3

TEXTO BASE

ANÁLISE E MODELAGEM DE SISTEMAS

Faculdade IMPACTA

UML

Histórico e Diagramas

Prof. Renato de Tarso Silva

Resumo

Neste documento são realizadas as primeiras explanações sobre a linguagem de modelagem unificada, a UML (Unified Modeling Language), bem como um breve histórico será abordado a fim de se entender a origem e o processo de amadurecimento desta linguagem, que se tornou um padrão para modelagem de sistema de software.

1.1. A UML

A UML oferece formas padronizadas que viabilizam a construção de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais: processos de negócios e funções do sistema, e itens concretos: classes, esquemas de bancos de dados e componentes de software reutilizáveis.

De acordo com Booch (2005), "A UML, Linguagem Unificada de Modelagem, é uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas complexos de software."

Existiram várias metodologias de modelagem de sistemas até o surgimento da UML. Uma falta de padronização em tais práticas perdurava entre desenvolvedores, que foi sanada através da união das melhores ideias de cada metodologia.

1.1.1 UML: Versões

O início da evolução da UML se dá em 1994, quando a concepção de suas primeiras versões foi iniciada. Iniciativas conjuntas de determinados métodos de modelagem foram gradativamente aglomerando-se com intuito de unificar os meios conhecidos na comunidade para se especificar soluções de sistemas de software.

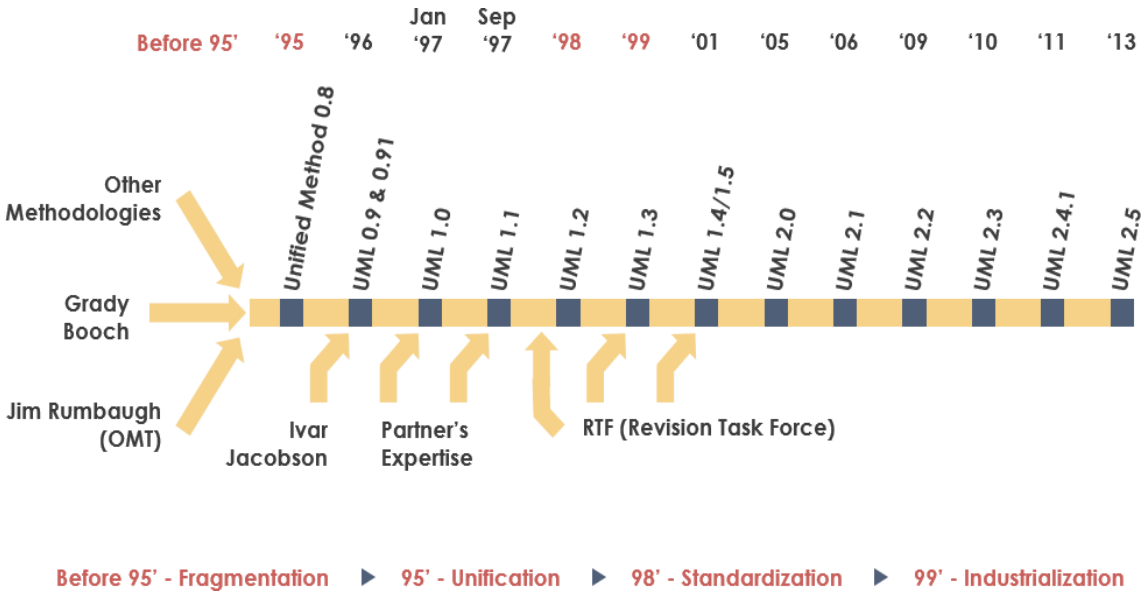

Figura 1.1. Evolução da UML

Três iniciativas, na história da UML, são considerados fundamentais para sua criação, vista a essencial participação de cada uma delas e de seus respectivos métodos.

- Booch – O método de Grady Booch para desenvolvimento orientado a objetos está disponível em muitas versões. Booch definiu a noção de que um sistema é analisado a partir de um número de visões, no qual cada visão é descrita por um número de modelos e diagramas.
- OMT – Técnica de Modelagem de Objetos (Object Modelling Technique). Um método desenvolvido pela GE (General Electric), onde James Rumbaugh trabalhava. O método é especialmente voltado para o teste dos modelos, baseado nas especificações da análise de requisitos do sistema.
- OOSE/Objectory – Desenvolvido/baseado em um mesmo ponto de vista formado por Ivar Jacobson. O OOSE é um método orientado a objetos, e o Objectory é usado para a construção de sistemas tão diversos quanto forem. Ambos os métodos são baseados na utilização de *use-cases*, que definem os requisitos iniciais do sistema, vistos por atores.

Além de outros métodos menos conhecidos na época, a colaboração de empresas de renome, com alto alcance de mercado, ajudaram na idealização dessa linguagem unificada de modelagem de sistemas de software. Cita-se aqui algumas das mais notáveis: Rational Software Corporation, Hewlett-Packard (HP), IBM, Microsoft, Oracle, Unisys.

Por vários anos, a UML foi mantida pela OMG (Object Management Group), que produziu as versões 1.3 a 1.5. Em 2003, a versão 2.0 foi revisada por um ano em uma força-tarefa de finalização liderada por Bran Selic, da IBM, e a versão oficial da

UML 2.0 foi adotada pelo OMG em 2005. A UML 2.0 é uma revisão importante da UML e inclui um grande número de recursos adicionais. Além disso, muitas alterações foram feitas nas construções anteriores com base na experiência obtida. Documentos de especificação UML reais podem ser encontrados no site da OMG (www.omg.org). A última versão da UML, a 2.5, traz, além de melhorias das características da própria linguagem, um novo diagrama, o Diagrama de Perfil.

A UML está destinada a ser dominante como linguagem de modelagem comum nas indústrias. Ela está totalmente baseada em conceitos e padrões extensivamente testados, provenientes das metodologias já existentes, muito bem documentadas, e com toda a especificação da semântica da linguagem disponibilizada.

1.2. Definição

A UML (Unified Modeling Language) é uma linguagem padrão para se criar projetos de software. É apropriada para modelar sistemas que variam de sistemas de informações corporativas a aplicativos baseados na Web e até sistemas distribuídos embarcados em tempo real.

Muito expressiva, a UML é uma linguagem que aborda todas as visões necessárias para desenvolver e implantar esses sistemas. Contudo, a UML não é difícil de entender e nem de utilizar. Aprender a aplicar a UML efetivamente começa com a formação de um modelo conceitual da linguagem, o que requer o aprendizado de três principais elementos: os componentes básicos da UML, as regras que determinam como esses componentes podem unir-se e alguns mecanismos comuns que se aplicam a toda a linguagem.

A UML é apenas uma linguagem, portanto, é apenas uma parte de qualquer método de desenvolvimento de software. A UML é independente do processo, embora, idealmente, deva ser usada em um processo orientado à arquitetura de casos de uso, centrada, iterativa e incremental.

1.3. Visão Geral dos Diagramas

Os diagramas da UML são divididos, basicamente, em dois supertipos: Diagramas Estruturais (Structural) e Diagramas Comportamentais (Behavioral).

Os Diagramas Estruturais, como o próprio nome já diz, denotam aspectos da estrutura dos elementos especificados no sistema, como, por exemplo, o Diagrama de Classes.

Os Diagramas Comportamentais trazem uma visão de como a dinâmica nos processos levantados acontecerá no sistema, como, por exemplo, o Diagrama de Casos de Uso.

Neste curso, serão abordados cinco diagramas da UML, eleitos devido à importância e usualidade destes perante à comunidade de desenvolvedores. São eles:

Diagrama de Casos de Uso, Diagrama de Classes, Diagrama de Atividades, Diagrama de Máquina de Estados e Diagrama de Sequência.

Note, na figura 1.2, como se dá a divisão dos diagramas entre os tipos supracitados.

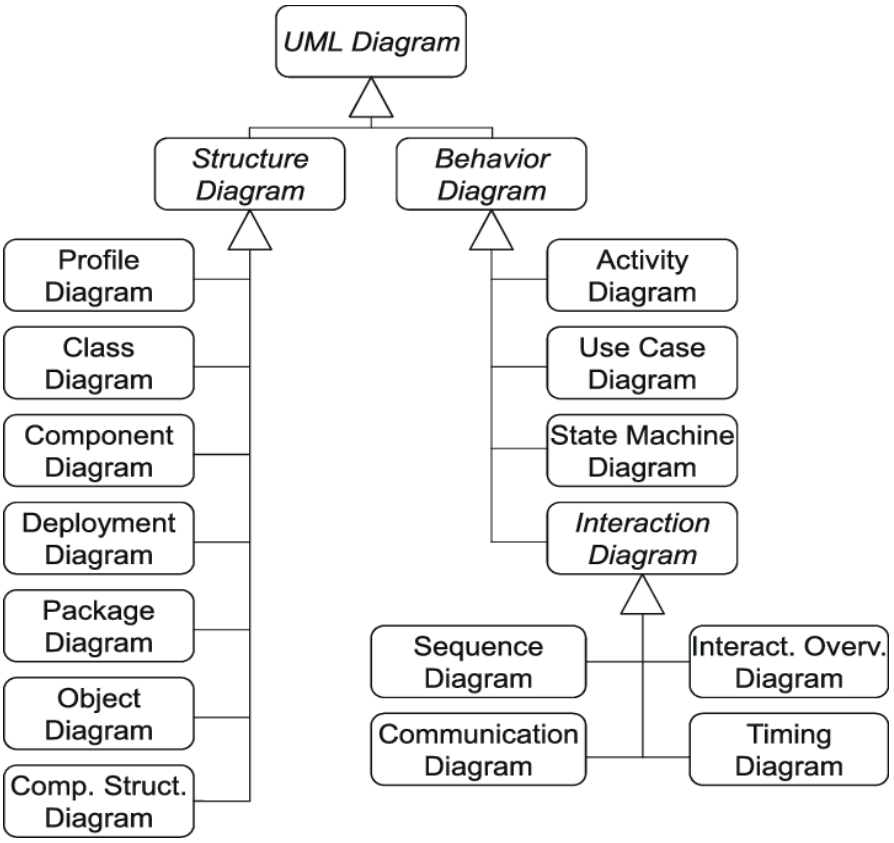

Figura 1.2. Diagramas UML

Os diagramas serão apresentados de forma superficial. Em aulas posteriores, cada diagrama será esmiuçado com seus elementos, características, e exemplificações práticas.

1.3.1. Diagrama de Casos de Uso

Sendo o primeiro Diagrama Comportamental a ser visto, o Diagrama de Casos de Uso traz entendimentos da dinâmica sistêmica, como: quem poderá participar/interagir com o sistema ou subsistema, os atores; as metas que atores podem atingir no sistema (Casos de Uso); as trocas de mensagens entre os atores e tais metas; a fronteira sistêmica, que reflete o escopo do sistema/subsistema, e contém objetivos de valor (metas/funcionalidades) disponíveis para que os sistemas/usuários (atores) atinjam. Está ilustrado, na figura 1.3.1, um exemplo de diagrama de Casos de Uso.

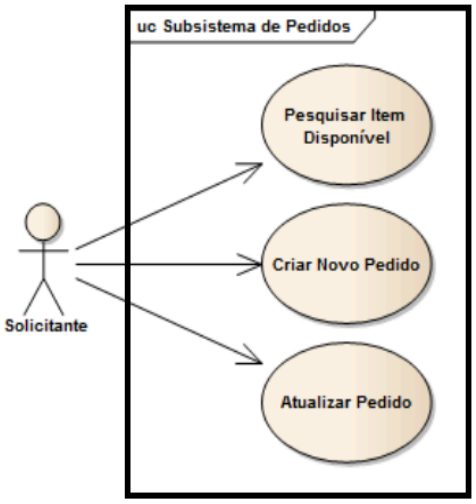

Figura 1.3.1. Diagrama de Casos de Uso

O diagrama de Casos de Uso é essencial devido ao seu alto nível de abstração. Facilidade que pessoas (mesmo que leigas) visualizem as principais funcionalidades e responsabilidades presentes no sistema.

1.3.2. Diagrama de Classes

Este é o primeiro Diagrama Estrutural abordado, o Diagrama de Classes é um diagrama que representa um conjunto de classes, interfaces e colaborações, assim como seus relacionamentos. É utilizado para fazer a modelagem de uma visão estática da estrutura de entidades e elementos substantivos do sistema.

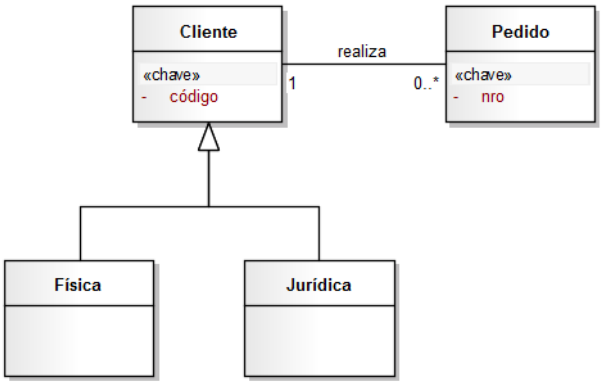

Figura 1.3.2. Diagrama de Classes

1.3.3. Diagrama de Sequência

Este é um diagrama comportamental dos mais refinados e úteis ao desenvolvimento, pois dá ênfase à ordenação sequencial das mensagens. Atente-se à figura 1.3.3. Nele, tipicamente, o objeto que inicia a interação é colocado à esquerda e objetos mais subordinados dispõe-se à direita, retratando a história de objetos enquanto trocam mensagens entre si para concluir determinado processo especificado no sistema.

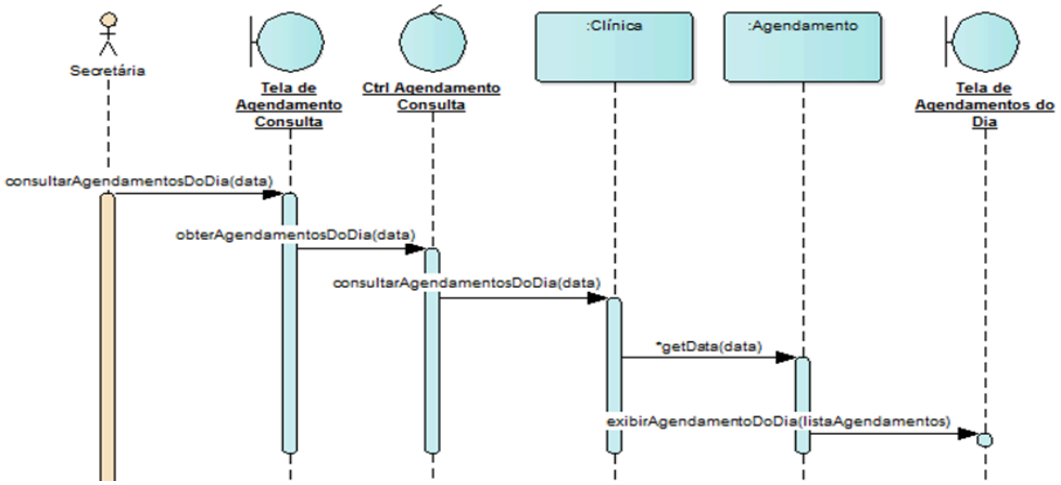

Figura 1.3.3. Diagrama de Sequência

1.3.4. Diagrama de Atividades

Este também é um Diagrama Comportamental. Bastante confundido com fluxograma, é essencialmente um gráfico que mostra os fluxos de controle de uma atividade para outra (com possíveis concorrências) e como as ramificações de controle acontecem em um determinado processo computacional. Note a Figura 1.3.4, e note o início, os fluxos, e atividades realizadas em um processo nomeado “Solicitar Produtos”.

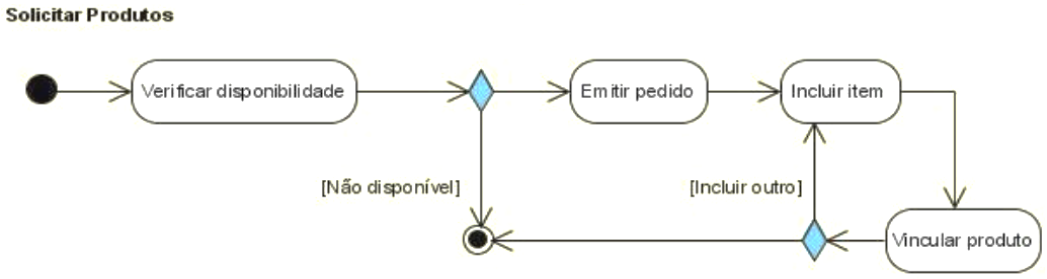

Figura 1.3.4. Diagrama de Atividades

1.3.5. Diagrama Máquinas de Estados

Este é o último dos Diagramas Comportamentais que abordaremos. Ilustrado na Figura 1.3.5, o Diagrama Máquinas de Estados, ou simplesmente Diagrama de Estados, especifica as sequências de estados pelos quais um determinado objeto passa durante seu tempo de vida em resposta a eventos, juntamente com suas respostas a esses eventos. Tais eventos são conhecidos como transições.

Figura 1.3.5. Diagrama de Atividades

Dependendo da cultura de desenvolvimento inserida, alguns dos diagramas da UML são tratados de maneira mais ou menos formal. Esses artefatos não são apenas parte do projeto a ser entregue, mas são críticos mediante o controle, a medição e a comunicação de determinado sistema durante seu desenvolvimento e após sua implantação. A UML pode abranger documentação arquitetural de sistemas, com todos os seus detalhes, e proporciona uma linguagem para expressão dos requisitos levantados. Nem todos os diagramas UML devem sempre ser utilizados, isso varia em função de escopo e criticidade da solução a ser especificada e desenvolvida.

Referências

Booch G., Rumbaugh J., Jacobson I. “The Unified Modeling Language user Guide” 2ª Edição. 2005.