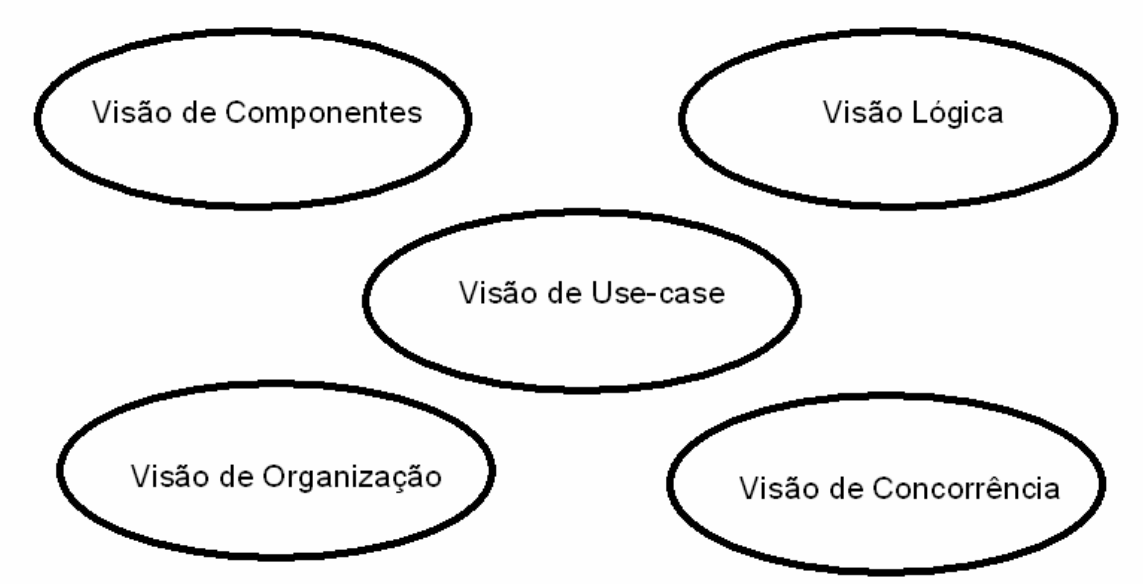
1.3 Fases do Sistema

- Visões
- Modelos de elementos
- Diagramas

- Serão utilizados na criação dos diagramas e mecanismos gerais que todos em conjunto especificam e exemplifica a definição do sistema, tanto a definição no que diz respeito à funcionalidade estática e dinâmica do desenvolvimento de um sistema.
- Um sistema é composto por diversos aspectos: funcional (que é sua estrutura estática e suas interações dinâmicas), não funcional (requisitos de tempo, confiabilidade, desenvolvimento, etc.) e aspectos organizacionais (organização do trabalho, mapeamento dos módulos de código, etc.)
- O sistema é descrito em um certo número de visões, cada uma representando uma projeção da descrição completa e mostrando aspectos particulares do sistema.

1.3.1 Visões

- São abstrações consistindo em uma série de diagramas.
- São descritas por um número de diagramas que dão ênfase aos aspectos particulares do sistema.
- Mostram diferentes aspectos do sistema que está sendo modelado: aspectos particulares do sistema, dando enfoque a ângulos e níveis de abstrações diferentes e uma figura completa do sistema poderá ser construída.
- Servem de ligação entre a linguagem de modelagem e o método/processo de desenvolvimento escolhido.



- Os diagramas que compõem as visões contêm os modelos de elementos do sistema e podem fazer parte de mais de uma visão.

- Visão “caso de uso”
  - Descreve a funcionalidade do sistema desempenhada pelos atores externos do sistema (usuários).
  - É central, já que seu conteúdo é base do desenvolvimento das outras visões do sistema.
  - É montada sobre os diagramas de use-case e eventualmente diagramas de atividade.
- Visão Lógica
  - Descreve como a funcionalidade do sistema será implementada.
  - É feita principalmente pelos analistas e desenvolvedores. Em contraste com a visão use-case, a visão lógica observa e estuda o sistema internamente.
  - Descreve e especifica a estrutura estática do sistema (classes, objetos, e relacionamentos) e as colaborações dinâmicas (diagramas de estado, seqüência, colaboração e atividade) quando os objetos enviarem mensagens uns para os outros para realizarem as funções do sistema.

- Visão de Componentes
  - É uma descrição da implementação dos módulos e suas dependências. É principalmente executado por desenvolvedores, e consiste nos componentes dos diagramas.
- Visão de Organização
  - Mostra a organização física do sistema, os computadores, os periféricos e como eles se conectam entre si.
  - Será executada pelos desenvolvedores, integradores e testadores, e será representada pelo diagrama de execução.
- Visão de concorrência
  - Trata a divisão do sistema em processos e processadores (propriedade não funcional do sistema), permitindo uma melhor utilização do ambiente de execução do sistema
  - Considera se o sistema possui execuções paralelas, e se existe dentro do sistema um gerenciamento de eventos assíncronos, como se dá a comunicação e a concorrência destas threads.
  - É suportada pelos diagramas dinâmicos, que são os diagramas de estado, seqüência, colaboração e atividade, e pelos diagramas de implementação, que são os diagramas de componente e execução.

1.3.2 Modelo de Elementos

- Os conceitos usados nos diagramas são chamados de modelos de elementos.
- Um modelo de elemento é definido com a semântica, a definição formal do elemento com o exato significado do que ele representa sem definições duvidosas ou ambíguas e também define sua representação gráfica que é mostrada nos diagramas da UML.
- Um elemento pode existir em diversos tipos de diagramas, mas existem regras que definem que elementos podem ser mostrados em que tipos de diagramas.

- Modelo de Elementos
  - Alguns exemplos de modelos de elementos são as classes, objetos, estados, pacotes e componentes.
  - Os relacionamentos também são modelos de elementos, e são usados para conectar outros modelos de elementos entre si.

1.3.3Diagramas

- Os diagramas utilizados pela UML são compostos de nove tipos: diagrama de use case, de classes, de objeto, de estado, de seqüência, de colaboração, de atividade, de componente e o de execução.
- Todos os sistemas possuem uma estrutura estática e um comportamento dinâmico.
- A UML suporta modelos estáticos (estrutura estática), dinâmicos (comportamento dinâmico) e funcional. A Modelagem estática é suportada pelo diagrama de classes e de objetos, que consiste nas classes e seus relacionamentos.
- Os relacionamentos podem ser de associações, herança (generalização), dependência ou refinamentos.
- Os modelamentos dinâmicos são suportados pelos diagramas de estado, seqüência, colaboração e atividade.
- E a modelagem funcional é suportada pelos diagramas de componente e execução.