

UML 2

Guia Prático

Gilleanes T.A. Guedes

Obra revisada e ampliada a partir do título Guia de Consulta Rápida UML 2

Novatec

CAPÍTULO 1

Introdução à UML

A UML (Unified Modeling Language ou Linguagem de Modelagem Unificada) é uma linguagem visual utilizada para modelar sistemas computacionais por meio do paradigma de Orientação a Objetos. Essa linguagem se tornou, nos últimos anos, a linguagem-padrão de modelagem de software adotada internacionalmente pela indústria de Engenharia de Software.

Deve ficar bem claro, no entanto, que a UML não é uma linguagem de programação, mas uma linguagem de modelagem, cujo objetivo é auxiliar os engenheiros de software a definir as características do software, tais como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema deverá ser implantado. Todas essas características são definidas por meio da UML antes de o software começar a ser realmente desenvolvido.

1.1 Breve Histórico da UML

A UML surgiu da união de três metodologias de modelagem: o método de Booch, o método OMT (Object Modeling Technique) de Jacobson e o método OOSE (Object-Oriented Software Engineering) de Rumbaugh. Essas eram, até meados da década de 1990, as três metodologias de modelagem orientada a objetos mais populares entre os profissionais da área de engenharia de software. A união dessas metodologias contou com o amplo apoio da Rational Software, que incentivou e financiou tal união.

O esforço inicial do projeto começou com a união do método de Booch com o método OMT de Jacobson, o que resultou no lançamento do Método Unificado no final de 1995. Logo em seguida, Rumbaugh juntou-se a Booch e Jacobson na Rational Software e seu método OOSE começou também a ser incorporado à nova metodologia. O trabalho de Booch, Jacobson e Rumbaugh, conhecidos popularmente como “Os Três Amigos”, resultou no lançamento, em 1996, da primeira versão da UML propriamente dita.

Tão logo a primeira versão foi lançada, diversas grandes empresas atuantes na área de engenharia e desenvolvimento de software passaram a contribuir com o projeto, fornecendo sugestões para melhorar e ampliar a linguagem. Finalmente a UML foi adotada pela OMG (Object Management Group ou Grupo de Gerenciamento de Objetos) em 1997, como uma linguagem-padrão de modelagem. A UML em sua versão 2.0 trouxe grandes novidades em relação à estrutura geral da linguagem principalmente com relação à abordagem de quatro camadas e à possibilidade de se desenvolver “perfis” particulares a partir da UML, cuja documentação oficial pode ser consultada no site da OMG em www.omg.com.

1.2 Por Que Tantos Diagramas?

O objetivo disso é fornecer múltiplas visões do sistema a ser modelado, analisando-o e modelando-o sob diversos aspectos, procurando-se assim atingir a completitude da modelagem, permitindo que cada diagrama complemente os outros. Cada diagrama da UML analisa o sistema, ou parte dele, sob uma determinada ótica; é como se o sistema fosse modelado em camadas. Alguns diagramas enfocam o sistema de forma mais geral, apresentando uma visão externa do sistema, como é o objetivo do Diagrama de Casos de Uso, ao passo que outros oferecem uma visão de uma camada mais profunda do software, apresentando um enfoque mais técnico ou ainda visualizando apenas uma característica específica do sistema ou um determinado processo.

A utilização de diversos diagramas permite que falhas possam ser descobertas nos diagramas anteriores, diminuindo a possibilidade da ocorrência de erros durante a fase de desenvolvimento do software. É importante destacar que, embora cada diagrama tenha sua utilidade,

nem sempre é necessário modelar um sistema utilizando-se de todos os diagramas, pois alguns deles possuem funções muito específicas, como é o caso do Diagrama de Tempo, por exemplo.

1.3 Resumo dos Diagramas da UML

A seguir descreveremos rapidamente cada um dos diagramas oferecidos pela UML, destacando suas principais características e objetivos. O leitor irá notar que cada diagrama possui uma aba em seu canto superior esquerdo contendo um operador e a descrição do diagrama. O operador serve para determinar qual é o tipo de diagrama, assim **uc** refere-se a Use Case Diagram (Diagrama de Caso de Uso); **class** a Class Diagram (Diagrama de Classes); **object**, a Object Diagram (Diagrama de Objetos); **sd**, a Sequence Diagram (Diagrama de Sequência); **stm**, a State-Machine Diagram (Diagrama de Máquina de Estados) e assim por diante.

1.3.1 Diagrama de Casos de Uso

Este é o diagrama mais geral e informal da UML, sendo utilizado principalmente para auxiliar no levantamento e análise dos requisitos, em que são determinadas as necessidades do usuário, e na compreensão do sistema como um todo, embora venha a ser consultado durante todo o processo de modelagem e sirva de base para todos os outros diagramas.

O Diagrama de Casos de Uso apresenta uma linguagem simples e de fácil compreensão para que os usuários possam ter uma idéia geral de como o sistema irá se comportar. Ele procura identificar os atores (usuários, outros softwares que interajam com o sistema ou até mesmo algum hardware especial), que utilizarão de alguma forma o software, bem como os serviços, ou seja, as opções que o sistema disponibilizará aos atores, conhecidas neste diagrama como Casos de Uso. A Figura 1.1 apresenta um exemplo desse diagrama.

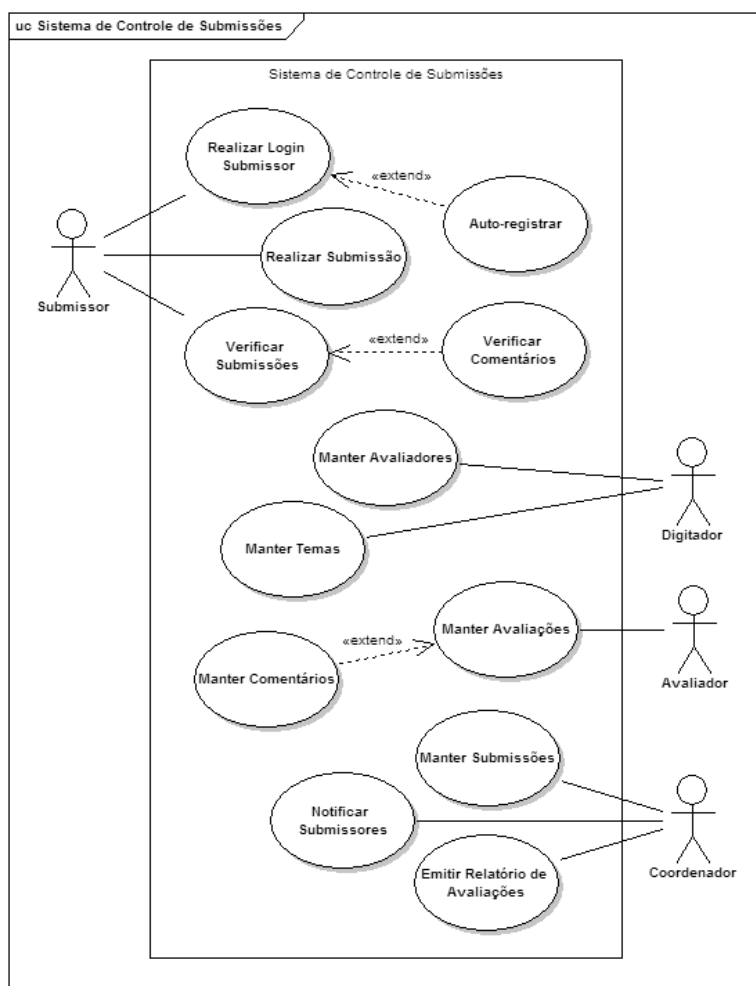


Figura 1.1 – Exemplo de Diagrama de Casos de Uso.

1.3.2 Diagrama de Classes

Este é o diagrama mais utilizado e o mais importante da UML, servindo de apoio para a maioria dos outros diagramas. Como o próprio nome diz, esse diagrama define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos possuídos por cada classe, além de estabelecer como as classes se relacionam e trocam informações entre si. A Figura 1.2 demonstra um exemplo desse diagrama.

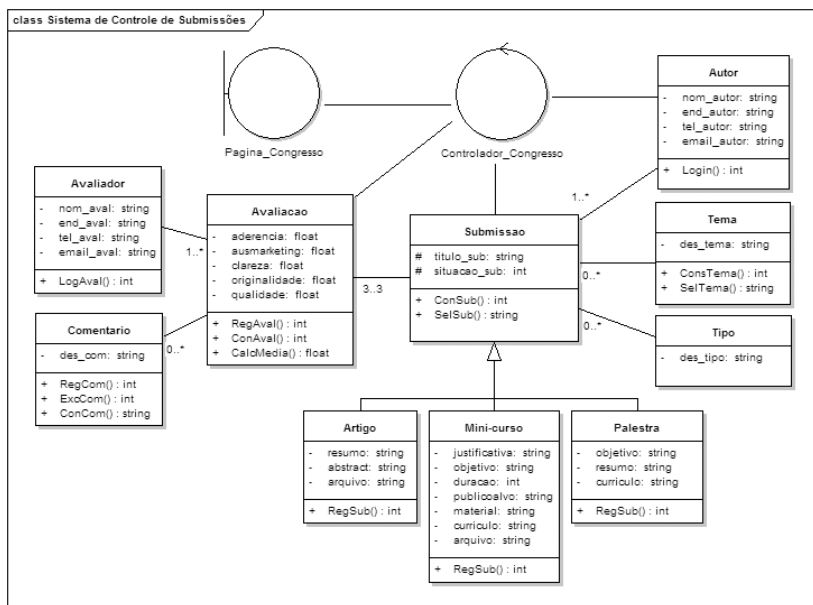


Figura 1.2 – Exemplo de Diagrama de Classes.

1.3.3 Diagrama de Objetos

Este diagrama está amplamente associado ao Diagrama de Classes. Na verdade, o Diagrama de Objetos é praticamente um complemento do Diagrama de Classes, sendo bastante dependente deste. O Diagrama de Objetos fornece uma visão dos valores armazenados pelos objetos de um Diagrama de Classes em um determinado momento da execução de um processo. A Figura 1.3 apresenta um exemplo de Diagrama de Objetos.

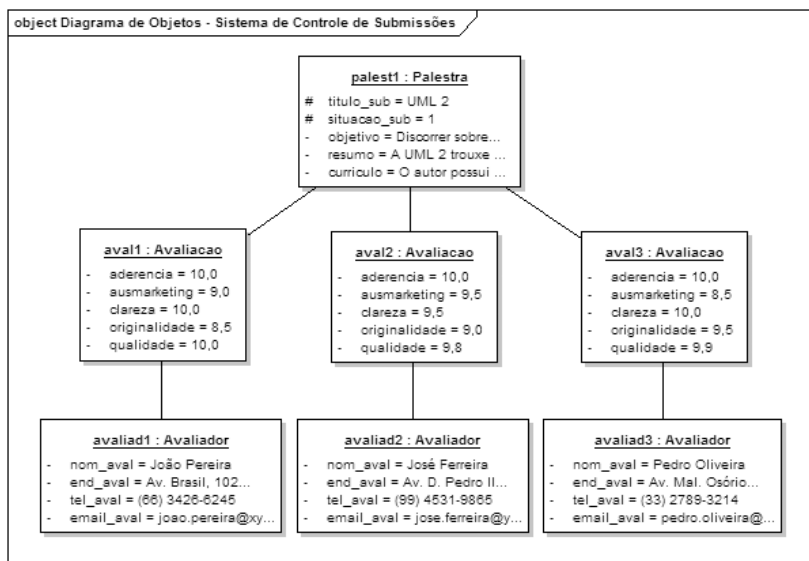


Figura 1.3 – Exemplo de Diagrama de Objetos.

1.3.4 Diagrama de Estrutura Composta

O Diagrama de Estrutura Composta é utilizado para modelar Colaborações. Uma colaboração descreve uma visão de um conjunto de entidades cooperativas interpretadas por instâncias que cooperam entre si para executar uma função específica. O termo estrutura desse diagrama refere-se a uma composição de elementos interconectados, representando instâncias de tempo de execução colaboram, por meio de vínculos de comunicação, para atingir algum objetivo comum. Esse diagrama também pode ser utilizado para definir a estrutura interna de um classificador. A Figura 1.4 apresenta um exemplo de Diagrama de Estrutura Composta.

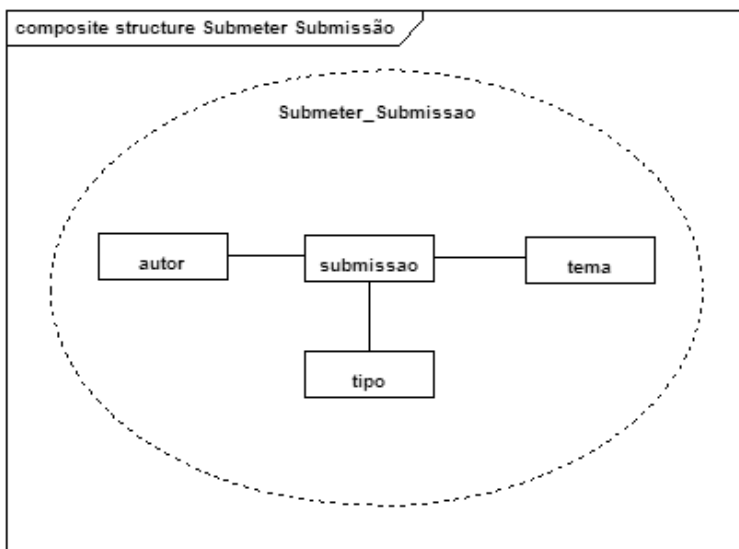


Figura 1.4 – Exemplo de Diagrama de Estrutura Composta.

1.3.5 Diagrama de Seqüência

O Diagrama de Seqüência preocupa-se com a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo. Em geral, baseia-se em um Caso de Uso definido pelo diagrama de mesmo nome e apóia-se no Diagrama de Classes para determinar os objetos das classes envolvidas em um processo, bem como os métodos disparados entre os mesmos. Um Diagrama de Seqüência costuma identificar o evento gerador do processo modelado, bem como o ator responsável por este evento, e determina como o processo deve se desenrolar e ser concluído por meio do envio de mensagens, que em geral disparam métodos entre os objetos. A Figura 1.5 apresenta um exemplo desse diagrama.

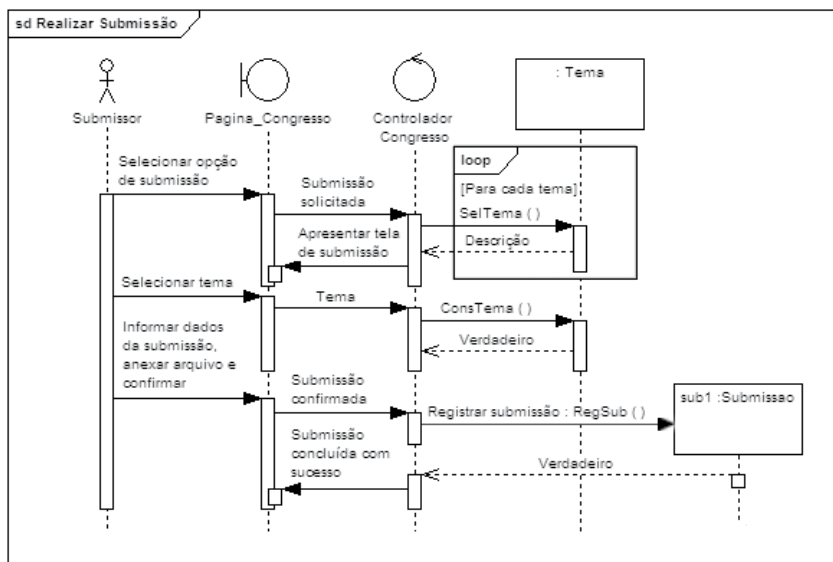


Figura 1.5 – Exemplo de Diagrama de Seqüência.

1.3.6 Diagrama de Comunicação

O Diagrama de Comunicação era conhecido como Diagrama de Colaboração até a versão 1.5 da UML, tendo seu nome modificado para Diagrama de Comunicação a partir da versão 2.0. Esse diagrama está amplamente associado ao Diagrama de Seqüência; na verdade, um complementa o outro. As informações mostradas no Diagrama de Comunicação são, com frequência, praticamente as mesmas apresentadas no Diagrama de Seqüência, porém com um enfoque diferente, visto que este diagrama não se preocupa com a temporalidade do processo, concentrando-se em como os objetos estão vinculados e quais mensagens trocam entre si durante o processo. A Figura 1.6 apresenta um exemplo de Diagrama de Comunicação.

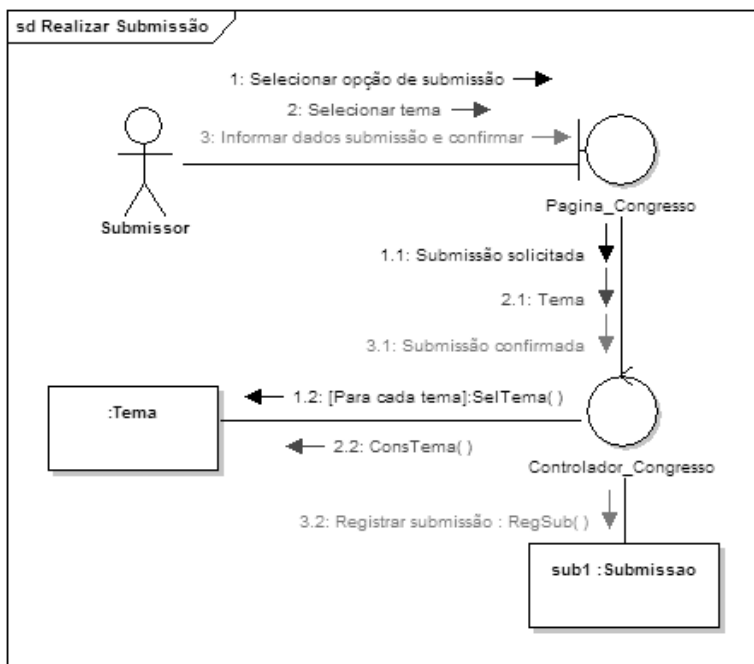


Figura 1.6 – Exemplo de Diagrama de Comunicação.

1.3.7 Diagrama de Máquina de Estados

O Diagrama de Máquina de Estados era conhecido nas versões anteriores da linguagem como Diagrama de Gráfico de Estados ou simplesmente como Diagrama de Estados, tendo assumido nova nomenclatura a partir da versão 2. Esse diagrama procura acompanhar as mudanças sofridas nos estados de uma instância de uma classe, de um Caso de Uso ou mesmo de um subsistema ou sistema completo. Como o Diagrama de Seqüência, o Diagrama de Máquina de Estados muitas vezes se baseia em um Caso de Uso e se apóia no Diagrama de Classes. A Figura 1.7 apresenta um exemplo de Diagrama de Máquina de Estados.

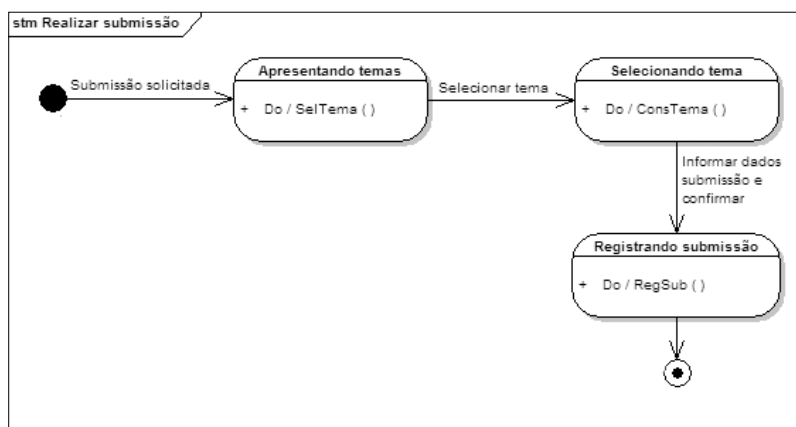


Figura 1.7 – Exemplo de Diagrama de Máquina de Estados.

1.3.8 Diagrama de Atividade

O Diagrama de Atividade era considerado um caso especial do antigo Diagrama de Gráfico de Estados, mas, a partir da UML 2.0, esse diagrama se tornou independente, deixando inclusive de se basear em máquinas de estados e passando a se basear em Redes de Petri. O Diagrama de Atividade se preocupa em descrever os passos a serem percorridos para a conclusão de uma atividade específica, muitas vezes representada por um método com um certo grau de complexidade, podendo, no entanto, modelar um processo completo. Concentra-se na representação do fluxo de controle e no fluxo de objeto de uma atividade. A Figura 1.8 apresenta um exemplo desse diagrama.