A complexibilidade da UML e seus diagramas

Bonny Rodrigues Martins 201522417
Walisson Gama Diniz 201521359
Rogério oliveira da Silva

Resumo O artigo aborda sobre diagramas da *UML,* com o objetivo de apresentar todos os diagramas existentes de modelagem e a sua importância no desenvolvimento de um produto computacional. A modelagem de software normalmente indica a construção de modelos gráficos que representam os artefatos dos componentes de software utilizados e os seus Inter-relacionamentos. Uma forma comum de modelagem de programas orientados a Objeto e através da linguagem unificada UML.A UML (Linguagem Unificada de Modelagem) é uma linguagem gráfica de modelagem para visualização, especificação, construção e documentação para desenvolver sistemas computacionais orientados a objeto, esta modelagem é apresentada através de diagramas.

Palavras-Chave: UML; Engenharia de Software; Modelagem de Software.

Abstract. The article discusses UML diagrams with the objective of presenting all the existing modeling diagrams and their importance in the development of a computational product. Software modeling usually indicates the construction of graphical models that represent the artifacts of the software components used and their Interrelationships. A common way of modeling object-oriented programs and using the unified UML language. The Unified Modeling Language (UML) is a graphing modeling language for visualization, specification, construction, and documentation to develop object-oriented computational systems. Presented through diagrams.

Keywords: UML; Software Engineering; Software Modeling.

1. Introdução

A UML (Linguagem Unificada de Modelagem) é uma linguagem gráfica de modelagem para visualização, especificação, construção e documentação para desenvolver sistemas computacionais orientados a objeto, esta modelagem é apresentada através de diagramas. É utilizada para uma melhor compreensão do sistema que será desenvolvido, especificar comportamentos internos que são as variáveis que interagem com o sistema (usuário e ambiente) e internos (sistema e suas especificações) e documentar as decisões tomadas (Vergilio, 2011).

Este artigo tem como objetivo apresentar os 13 diagramas da UML, com uma breve apresentação de todos e uma explicação de como são utilizados atualmente, dando ênfase aos 6 principais. Os diagramas apresentados neste artigo são diagrama de Casos de Uso, diagrama de Classes, diagrama de Objetos, diagrama de Pacotes, diagrama de Componentes, diagrama de Implantação, diagrama de Sequência, diagrama de Máquina de Estados, diagrama de Comunicação, diagrama de Atividades, diagrama de Visão Geral de Integração, diagrama de Temporização e diagrama de Estrutura Composta.

2. História de UML

No início da utilização do paradigma de orientação a objetos, diversos métodos foram apresentados para a comunidade. Chegaram a mais de cinquenta entre os anos de 1989 a 1994, porem a maioria deles cometeu o erro de tentar estender os métodos estruturados da época. Com isso os maiores prejudicados foram os usuários que não conseguiam encontrar uma maneira satisfatória de modelar seus sistemas. Foi a partir da década de 90 que começaram a surgir teorias que procuravam trabalhar de forma mais ativa com o paradigma da orientação a objetos. Diversos autores famosos contribuíram com publicações de seus respectivos métodos. (SAMPAIO, 2007)

Em outubro de 1994, começaram os esforços para unificação dos métodos. Já em outubro de 1995, Booch e Rumbaugh lançaram um rascunho do Método Unificado" unificando o Booch'93 e o OMT-2. Após isso, Jacobson se juntou a equipe do projeto e o "Método Unificado" passou a incorporar o OOSE. Em junho de 1996, os três amigos, como já eram conhecidos, lançaram a primeira versão com os três métodos - a versão 0.9 que foi batizada como UML (FOWLER, 2003)

3. Os diagramas da UML

A UML possui um total de treze diagramas. Eles são divididos em dois grupos: Diagramas Estruturais e Diagramas Comportamentais, sendo que os comportamentais possuem uma subdivisão chamada de Diagramas de Interação [Martinez, 2015]. Sendo estes os diagramas da UML. A UML define um número de diagramas que permite dirigir o foco para aspectos diferentes do sistema de maneira independente. Se bem usados, os diagramas facilitam a compreensão do sistema que está sendo desenvolvido.

3.1. Diagrama de Casos de Uso

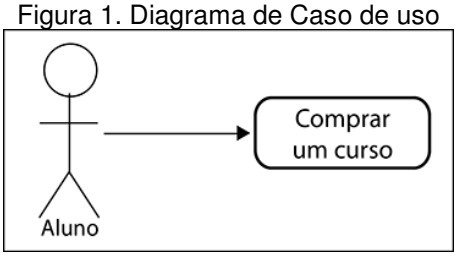
O diagrama de casos de uso especifica um conjunto de funcionalidades, através do elemento sintático "casos de uso", e os elementos externos que interagem com o sistema, através do elemento sintático "ator" (SILVA, 2007). Além de casos uso e atores, este diagrama contem relacionamentos de dependência, generalização e associação e são basicamente usados para fazer a modelagem de visão estática do caso de uso do sistema. Essa visão proporciona suporte principalmente para o comportamento de um sistema, ou seja, os serviços externamente visíveis que o sistema fornece no contexto de seu ambiente. Neste caso os diagramas de caso de uso são usados para fazer a modelagem do contexto de um sistema e fazer a modelagem dos requisitos de um sistema.

Para esse exemplo vamos começar imaginando que fomos contratados para desenvolver esse sistema, e precisamos conversar com nosso cliente para entender melhor os requisitos. O que vamos usar aqui é o que chamamos de diagramas de caso de uso.

Esse diagrama tenta deixar bem claro quais são as funcionalidades do sistema e quem irá fazer uso delas. O que um aluno pode no sistema? Comprar um curso, assistir um capítulo, resolver um exercício, etc. O que um professor pode fazer? Criar um curso, dar *feedback* em um exercício, e etc. Um caso de uso nos ajudará a modelar e deixar as coisas claras para a equipe.

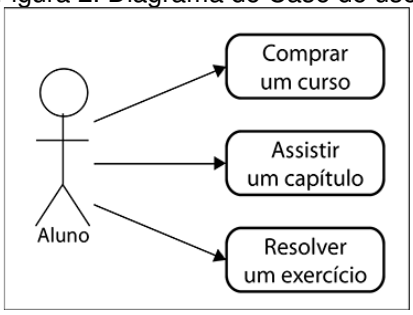
Como funciona o diagrama? Os atores (ou as pessoas que usam o nosso sistema) são representados por bonecos. As funcionalidades são representadas por círculos com texto dentro. Uma seta liga ambos.

Figura 1. Diagrama de Caso de uso



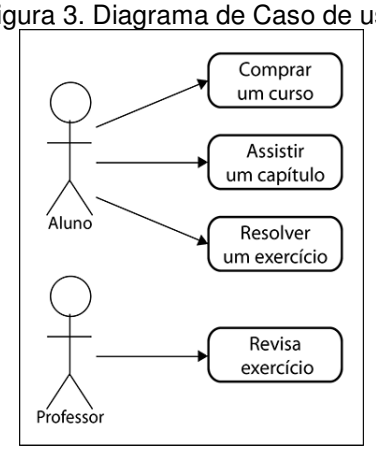
Podemos colocar mais funcionalidades, que também são feitas por alunos. Por exemplo:

Figura 2. Diagrama de Caso de uso



Veja que não há segredo. Basta representar cada funcionalidade como um círculo, ligando ao ator correspondente. Podemos ter mais de um ator no diagrama:

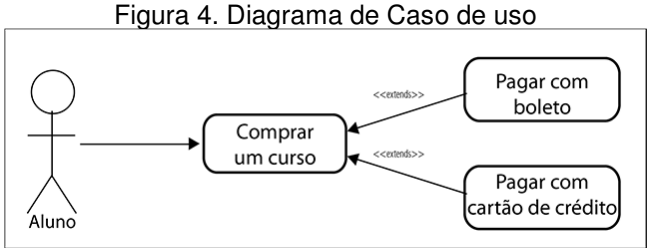
Figura 3. Diagrama de Caso de uso



Um dos grandes problemas de equipes que usam UML é que elas, por algum motivo, resolvem seguir a UML da maneira rígida. E a própria UML fala que você deve usar somente o que fizer sentido para o seu problema; nada também impede você de estender a UML e criar uma nova notação para simplesmente facilitar o entendimento da sua equipe.

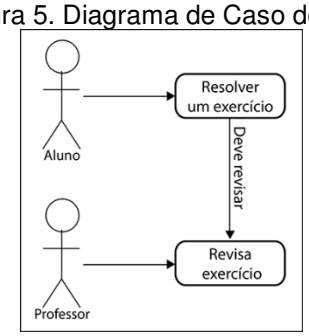
Vamos evoluir ainda mais nossos diagramas. Imagine que a funcionalidade de pagamento possa ser dividida em 2: pagamento por cartão de crédito e pagamento por boleto bancário. O fluxo das duas é diferente, mas ambas pertencem a funcionalidade maior "Comprar um Curso". Podemos representar isso na UML, através da notação ***extends***. Veja:

Figura 4. Diagrama de Caso de uso



Como dissemos anteriormente, nada nos impede de evoluir a UML para facilitar nossa comunicação. Podemos, por exemplo, fazer uma ligação entre "aluno resolver um exercício" e "professor revisar um exercício", afinal um vai acontecer somente depois do outro:

Figura 5. Diagrama de Caso de uso



Devemos usar a UML como se fosse uma linguagem. Nem sempre fazemos uso de todas as possíveis construções dela; usamos somente o necessário. Todo diagrama de caso de uso é geralmente acompanhado de um documento de texto que explica melhor aquela funcionalidade. Afinal, o diagrama que vimos até então apenas nos dá uma ideia das funcionalidades existentes no sistema.

Existem diversos *templates* para isso. A UML não diz qual você deve usar. Novamente, devemos usar aquele que faz mais sentido pra nos. Um *template* bastante comum é aquele que possui um título, uma lista de atores, as pré e pós-condições daquela funcionalidade, e um conjunto de possíveis fluxos dela. Esse documento é o que é utilizado pela equipe na hora de desenvolver.

3.2. Diagrama de Classes

Um diagrama de classes é um modelo fundamental de uma especificação orientada a objetos. Produz a descrição mais próxima da estrutura do código de um programa, ou seja, mostra o conjunto de classes com seus atributos e métodos e os relacionamentos entre classes. Classes e relacionamentos constituem os elementos sintáticos básicos do diagrama de classes (SILVA, 2007).

3.2.1. O que é um diagrama de classes? Para que serve?

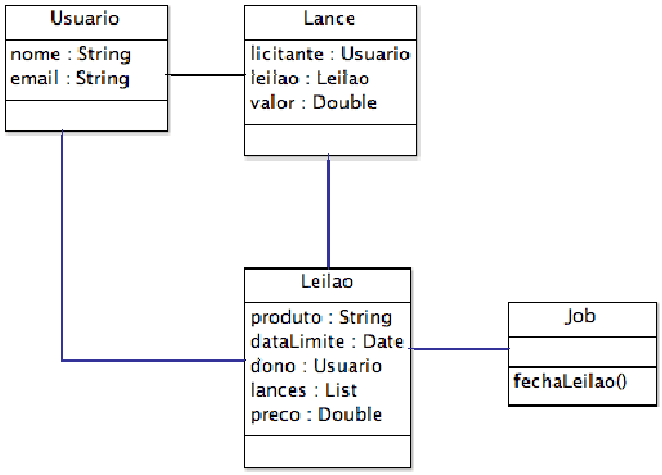
Um diagrama de classes modela as classes do sistema, bem como seus relacionamentos. Nele, definimos nossas classes, atributos, métodos, interfaces, heranças, e etc.

Abaixo como exemplo temos o diagrama de classe de um sistema de leilão:

- Um leilão tem produto, preço, data limite, dono, e uma lista de lances.

- Um usuário tem nome e e-mail.
- Um lance tem um usuário, um leilão e um valor.
- Um *job* tem o comportamento de fechar leilões vencidos.

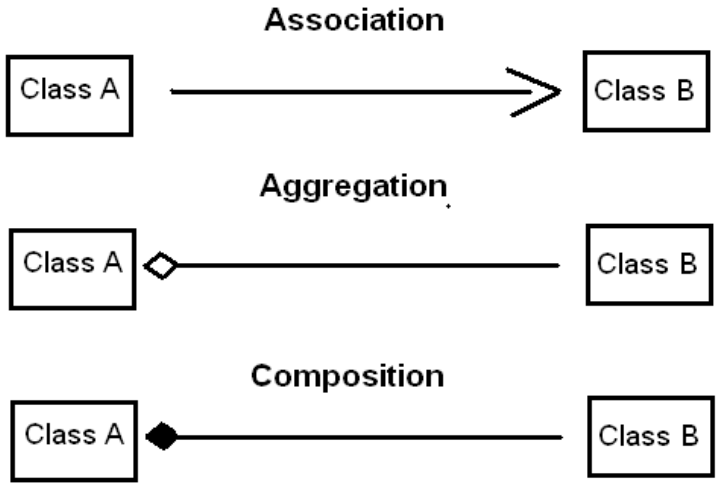
Figura 6. Diagrama de Classe



O usuário é dono de um leilão, mas também pode fazer lances e a classe Lance representa o relacionamento entre leilão e usuário. Para representarmos um relacionamento em um diagrama de classes usamos traços que ligam ambas as classes. Os traços também podem ter uma seta para deixar clara a direção do relacionamento.

Além disso, podemos distinguir os relacionamentos em associação, agregação e composição (entre outros como a própria herança).

Figura 7.Diagrama de Classe



A associação deixa claro, que uma classe possui uma referência da outra. Esse relacionamento também é chamado de *has-a* ou tem-um.

A agregação e composição também são associações! Uma associação pode ser uma agregação, no entanto as partes podem viver separadamente! Na composição o ciclo de vida é mais atrelado, um controla ou outro (um não existe sem o outro). Exemplos de composições:

- Casa-Quarto (quando a casa "morre", morre também o quarto);
- Contrato-Matricula (quando o contrato morre, também morrem as matriculas);
- Leilão-Lance;
- Curso-Capítulo.

Exemplos de agregação:

- Usuário-Lance (o usuário continua existindo, mesmo sem lance);
- Professor-Turma (ambos têm um ciclo de vida separado);
- Aluno-Curso.

A diferença entre agregação e composição fica realmente claro, quando olhamos no código. São detalhes da implementação e por isso não tão relevante nesse momento.

3.3. Diagrama de Objetos

O diagrama de objetos consiste em uma variação do diagrama de classes em que, em vez de classes, são representadas instancias e ligações entre instancias. (CLAIR 2010) A finalidade e descrever um conjunto de objetos e seus relacionamentos em um ponto no tempo. Contém objetos e vínculos e são usados para fazer a modelagem da visão de projeto estática de um sistema a partir da perspectiva de instancias reais ou prototípicas. Este mostra uma imagem de um sistema orientado a objeto sendo executado com seus objetos e atributos contendo valores e também a ligação entre eles.

O diagrama de objetos modela as instâncias das classes contidas no diagrama de classes, isto é, o diagrama de objetos mostra um conjunto de objetos e seus relacionamentos no tempo. Estes diagramas são importantes para construir os aspectos estáticos do sistema. Normalmente, são compostos por: objetos e vínculos.

Figura 8. Diagrama de Objetos

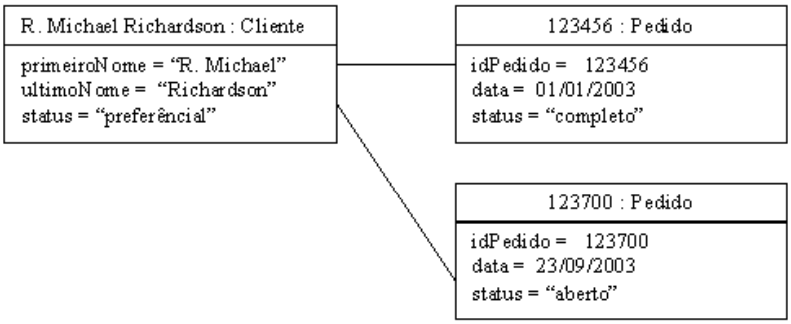


Figura 9. Diagrama de Objetos

O exemplo acima mostra um diagrama de objetos para o cliente Michael Richardson e seus dois pedidos na Virtual LTDA. O diagrama pode ser lido da seguinte maneira: O objeto R. Michael Richardson da classe Cliente está associado a ambos os objetos 123456 e 123700 da classe Pedido. Usa-se o diagrama de objetos para modelar a visão estática de um sistema. Ele mostra o retrato do sistema em determinado momento.

3.4. Diagrama de Pacotes

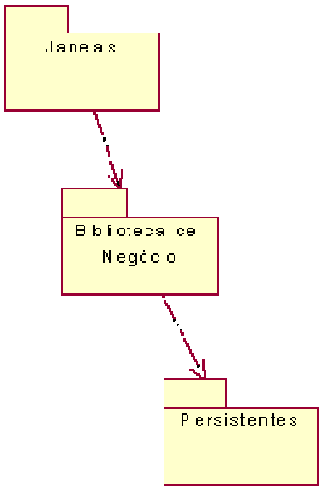
O pacote e um elemento sintático voltado a conter elementos sintáticos de uma especificação orientada a objetos. Esse elemento foi definido na primeira versão de UML para ser usado nos diagramas então existentes, como diagrama de

classes, por exemplo. Na segunda versão da linguagem, foi introduzido um novo diagrama, o diagrama de pacotes, voltado a conter exclusivamente pacotes e relacionamentos entre pacotes (SILVA, 2007). Sua finalidade e tratar a modelagem estrutural do sistema dividindo o modelo em divisões logicas e descrevendo as interações entre ele em alto nível.

Em muitos casos um único diagrama de classes pode ser exageradamente grande para representar todo o sistema. Assim é conveniente utilizar um elemento para organizar os modelos. Para isto utiliza-se o diagrama de pacotes. Um diagrama de pacotes pode ser utilizado em qualquer fase do processo de modelagem.

O diagrama de pacotes abaixo é uma proposta para a Virtual LTDA. Um diagrama de pacotes é composto de pacotes e relacionamentos entre pacotes. O critério para definir os pacotes é subjetivo e depende da visão e das necessidades do projetista. Este deve definir certa semântica e colocar os elementos similares e que tendem a serem modificados em conjunto num mesmo pacote. Como se vê na figura abaixo se pode usar os pacotes para mostrar a arquitetura do sistema.

Figura 10.Digrama de Pacotes



3.5. Diagrama de Componentes

O diagrama de componentes e um dos dois diagramas de UML voltados a modelar software baseado em componentes. Tem por finalidade indicar os componentes do software e seus relacionamentos. (CLAIR 2010) Este diagrama mostra os artefatos de que os componentes são feitos, como arquivos de código fonte, bibliotecas de programação ou tabelas de bancos de dados. As interfaces e que possibilitam as associações entre os componentes. Concluindo, tem por finalidade indicar os componentes de software que o sistema possuirá e também seus relacionamentos, onde são determinadas as configurações de hardware e também os componentes de software.

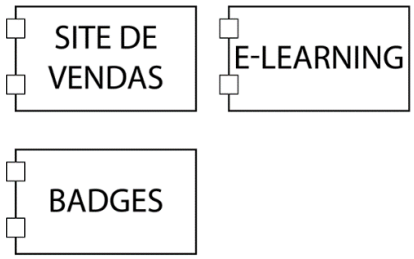
Imagine um sistema grande, onde temos uma aplicação web, que fala com diversos bancos de dados, comunica-se com diferentes serviços web, e etc. Nesses casos, é bastante importante que toda a equipe entenda como os sistemas se relacionam. É para isso que usaremos o diagrama de componentes**.**

Imagine que o sistema de *e-learning* tenha um site de vendas. A representação de um componente é assim:

Figura 11. Diagrama de componentes

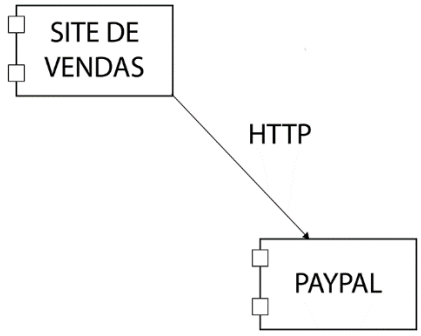
Temos também o sistema de e-learning em si, bem como o sistema que lida com as badges. Também vamos representá-los como componentes:

Figura 12 . Diagrama de componentes



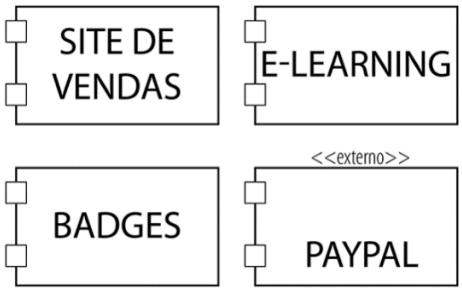
Temos também o sistema de pagamento, que no caso, é o Paypal. Vamos representá-lo como pagamento, e deixar bem claro que ele é um sistema externo, por meio de estereótipos:

Figura 13. Diagrama de componentes



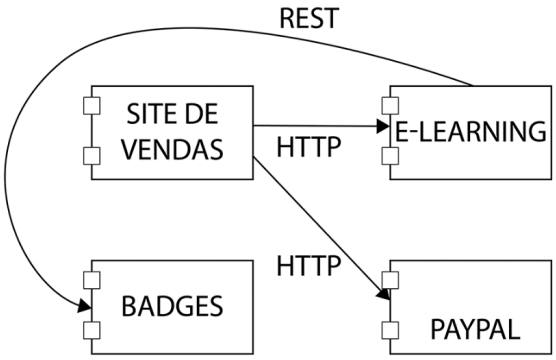
Estereótipos são bastante úteis. Como já visto anteriormente, ele nos possibilita passar informações sobre um determinado elemento. Vamos fazer a mesma coisa então com os outros componentes, deixando claro que temos aplicações web (WEB) e serviços web (WS):

Figura 14. Diagrama de componentes



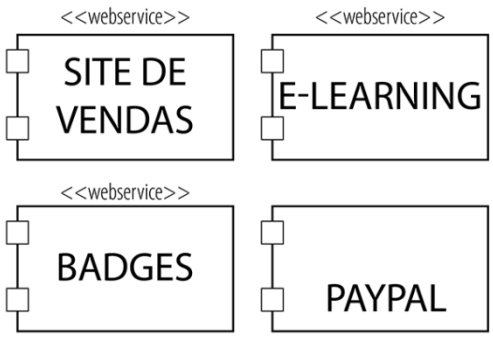
Com todos os componentes devidamente anotados, vamos começar a marcar a relação entre eles. Para isso, usaremos flechas, e deixamos claro como um componente se comunica com o outro (qual protocolo, ou coisa parecida). Por exemplo, o site de vendas comunica-se com o Paypal por meio de HTTP:

Figura 15. Diagrama de componentes



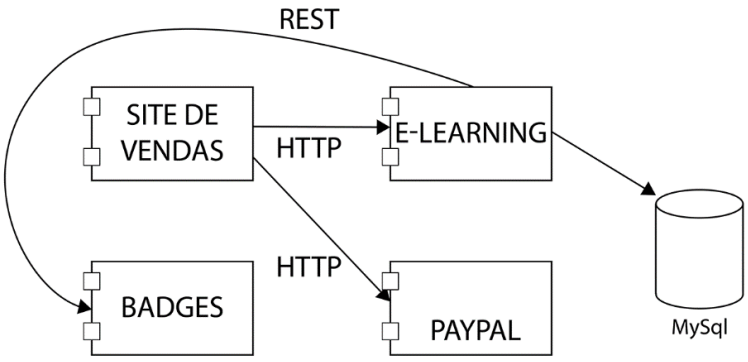
Vamos completar essas relações:

Figura 16. Diagrama de componentes



Podemos também representar nosso banco de dados. Geralmente usamos um símbolo um pouco diferente, que se parece com um banco de dados:

Figura 17. Diagrama de componentes



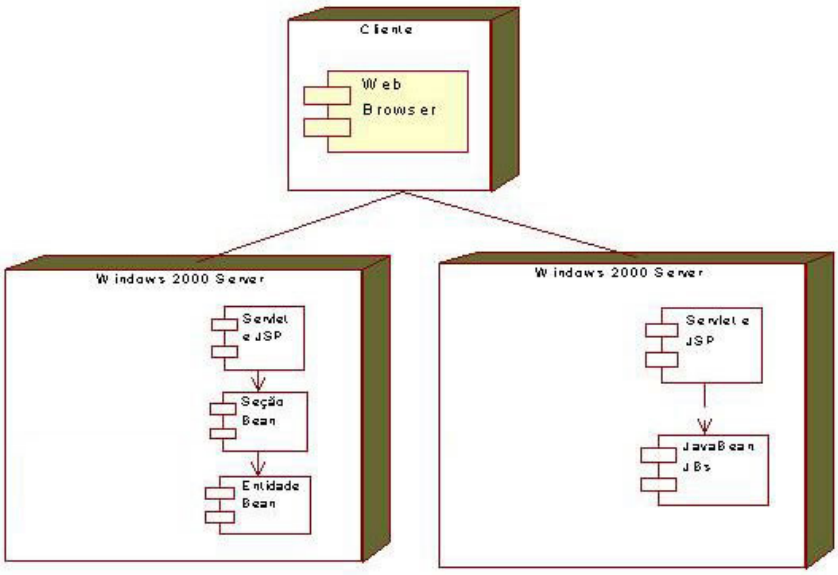
Veja então que temos cada parte do nosso sistema representado como um componente. É comum que esse diagrama tenha bem explicado como cada componente conversa com o outro, restrições, e etc. Lembre-se: a ideia do diagrama é facilitar a vida de quem irá lê-lo; quanto mais útil e informativo, melhor.

3.6. Diagrama de Implantação

O diagrama de utilização, também denominado diagrama de implantação, consiste na organização do conjunto de elementos de um sistema para a sua execução. O principal elemento deste diagrama e o nodo, que representa um recurso computacional. Podem ser representados em um diagrama tantos os nodos como instancias de nodos. O diagrama de implantação e útil em projetos onde há muita interdependência entre pedaços de hardware e software. (CLAIR 2010) É quem determina as configurações de hardware e características físicas que o sistema terá.

Os diagramas de implantação mostram a distribuição de hadware do sistema, identificando os servidores como nós do diagrama e a rede que relaciona os nós. Os componentes de software vão estar mapeados nestes nós. O diagrama de implantação é composto por: nós, relacionamentos entre nós. Esse diagrama mostra uma visão estática do funcionamento do sistema. São necessários quando o sistema é distribuído por vários processadores. Abaixo é mostrada uma proposta para o diagrama de implantação da Virtual LTDA.

Figura 18.Diagrama de implantação



Pode-se representar no diagrama de implantação componentes. Estes devem ser colocados em algum nó.

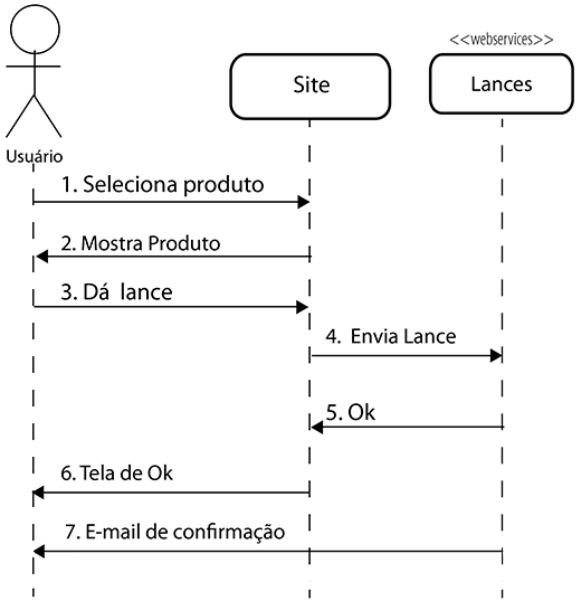
3.7. Diagrama de Sequência

O diagrama de sequência mostra a troca de mensagens entre diversos objetos, em uma situação especifica e delimitada no tempo. Coloca ênfase especial na ordem e nos momentos nos quais mensagens para os objetos são enviadas. Em diagramas de sequência, objetos são representados através de linhas verticais tracejadas (denominadas como linha de existência), com o nome do objeto no topo.

O eixo do tempo e também vertical, aumentando para baixo, de modo que as mensagens são enviadas de um objeto para outro na forma de setas com a operação e os nomes dos parâmetros (CLAIR 2010). Vejamos um exemplo, um diagrama de sequência para o seguinte fluxo:

- Usuário seleciona um produto no site
- Site mostra o produto para o usuário
- Usuário dá lance no site
- Site envia lance para serviço web de lances
- Serviço web confirma para o site
- Site exibe tela de OK para o usuário
- Serviço web dispara e-mail para usuário com as informações do lance
  Para ver uma possível solução clique no botão abaixo.

Figura 19. Diagrama de Sequência



3.8. Diagrama de Máquina de Estados

O diagrama de máquina de estados tem como elementos principais o estado, que modela uma situação em que o elemento modelado pode estar ao longo de sua existência, e a transição, que leva o elemento modelado de um estado para o outro.

O diagrama de máquina de estados vê os objetos como maquinas de estados ou autômatos finitos que poderão estar em um estado pertencente a uma lista de estados finita e que poderão mudar o seu estado através de um estimulo pertencente a um conjunto finito de estímulos. (CLAIR 2010) Ou seja, é utilizado para acompanhar as mudanças que aconteceram em um determinado objeto dentro de um processo que está sendo executado no sistema, sendo este um diagrama do tipo comportamental.

3.9. Diagrama de Comunicação

Os elementos de um sistema trabalham em conjunto para cumprir os objetos do sistema e uma linguagem de modelagem precisa poder representar esta característica. O diagrama de comunicação e voltado a descrever objetos interagindo e seus principais elementos sintáticos são “objeto” e “mensagem”. Corresponde a um formato alternativo para descrever interação entre objetos. Ao contrário do diagrama de sequência, o tempo não e modelado explicitamente, uma vez que a ordem das mensagens e definida através de enumeração.

Vale ressaltar que tanto o diagrama de comunicação como o diagrama de sequência são diagramas de interação. (CLAIR 2010) Este tem sua utilização voltada para descrever os objetos que estão interagindo e seus principais elementos sintáticos. Este diagrama corresponde a um formato diferenciado para descrever a interação entre objetos.

3.10. Diagrama de Atividades

O diagrama de atividades representa a execução das ações e as transições que são acionadas pela conclusão de outras ações ou atividades. Uma atividade pode ser descrita como um conjunto de ações e um conjunto de atividades. A diferença básica entre os dois conceitos que descrevem comportamento e que a ação e atômica, admitindo particionamento, o que não se aplica a atividade, que pode ser detalhada em atividades e ações (SILVA, 2007). Tem como objetivo

demonstrar o fluxo de atividades de um único processo e também mostrando como uma atividade depende da outra.

Um diagrama de atividade ilustra a natureza dinâmica de um sistema pela modelagem do fluxo de controle de atividade à atividade. Uma atividade representa uma operação em alguma classe no sistema que resulta em uma mudança no estado do sistema. Tipicamente, diagramas de atividades são usados para modelar fluxos de processos, processos de negócios ou operações internas.

O diagrama de atividades é similar a uma máquina de estados, mas tem um propósito diferente, o qual envolve capturar ações e seus resultados em termos de mudanças do estado do objeto. O diagrama de atividades é representado por um gráfico de atividades que mostram o fluxo de uma atividade para outra. Esse fluxo é mostrado através de transições, que são setas direcionadas, mostrando o caminho entre os estados de atividade (ação).

O diagrama acima, mostra o diagrama de atividades para a operação averiguar Credito na classe Pedido da Virtual LTDA. Note que a atividade “Preparar Mensagem de Credito” define o que fazer, mas não como fazer.

Um diagrama de atividades é normalmente composto pelos seguintes elementos: atividades (ações), estados de atividade (ação), transição, fluxo de objeto, estado inicial, estado final, branching, sincronização, raias.

3.11. Diagrama de Visão Geral de Integração

O diagrama de visão geral de interação e uma variação do diagrama de atividades, proposto na versão atual de UML. Seus elementos sintáticos são os mesmos do diagrama de atividades. As interações que fazem parte do diagrama de visão geral de interação podem ser referências a diagramas de interação existentes na especificação tratada. (CLAIR 2010) Sendo este um dos diagramas que compõe os diagramas de interação da UML. É uma variação do diagrama de atividades fornecendo uma visão geral dentro do processo da especificação tratada.

3.12. Diagrama de Temporização

O diagrama de temporização consiste na modelagem de restrições temporais do sistema. E um diagrama introduzido na segunda versão de UML, classificado como diagrama de interação. Este diagrama modela interação e evolução de estados. (CLAIR 2010) E por fim, este vem com o propósito de descrever a mudança de estado ou condição de uma instância de um objeto de uma classe ou seu papel durante este tempo.

3.13. Diagrama de Estrutura Composta

O diagrama de estrutura composta fornece meios de definir a estrutura de um elemento e de focaliza-la no detalhe, na construção e em relacionamentos internos. E um dos novos diagramas propostos na segunda versão de UML, voltado a detalhar elementos de modelagem estrutural, como classes, pacotes e componentes, descrevendo sua estrutura interna. O diagrama de estrutura composta introduz a noção de “porto”, um ponto de conexão do elemento modelado, a quem podem ser associadas interfaces. Também utiliza a noção de “colaboração”, que consiste em um conjunto de elementos interligados através de seus portos para a execução de uma funcionalidade especifica – recurso útil para a modelagem de padrões de projeto (SILVA, 2007). “Reflete a colaboração interna das classes para descrever a funcionalidade”. Sendo assim são as classes do sistema relacionadas entre si para executar uma função.

4. Conclusão

Embora a UML defina uma linguagem precisa, ela não é uma barreira para futuros aperfeiçoamentos nos conceitos de modelagem. O desenvolvimento da UML foi baseado em técnicas antigas e marcantes da orientação a objetos, mas muitas outras influenciarão a linguagem em suas próximas versões. Muitas técnicas avançadas de modelagem podem ser definidas usando UML como base, podendo ser estendida sem se fazer necessário redefinir a sua estrutura interna.

A UML está sendo a base para muitas ferramentas de desenvolvimento, incluindo modelagem visual, simulações e ambientes de desenvolvimento. Em breve, ferramentas de integração e padrões de implementação baseados em UML estarão disponíveis para qualquer um.

A UML integrou muitas ideias adversas, e esta integração acelera o uso do desenvolvimento de softwares orientados a objetos.

Referências

FOWLER, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. [S.l.: s.n.], 2003.

SAMPAIO, M. C. História de UML. http://www.dsc.ufcg.edu.br/sampaio: [s.n.], 2007.

SILVA, R. P. e. UML 2 em Modelagem Orientada a Objetos. Florian´ polis: Visual Books, 2007.

CLAIR, S. V. A história de UML e seus diagramas.

VERGILIO, Silva (2011) Introdução a UML.

OMG. Unified Modeling Language: diagram interchange. 2005.

OMG. Unified Modeling Language: superstructure. 2005.

OMG. Unified Modeling Language: infrastructure. 2006.