Curso de Sistemas de Informção
Disciplina: Análise e Projeto de Sistemas
Professor: José Maurício S. Pinheiro

AULA 3 – Processos e Modelagem de Sistemas

A modelagem de sistemas é o processo de desenvolvimento de modelos abstratos de um sistema, de maneira que cada modelo apresenta uma visão ou perspectiva diferente do sistema.
Desenvolver sistemas de software é a atividade de longo prazo, de criar um ou mais programas de computador de forma a atender necessidades específicas de um cliente ou grupo de clientes. No desenvolvimento de software são realizadas atividades de descoberta das necessidades e de criação do produto de software propriamente dito. Podemos dividir as atividades do processo de desenvolvimento em alguns grandes grupos:

- Atividades de Análise, cuja finalidade é descobrir “o que” deve ser feito;
- Atividades de Projeto, cuja finalidade é planejar “como” o software deve ser feito.
- Atividades de Implementação, cuja finalidade é produzir o produto de software de acordo com as especificações produzidas nas fases anteriores;
- Atividades de Controle de Qualidade, onde se incluem todas as atividades com objetivo de garantir a qualidade do produto, como testes e verificações.

De acordo com o processo de desenvolvimento escolhido, cada uma dessas atividades pode ser dividida em várias outras subatividades ou tarefas. Elas podem ser executadas de diferentes formas, em diferentes ordens. Também é possível que as atividades de análise e projeto sejam feitas de forma implícita, por exemplo, quando desenvolvemos o software unicamente por meio de protótipos.
Atualmente, a modelagem de sistemas se tornou a representação de um sistema usando algum tipo de notação gráfica, que hoje em dia quase sempre são baseadas em notações em Unified Modeling Language (UML). A modelagem de sistemas ajuda o analista a entender a funcionalidade do sistema e os modelos são usados para comunicação com os clientes.

1. Processos de Software

Parte da metodologia usada no planejamento de projetos de desenvolvimento de software trata do encadeamento das atividades técnicas de levantamento e especificação das necessidades dos usuários, especificação da solução (o sistema) dada pela equipe, construção dessa solução, testes, treinamento e colocação do sistema em produção. A esse conjunto de atividades e sua ordenação no tempo damos o nome de processo (de construção) de software.
Processo de software é, em outras palavras, um conjunto de atividades, métodos, práticas e transformações que pessoas empregam para definir, desenvolver e manter o software (Fig. 1).

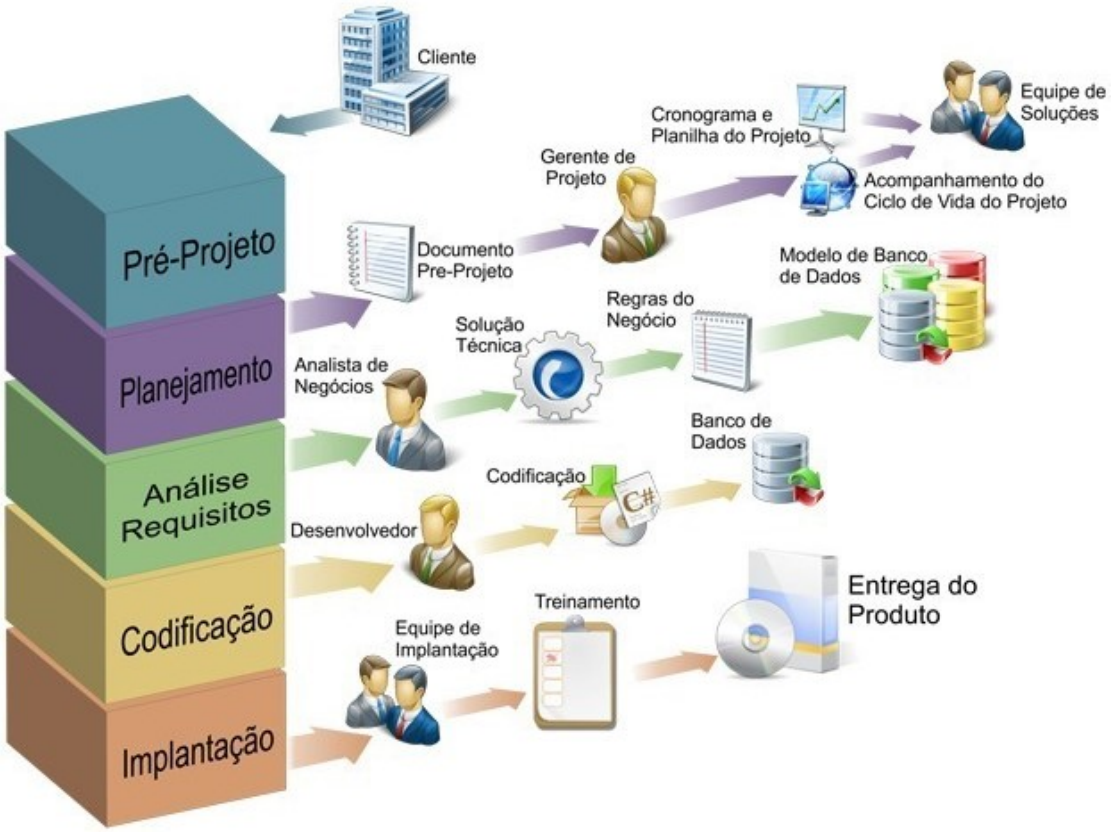

Figura 1 - Desenvolvimento de software

Oportuno lembrar que os produtos associados ao software, como plano do projeto, documentos do projeto, casos de teste, manuais do usuário etc., também fazem parte do produto software.

2. Modelagem de Sistemas

Modelo é a representação de algo, é uma abstração da realidade e representa uma seleção de características do mundo real que são relevantes para o propósito para o qual o modelo foi construído. Os modelos de sistemas existentes são usados durante a engenharia de requisitos. Eles ajudam a esclarecer o que o sistema existente faz e podem ser usados como uma base para discussão dos seus pontos fortes e fracos. O que pode levar a requisitos para um novo sistema.
Os modelos de um novo sistema são usados durante a engenharia de requisitos para ajudar a explicar os requisitos propostos para outros stakeholders do sistema. Os engenheiros de sistemas usam esses modelos para discutir propostas de projeto e para documentar o sistema para implementação.
Em um processo de engenharia dirigida a modelos, é possível gerar uma implementação do sistema completa ou parcial a partir do modelo do sistema.

2.1. Perspectivas de sistemas

É uma perspectiva externa, na qual se modela o contexto ou ambiente do sistema. Trata-se de uma perspectiva de interação, em que o analista modela as interações entre um sistema e seu ambiente ou entre os componentes de um sistema. Uma perspectiva estrutural, na qual se modela a organização de um sistema ou a estrutura dos dados processados pelo sistema. Também é uma perspectiva

comportamental, em que se modela o comportamento dinâmico do sistema e como ele responde a eventos.

3. Modelagem de Processos

O modelo de processo é o ponto central para que os participantes definam mudanças para melhoramento do processo ou mesmo um desenho completamente novo. Pode ser identificado se um processo é eficiente e eficaz ou mesmo antecipar sua complexidade, redundâncias e não conformidades.
Modelagem de Processos significa desenvolver diagramas (Diagramas de Processos) que mostram as atividades da empresa, ou de uma área de negócios, e a sequência na qual são executadas. Muitos negócios são relativamente complexos, assim um modelo poderá consistir em diversos diagramas, e o alvo da modelagem é ilustrar um processo completo, permitindo aos gestores, consultores e colaboradores melhorarem o fluxo e aperfeiçoarem o processo.
A modelagem de processos tem sido desenvolvida como uma tecnologia para descrever processos tais que eles possam ser entendidos e desenvolvidos com maior visibilidade organizacional. Dentro da área de modelagem de processos existem muitos métodos e notações que podem ser usados para descrever o processo sobre uma ótica mais detalhista. Estes métodos variam desde notações formais rigorosas (notação matemática), até notações mais gráficas. Cada um desses tipos de notações tem as suas vantagens e problemas.
Geralmente, as notações formais podem ser executadas em um computador como programas para se estudar em detalhes o comportamento dos processos. Contudo, o maior problema com estas notações é que elas são difíceis para apresentarem para outra pessoa a não ser para uma que seja experiente no assunto. Por conseguinte, é difícil validar os cenários do processo com os usuários. Por outro lado, notações gráficas são excelentes recursos para levantamento e apresentação, desde que elas possam ser compreendidas com relativa facilidade em um curto espaço de tempo. Contudo, elas não oferecem os benefícios do experimento rigoroso nos quais podem ser obtidos com notações mais precisas.
Os resultados da modelagem de um processo são essencialmente o acréscimo de valor para o cliente e redução de custos para a empresa. O que, consequentemente, conduz a empresa ao aumento de lucros. Um diagrama de modelo de processo de negócio é uma ferramenta, ou seja, um meio para se atingir um fim determinado e não um resultado de desempenho por si só.

3.1. Ferramentas CASE

Para a modelagem de sistemas utilizando a UML, usualmente empregamos ferramentas CASE (Computed-Aided Software Engineering – engenharia de software auxiliada por computador), apelidadas simplesmente de CASE (Fig. 2).
O que faz a diferença real no uso de ferramentas CASE para a modelagem de sistemas é o suporte organizado ao processo de modelagem, por conta da garantia que eles normalmente dão de que o modelo gerado está sintaticamente correto e de que os diversos diagramas que compõem o modelo estão consistentes entre si. Por exemplo, se dois determinados atores estão relacionados entre si por generalização/especialização em um diagrama, esse relacionamento é representado automaticamente pelo CASE em todos os demais diagramas do modelo onde eles aparecerem. Caso removamos esse relacionamento em um diagrama, todas as representações desse relacionamento nos demais diagramas serão automaticamente removidas. Além disso, CASE não permite que atribuamos um

identificador já existente a um novo elemento do modelo em um mesmo espaço de nomes. CASE também não deve permitir que estabelecer associações entre dois atores, o que é vedado pela UML.

Figura 2 - Arquitetura de conjunto de ferramentas CASE

3.2. Objetivo da Modelagem de Processos Software

A modelagem de processos de software tem sido utilizada na engenharia de software ao longo dos anos para melhor entender, gerenciar e controlar o processo de desenvolvimento. Contudo, a descrição dos processos do cliente oferece uma nova perspectiva aos engenheiros de software: a necessidade de diferentes abordagens e o uso de diferentes notações e técnicas. Assim, o principal objetivo da modelagem de processos é representar os processos de uma maneira clara e formal em diferentes níveis de abstração. A disponibilidade de modelos completos permite uma análise crítica das atividades existentes para definir melhorias e racionalizações dos processos.

3.3. Vantagens e Desvantagens da Utilização de Modelagem de Processos

Algumas vantagens e desvantagens na utilização de uma modelagem de processos:

Vantagens

- Bons modelos de processos são a chave para a boa comunicação;
- Se é alguma coisa nova que a empresa está planejando executar, o modelo pode ajudar a assegurar sua eficiência desde o início;
- Revelar anomalias, inconsistências, ineficiências e oportunidades de melhoria, permitindo à organização compreender melhor e auxiliar na reengenharia desses processos;
- Fornecer visão clara e uniformizada das atividades, suas razões e formas de execução;
- Utilizar o modelo como um meio para distribuição de conhecimento dentro da organização e treinar as pessoas, ajudando-as a conhecer melhor seus papéis e as tarefas que executam.

Desvantagens

- Maior ênfase à estrutura detalhada do processo e menor esforço na estrutura principal do Processo de Negócio;
- Ocultam a complexidade do trabalho;
- Dificuldade em expressar uma lógica complexa;
- Dificuldade em identificar qual parte é o cliente e qual parte é o patrocinador, podendo ocorrer comportamentos diferentes para processos de negócio distintos;
- Não fica claro se são dedicadas a criar novos processos ou analisar processos existentes.

3.4. Recursos para Modelagem e Construção de Sistemas

São cinco os recursos utilizados para a modelagem e construção de sistemas computacionais:

3.4.1. Abstração

Consiste em "esquecer" o que não interessa em determinado momento. Inicia a abordagem no chamado nível conceitual, quando estamos interessados em aspectos de mais alto nível, mais gerais, ou seja, do domínio do problema.
No início do desenvolvimento de um sistema não devemos pensar nos detalhes, nas questões de tecnologia, nos aspectos ditos físicos. Devemos nos concentrar no que o usuário realmente precisa em termos de informação, ou seja, devemos nos concentrar na descrição do problema.
No extremo oposto, ou seja, imediatamente antes de escrevermos os programas, devemos ter todos os detalhes definidos, incluindo o desenho do banco de dados na tecnologia escolhida (fabricante do sistema de gerenciamento do banco de dados – SGBD, versão etc.), os nomes das classes, métodos e atributos e aspectos da linguagem de programação, entre outros, enfim, tudo que é necessário para a programação. Nesse nível, chamado nível de implementação, a abstração (o "esquecimento") é zero. No meio do caminho, entre o nível conceitual e o nível de implementação, estamos no nível de especificação, que recebe esse nome por estarmos exatamente especificando a solução que estamos construindo para o problema.
O nível de especificação começa assim que adicionamos o primeiro aspecto de nossa solução à especificação do problema e termina quanto todos os detalhes dessa solução estão definidos.

3.4.2. Rigor

Consiste em adotar uma abordagem metódica e controles estabelecidos a priori. A engenharia de software estabelece que o ciclo de vida do software é gerido por meio de processos bem comportados, envolvendo a definição, antes do início do ciclo de desenvolvimento, das etapas do processo de desenvolvimento e produção (incluindo manutenção) e seus pontos de controle (os marcos), dos níveis de qualidade, dos custos e dos prazos. Assim, são três os ingredientes que dão suporte à engenharia de software: o planejamento, a metodologia e a disciplina.
O planejamento trata da definição das etapas, da sequência de realização dessas etapas, do estabelecimento das metas e dos recursos que serão utilizados, da definição da metodologia, das ferramentas e de tudo o mais que trata do que fazer no sistema e como fazer.

A metodologia é o conjunto de procedimentos e técnicas que serão usadas, normalmente já estabelecidas por experiência anterior, pela academia ou pela instituição a qual estamos ligados.
A disciplina é o esforço que precisa ser desenvolvido no desenrolar do processo, no sentido de cumprir rigorosamente o planejamento, de acordo com a metodologia escolhida.

3.4.3. Formalismo

É expressado por meio do uso de linguagens de especificação precisas, usualmente gráficas, como diagramas de fluxos de dados (DFD), diagramas de entidades e relacionamentos (DER), diagramas da UML etc. O formalismo reduz a ambiguidade da linguagem de descrição, facilitando e tornando precisa a comunicação entre desenvolvedores e usuários e entre os membros da equipe de desenvolvimento.

3.4.4. Divisão e conquista

Também chamada de divisão no domínio, consiste em dividir o problema em partes pequenas e, idealmente, independentes, de forma a poder tratar cada parte mais facilmente. A expectativa é que, dando solução para cada uma dessas partes, podemos compor as soluções para obter a solução do problema original, maior.

3.4.5. Organização hierárquica

Ou divisão no conceito, consiste, tal qual a divisão no domínio, em dividir o problema em partes pequenas, dessa vez no conceito ou, em outras palavras, em níveis de abstração cada vez menores. A ideia é obter partes conceitualmente simples o suficiente de forma a descrever o problema e dar solução a ele mais facilmente. A expectativa também é poder compor as soluções das partes do problema, obtendo a solução de todo o problema.

3.5. Análises Estruturada e Essencial

A análise estruturada empregava bem o recurso da organização hierárquica dos sistemas por meio dos DFD e suas "explosões". Os métodos existentes estabelecem uma notação rigorosa e um processo formal para derivação dos modelos do sistema, ou seja, rigor e formalismo são, com isso, bem aplicados. Embora já houvesse a recomendação para tratar, na fase de análise, apenas os aspectos lógicos, havia a dificuldade de promover a abstração, provendo um isolamento entre as características tecnológicas do problema e o modelo conceitual. Com isso, a separação entre o lógico e o físico não era clara e a experiência do analista contava muito. Faltava também uma técnica para dividir o modelo em partes tão independentes quanto o possível, de forma a aplicar sistemática e efetivamente os recursos de divisão e conquista.
A análise essencial veio a seguir, com o intuito de resolver esses problemas. Com o uso dos conceitos da "tecnologia perfeita", as questões físicas eram sistematicamente postas de lado na fase de análise, e a aplicação do recurso de abstração era garantida. Além disso, por meio dos DFD particionados por eventos, a divisão do sistema em partes independentes era também garantida (Fig. 3).

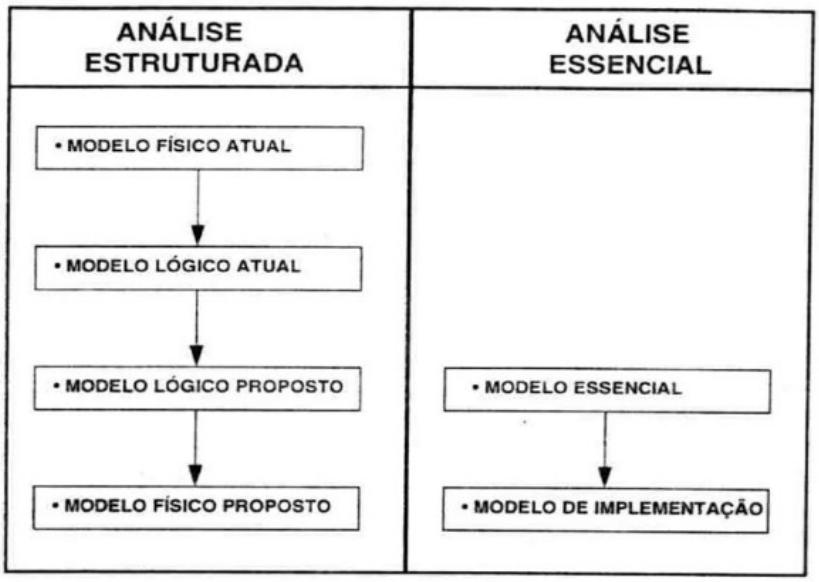

Figura 3 - Análises estrutura e essencial

3.6. Análise e Modelagem Orientadas a Objetos

A maioria dos métodos utilizados em ambientes de desenvolvimento de software se baseia em uma decomposição funcional e/ou controlada por dados dos sistemas. Estas abordagens se diferem em diversos aspectos das abordagens que adotam metodologias orientadas a objetos, onde dados e funções são altamente integrados. Por exemplo, com a análise essencial, os engenheiros de software poderiam, então, ter se dado por satisfeitos se não fossem dois outros aspectos que os incomodavam bastante:

- A separação entre dados e processos nos modelos. Os diagramas de fluxo de dados especificavam exclusivamente processos, e os diagramas de entidades e relacionamentos especificavam exclusivamente os dados, não sendo possível estabelecer na notação a associação existente entre os processos e os dados que processavam;
- A descontinuidade entre as fases de análise e projeto. O início do projeto obrigava que os analistas e projetistas aplicassem regras de derivação para transformação dos modelos de análise em modelos de projeto, obrigando-os a usar linguagens diferentes em suas especificações, dependendo do tipo de sistema que estava sendo modelado. Essas questões motivaram a busca por um novo paradigma cujos benefícios se aliassem às conquistas realizadas, até então, no intuito de transformarem o desenvolvimento de sistemas em uma atividade de engenharia.

A Figura 4, apresenta um comparativo entre as análises estruturada e essencial e a Análise orientada a Objetos.

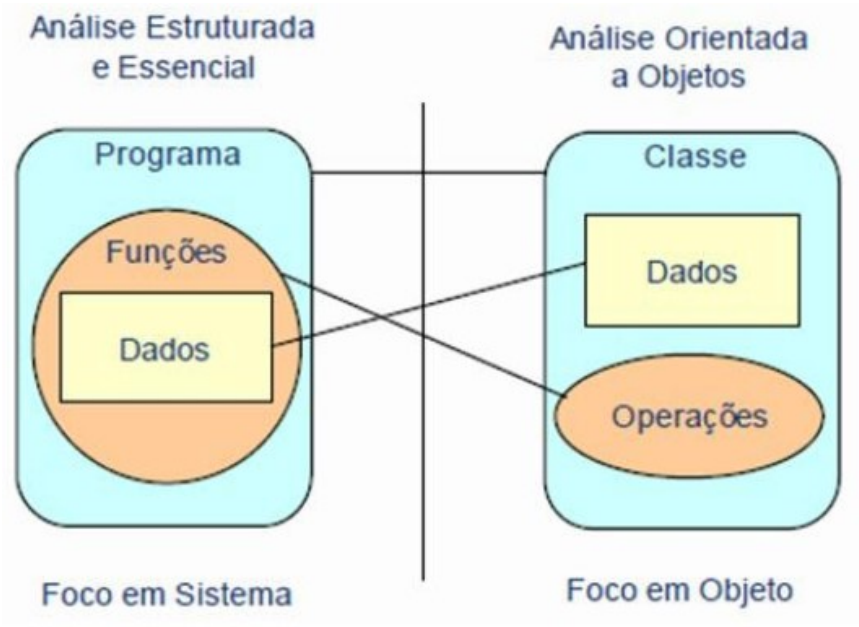

Figura 4 - Comparação entre análises estruturada e essencial e análise orientada a objetos

A análise e modelagem orientados a objetos (Object Oriented Analysis and Design) permite reunir, em um mesmo conceito e no mesmo modelo, dados e operações. Consiste na construção de módulos independentes ou objetos que podem ser facilmente substituídos, modificados e reutilizados. Ela retrata a visão do mundo real como um sistema de objetos cooperativos e colaborativos. Neste caso, o software é uma coleção de objetos discretos que encapsulam dados e operações executadas nesses dados para modelar objetos do mundo real. A classe descreve um grupo de objetos que têm estruturas semelhantes e operações similares.
A filosofia Orientada a Objetos é muito similar ao mundo real e, portanto, vem ganhando popularidade pois os sistemas aqui são vistos como um conjunto de objetos que interagem assim como no mundo real. Para implementar este conceito, a programação estruturada baseada em processos não é utilizada; em vez disso, os objetos são criados usando estruturas de dados. Assim como toda linguagem de programação oferece vários tipos de dados, da forma similar, no caso dos objetos certos tipos de dados são pré-definidos.
A abordagem orientada a objetos possibilita uma melhor organização, versatilidade e reutilização do código fonte, o que facilita atualizações e melhorias nos programas. É caracterizada pelo uso de classes e objetos e de outros conceitos:

- **Classes** – são espécies de montadoras de objetos, que definem suas características como, quais funções são capazes de realizar e quais os atributos que o objeto possui. Essa forma de programar permite ao usuário resolver problemas utilizando conceitos do mundo real;
- **Objeto** - é uma instancia gerada a partir de uma classe. Um objeto é identificado a partir dos métodos e dos atributos que possui;
- **Encapsulamento** - é o ato de esconder do usuário os processos internos de um objeto, classe ou método;
- **Herança (e Polimorfismo)** - característica que permite a determinada classe herdar as características de outra classe. Ou seja, a classe descendente adquire todos os métodos e atributos da classe pai.
- **Métodos** - são as funções que objeto pode realizar;
- **Atributo** - é tudo que um objeto possui como variável.

3.7. UML - Unified Modeling Language

A UML serve para construir modelos concisos, precisos, completos e sem ambiguidades, tendo, de maneira geral, as seguintes características:

- Modela os aspectos estruturais (estáticos) e comportamentais (dinâmicos) do sistema, ou seja, pode especificar os conceitos do negócio e seus relacionamentos (invariantes com o tempo) e os estados, sequências de atividades e de colaborações (aspectos que contemplam a dimensão temporal, ou seja, que variam conforme o tempo passa). A UML provê elementos de notação para modelar dados, funções de transformação dos dados e as restrições aplicáveis aos dados e às funções, como regras de negócio, por exemplo. Essas características são necessárias à produção de bons modelos.
- Provê uma linguagem que permite o entendimento e a utilização por humanos e a leitura por máquinas. Além dos elementos gráficos da notação que são compreensíveis aos humanos (que conheçam a linguagem), a UML conta com mecanismo padronizado para mapeamento entre a representação gráfica do modelo e a sua representação textual em XML (Extensible Markup Language – Linguagem Extensível de Marcação). A representação textual facilita o intercâmbio do modelo entre ferramentas de modelagem de fabricantes diferentes e possibilita a exportação desses dados para outras ferramentas, com finalidades diversas.
- Provê elementos de notação para modelar todos os tipos de sistemas de computação.
- Permite a modelagem do conceito ao artefato executável, utilizando técnicas Orientadas a Objetos. Usando os mesmos elementos da notação, podemos modelar desde os aspectos do negócio associados a níveis de abstração maiores até os níveis de implementação, associados a níveis de abstração "zero" (nenhuma abstração). Podemos especificar, portanto, negócios e sistemas com o uso de uma única linguagem, o que permite, quando necessário, a transição natural entre modelos de negócio e modelos de sistema.
- A linguagem é extensível e adaptável a necessidades específicas; palavras-chave permitem que se modifique a semântica de elementos da linguagem. Com o uso de palavras-chave é possível manter um conjunto relativamente reduzido de elementos gráficos da notação, porém permitindo a adaptação da UML para uso em modelagem em domínios.
- Contempla as necessidades de produção de modelos pequenos e simples a grandes e complexos. A UML possui diversos conectores e contêineres, o que permite dividir os modelos em agrupamentos pequenos no domínio e em níveis de abstração, de forma a torná-los compreensíveis independentemente da complexidade (se devida ao tamanho do que está sendo estudado ou ao domínio que está sendo tratado).
- Modela processos manuais ou automatizados, estes independentemente da tecnologia que usam.
- É uma linguagem para visualização do modelo, facilitando o entendimento pelas equipes de análise de negócio, desenvolvimento de sistemas e pelos clientes.
- Serve para construir código de computador, embora não seja uma linguagem de programação de computadores.

- A UML é o padrão *de facto* usado em análise e projeto de sistemas de informática orientados a objetos.
- Está em evolução constante e são publicadas novas versões resultantes das discussões entre membros de diversas áreas da indústria e meio acadêmico.

A UML foi elaborada por Grady Booch, James Rumbaugh e Ivar Jacobson e sua primeira versão foi lançada em janeiro de 1997. A UML, sendo uma linguagem gráfica, conta ainda com a facilidade de emprego, pois os elementos da notação são símbolos gráficos que podem ser compostos com a ajuda de ferramentas gráficas interativas que garantem a correção e a consistência do modelo (Fig. 5).

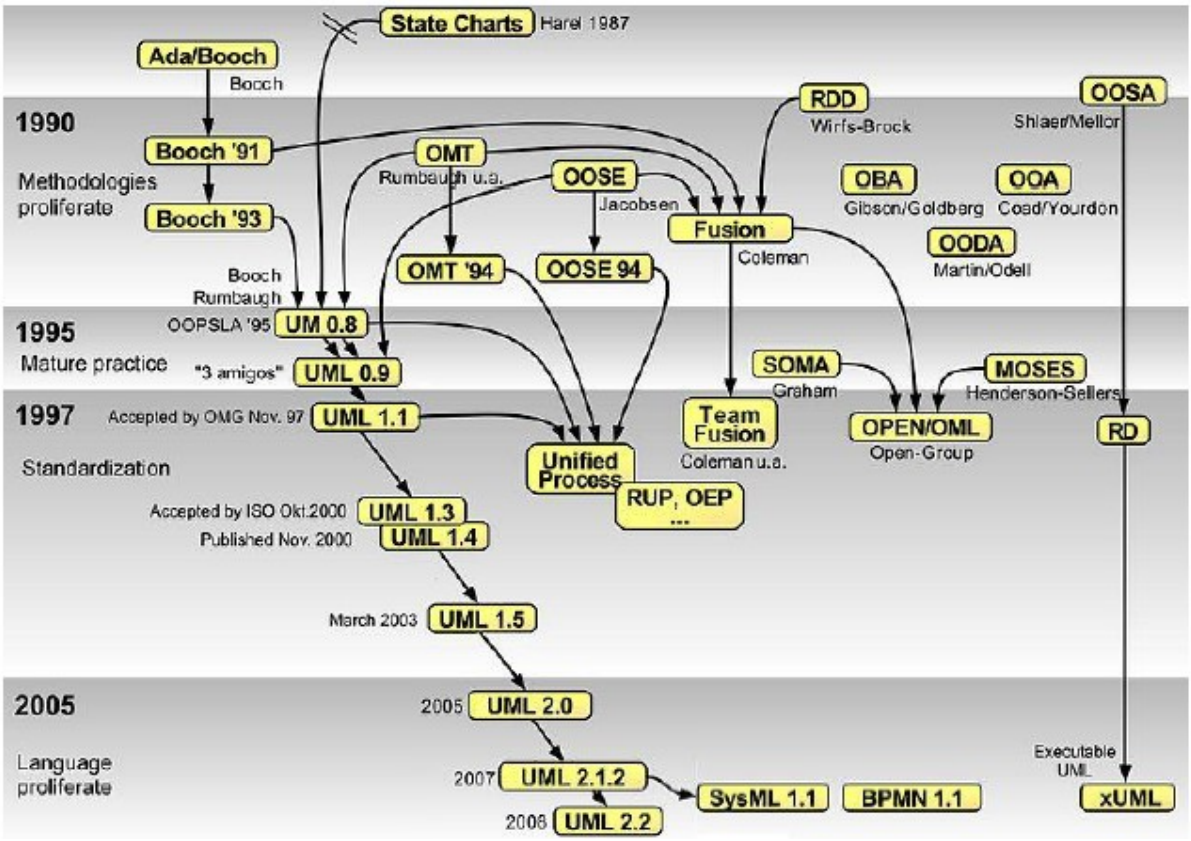

Figura 5 - Evolução da UML

Como consequência da extensão de sua especificação e da aplicabilidade em uma gama ampla de domínios, a UML é muitas vezes considerada intimidativa. No entanto, com um subconjunto mais reduzido de conceitos e elementos da notação contamos com um ferramental suficiente para tratar boa parte (senão a totalidade) das necessidades de modelagem de sistemas encontradas no dia a dia. A UML não é apenas uma linguagem de modelagem de software, mas também pode ser empregada em diversas outras áreas de conhecimento (Fig. 6):

- Fluxo de trabalho no sistema legal;
- Estrutura de sistemas de saúde;
- Projetos de hardware.

Figura 6 - UML é uma linguagem gráfica

Vantagens:

- Desenvolvimento de programas de forma rápida, eficiente e efetiva;
- Revela a estrutura desejada e o comportamento do sistema;
- Permite a visualização e controle da arquitetura do sistema;
- Melhor entendimento do sistema que está sendo construído e gerenciamento de riscos.

De acordo com a UML, deve-se ter:

- Visão de casos de uso, expondo as exigências do sistema;
- Visão de projeto, capturando o vocabulário do espaço do problema e do espaço da solução;
- Visão do processo, modelando a distribuição dos processos e linhas do sistema;
- Visão de implementação, dirigindo-se à realização física do sistema;
- Visão de distribuição, focando na edição da engenharia de sistema.

Cada uma dessas visões pode ter aspectos estruturais, assim como comportamentais. Juntas, essas visões representam a especificação completa de um sistema computacional (Fg. 7).

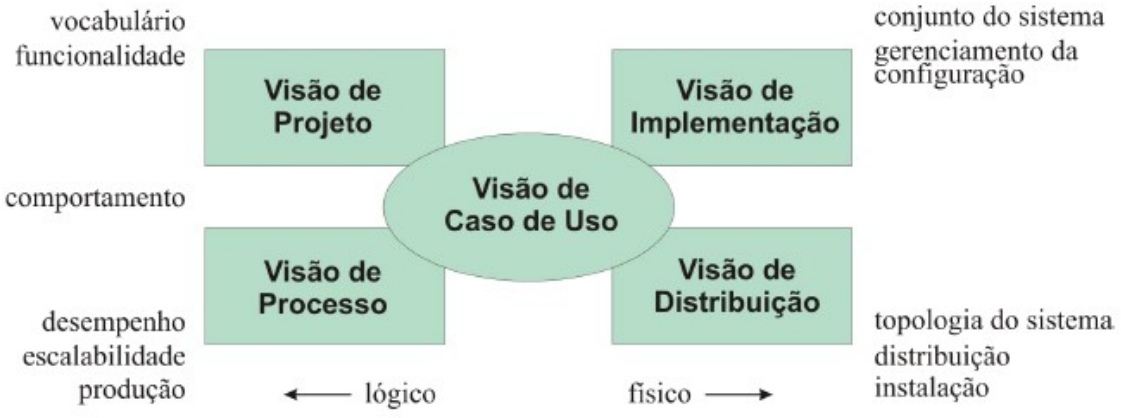

Figura 7 - Visão de caso de uso