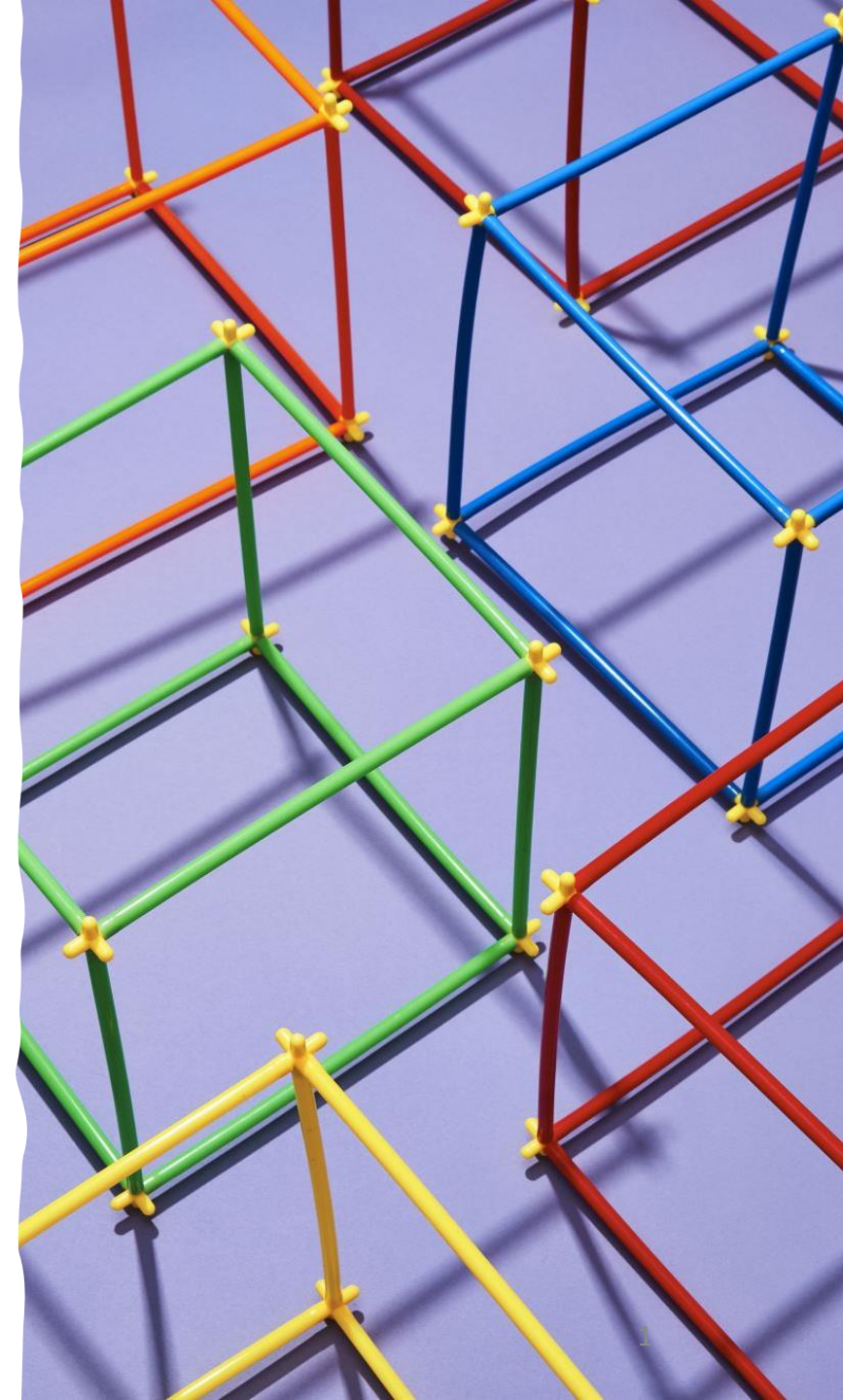


# Modelagem de Sistemas

---

Prof. Ma. Marina Girolimetto



# Modelagem de Sistemas



A modelagem de sistemas é um processo de desenvolvimento de modelos abstratos de um sistema, em que cada modelo apresenta uma visão ou perspectiva diferente desse sistema.



Representa um sistema usando algum tipo de notação gráfica baseada nos tipos de diagrama em UML (Unified Modeling Language - Linguagem de Modelagem Unificada).



Os modelos são utilizados durante o processo de engenharia de requisitos, para ajudar a derivar os requisitos detalhados de um sistema; durante o processo de projeto.



Utilizada para descrever o sistema aos engenheiros que estão implementando o sistema; e depois da implementação, para documentar a estrutura e operação do sistema.



Um modelo não é uma representação completa do sistema. Propositamente, ele deixa de fora alguns detalhes para facilitar a compreensão.



O modelo é uma abstração do sistema que está sendo estudado, e não sua representação alternativa. Uma abstração deliberadamente simplifica o projeto de um sistema e seleciona as características mais relevantes.

# Os modelos gráficos podem ser usados de três modos diferentes:

- 
- **Como forma de estimular e focar a discussão sobre um sistema existente ou proposto.** Os modelos podem ser incompletos, contanto que cubram os pontos-chave da discussão.
- 
- **Como forma de documentar um sistema existente.** Quando os modelos são utilizados como documentação, eles não precisam ser completos — já que podem ser usados apenas para documentar algumas partes de um sistema.
- 
- **Como uma descrição detalhada do sistema que pode ser usada para gerar a implementação deste.** Devem ser completos e corretos.
-



# UML

- A *Unified Modeling Language* (UML) é um conjunto de 14 tipos diferentes de diagramas que podem ser usados para modelar sistemas de software.
- A UML é aceita universalmente como abordagem padrão para desenvolver modelos de sistemas de software.

# Introdução à UML

- É uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos.
- Essa linguagem é atualmente a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de software.
- A UML não é uma linguagem de programação, e sim uma linguagem de modelagem, ou seja, uma notação cujo objetivo é auxiliar os engenheiros de software a **definirem as características do sistema, 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.**

# Introdução à UML

- Tais características podem ser definidas por meio da UML antes de o software começar a ser realmente desenvolvido.
- Além disso, a UML é totalmente independente, assim, ela **pode ser utilizada por qualquer processo de desenvolvimento ou mesmo da forma que o engenheiro de software considerar mais adequada.**

# Breve Histórico da UML

- Em 1996, saiu a primeira versão da UML propriamente dita. Tão logo a primeira versão foi lançada, muitas empresas atuantes na área de modelagem e desenvolvimento de software passaram a contribuir para o projeto, fornecendo sugestões para melhorar e ampliar a linguagem.
- Em 1997, a UML foi adotada pela OMG (Object Management Group ou Grupo de Gerenciamento de Objetos), como uma linguagem-padrão de modelagem.
- A versão 2.0 da linguagem foi oficialmente lançada em julho de 2005. Atualmente a UML encontra-se na versão 2.5. Essa última versão foi lançada com o objetivo de simplificar a estrutura da linguagem.
- **A documentação oficial da UML pode ser consultada no site da OMG em [www.omg.org](http://www.omg.org) ou mais exatamente em [www.uml.org](http://www.uml.org).**



# Por que Modelar Software?

## **Qual a real necessidade de se modelar um software?**

- Muitos “profissionais” podem afirmar que conseguem determinar todas as necessidades de um sistema de informação de cabeça e que sempre trabalharam assim.

## **Qual a real necessidade de se projetar uma casa? Um pedreiro experiente não é capaz de construí-la sem um projeto?**

- Isso pode ser verdade, mas a questão é muito mais ampla, envolvendo fatores complexos, como elicitação e análise de requisitos, projeto, prazos, custos, documentação, manutenção e reusabilidade, entre outros.
- Existe uma diferença gritante entre construir uma pequena casa e construir um prédio de vários andares.

# Por que Modelar Software?

- Grandes projetos não podem ser modelados de cabeça, nem mesmo a maioria dos pequenos projetos pode sê-lo, exceto, talvez, aqueles extremamente simples.
- Por mais simples que seja, **todo sistema deve ser modelado antes de se iniciar sua implementação, entre outras coisas, porque os sistemas de informação frequentemente costumam ter tendência a “crescer”, isto é, aumentar em tamanho, complexidade e abrangência.**
- Alguns profissionais costumam afirmar que sistemas de informação são “vivos” porque nunca estão completamente finalizados.

# Por que Modelar Software?






- **Os sistemas de informação estão em constante mudança**, tais mudanças são devidas a diversos fatores, como:
  - Os clientes desejam constantemente modificações ou melhorias no sistema.
  - O mercado está sempre mudando, o que força a adoção de novas estratégias pelas empresas e, conseqüentemente, de seus sistemas.
  - O governo frequentemente promulga novas leis e cria novos impostos e alíquotas ou, ainda, modifica as leis, os impostos e as alíquotas já existentes, o que acarreta a manutenção no software.
- Assim, um sistema de informação precisa ter uma documentação detalhada, precisa e atualizada para ser mantido com facilidade, rapidez e correção, sem produzir novos erros ao corrigir os antigos.






# Modelo de Software – Uma Definição



- A modelagem de um software implica criar modelos de software, mas o que é realmente um modelo de software?
- **Um modelo de software captura uma visão de um sistema físico, é uma abstração do sistema com um certo propósito, como descrever aspectos estruturais ou comportamentais do software.**
- Esse propósito determina o que deve ser incluído no modelo e o que é considerado irrelevante.
- Assim, um modelo descreve completamente aqueles aspectos do sistema físico relevantes ao propósito do modelo, no nível apropriado de detalhe.

# Elicitação e Análise de Requisitos

- Um método de desenvolvimento de software, se divide em quatro fases:
  - **Concepção**, em que é feita a elicitação de requisitos inicial e determina-se a viabilidade de desenvolver o software;
  - **Elaboração**, em que são feitos a análise dos requisitos e o projeto do software;
  - **Construção**, em que o software é implementado e testado;
  - **Transição**, em que o software será implantado.

- 
- Após a elicitación de requisitos, passa-se à fase em que as necessidades apresentadas pelo cliente são analisadas. Essa etapa é conhecida como **análise de requisitos**.
  - Aqui, o engenheiro examina os requisitos enunciados pelos usuários-chave ou stakeholders verificando se estes foram especificados corretamente e se foram realmente bem compreendidos.
  - A partir da etapa de análise de requisitos são determinadas as reais necessidades do sistema de informação. A grande questão é: **como saber se as necessidades dos clientes e usuários-chave foram realmente bem compreendidas?**
  - Durante a análise de requisitos, uma linguagem de modelagem auxilia a levantar questões que não foram concebidas durante as entrevistas iniciais.
- 
- 
- 
- 

- 
- Tais questões devem ser sanadas o quanto antes, para que o projeto do software não tenha que sofrer modificações quando seu desenvolvimento já estiver em andamento, o que pode causar significativos atrasos no desenvolvimento do software, sendo por vezes necessário reiniciar o projeto do começo.
  - Outro grande problema encontrado durante as entrevistas consiste no fato de que, na maioria das vezes, os usuários não têm realmente certeza do que querem e não conseguem enxergar as reais potencialidades de um sistema de informação. **Em geral, os engenheiros de software precisam sugerir inúmeras características e funções do sistema que o cliente não sabia como formular ou sequer havia imaginado.**
  - Esses profissionais precisam reestruturar o modo como as informações são geridas e utilizadas pela empresa e apresentar maneiras de combiná-las e apresentá-las de maneira que possam ser mais bem aproveitadas pelos usuários, auxiliando-os na tomada de decisões.
- 
- 
- 
- 

- 
- 
- É fundamental trabalhar bastante o aspecto social da implantação de um sistema informatizado na empresa, pois muitas vezes a resistência não é tanto da gerência, mas dos usuários finais, que serão obrigados a mudar a forma como estavam acostumados a trabalhar e aprender a utilizar uma nova tecnologia.
  - **Deve-se, assim, procurar destacar como o novo sistema melhorará e facilitará o trabalho desses usuários.**


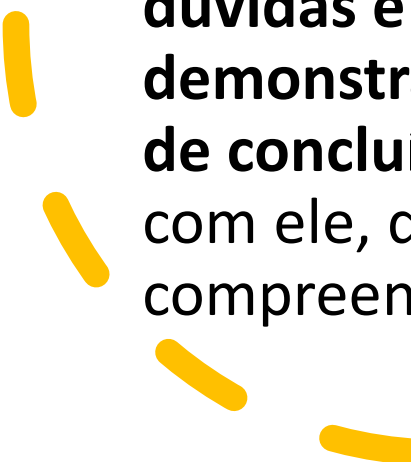







# Prototipação

- **A prototipação é uma técnica bastante popular e de fácil aplicação que permite validar se os requisitos do software foram compreendidos corretamente e se a proposta representada pelo protótipo satisfará realmente as necessidades do cliente.**
- Essa técnica consiste em desenvolver rapidamente um “rascunho” do que seria o sistema de informação quando estivesse finalizado.
- Um protótipo normalmente apresenta pouco mais do que a interface do software a ser desenvolvido, ilustrando como as informações seriam inseridas e recuperadas no sistema, apresentando alguns exemplos com dados fictícios de quais seriam os resultados apresentados pelo software, principalmente em forma de relatórios.

# Prototipação

- **A utilização de um protótipo pode evitar que, após meses ou até anos de desenvolvimento, descubra-se, ao implantar o sistema, que o software não atende completamente às necessidades do cliente em razão, sobretudo, de falhas de comunicação durante as entrevistas iniciais.**

- 
- Depois de determinar quais as modificações necessárias ao sistema de informação após o protótipo ter sido apresentado aos usuários, pode-se modificar a interface do protótipo de acordo com as novas especificações e reapresentá-lo ao cliente.
  - **A etapa de análise de requisitos deve produzir um protótipo para demonstrar como se apresentará e comportará o sistema em essência, bem como quais informações deverão ser inseridas no sistema e que tipo de informações deverá ser fornecido pelo software.**
  - **Por meio da ilustração que um protótipo pode apresentar, a maioria das dúvidas e erros de especificação pode ser sanada pelo fato de um protótipo demonstrar visualmente um exemplo de como funcionará o sistema depois de concluído, como será sua interface, de que maneira os usuários interagirão com ele, que tipos de relatórios serão fornecidos etc., facilitando a compreensão do cliente.**
- 

- 
- 
- 
- 
- 
- Apesar das grandes vantagens advindas do uso da técnica de prototipação, é necessária, ainda, uma ressalva: **um protótipo pode induzir o cliente a acreditar que o software encontra-se em um estágio bastante avançado de desenvolvimento.**
  - Por isso, é preciso deixar bem claro ao usuário que o software que lhe está sendo apresentado é apenas uma “maquete” do que será o sistema de informação quando estiver finalizado e que seu desenvolvimento ainda não foi realmente iniciado.

# Prazos e Custos

- **Como determinar o prazo real de entrega de um software? Como determinar a real complexidade de desenvolvimento? Quantos profissionais deverão trabalhar no projeto? Qual será o custo total para concluir o projeto? Qual deverá ser o valor estipulado para produzir o sistema?**
- Geralmente, após as primeiras entrevistas, os clientes estão bastante interessados em saber quanto lhes custará o software e em quanto tempo o terão implantado e funcionando em sua empresa.
- Por mais bem modelado que um sistema tenha sido, ainda, assim, fica difícil determinar com exatidão os prazos finais de entrega do software. **Uma boa modelagem auxilia a estimar a complexidade de desenvolvimento de um sistema, o que, por sua vez, ajuda a determinar o prazo final em que o software será entregue.**

# Prazos e Custos

- No entanto, é preciso ter **diversos sistemas de informação com níveis de dificuldade e características semelhantes aos do software que será construído, previamente desenvolvidos e bem documentados**, para determinar com mais exatidão a estimativa de prazos.
- Para auxiliar a estimativa de prazos e custos de um software, a documentação da empresa desenvolvedora deverá ter registros das datas de início e término de cada projeto já concluído, além do custo real de desenvolvimento que tais projetos acarretaram, o número de profissionais envolvidos em cada um deles, bem como as funções por eles desempenhadas e os salários recebidos, devendo envolver também custos com manutenção.

# Prazos e Custos

- **Uma empresa de desenvolvimento de software que nunca desenvolveu um sistema de informação antes e, portanto, não tem documentação histórica de projetos anteriores, terá dificuldades em apresentar uma estimativa correta de prazos e custos**, principalmente porque a equipe de desenvolvimento não saberá com certeza quanto tempo levará desenvolvendo o sistema, nem estimar corretamente o valor a ser cobrado pelo software.
- **Contudo, mesmo com o auxílio dessa documentação, ainda assim é muito difícil estipular uma data exata.** O máximo que se pode conseguir é apresentar uma data aproximada, com base na experiência documentada de desenvolvimento de outros softwares. Isso se deve ao fato de que cada software apresenta um desafio novo.

# Prazos e Custos

- **Assim, é recomendável acrescentar alguns meses à data de entrega, o que servirá como margem de segurança para possíveis erros de estimativa.**
- **É importante estimar o mais corretamente possível os prazos e custos de desenvolvimento, pois se a estimativa de prazo estiver errada, cada dia a mais de desenvolvimento do projeto acarretará prejuízos para a empresa que desenvolve o sistema.**



# Prazos e Custos

- Prejuízos por exemplo de pagamentos de salários aos profissionais envolvidos no projeto que não haviam sido previstos e desgaste dos equipamentos utilizados.
- Isso sem levar em conta prejuízos mais difíceis de contabilizar, como manter inúmeros profissionais ocupados em projetos que já deveriam estar concluídos, que deixam de trabalhar em novos projetos, além da insatisfação dos clientes por não receberem o produto no prazo estimado e a propaganda negativa daí decorrente.

# Projeto

- Na etapa de projeto é realizada a maior parte da modelagem do software a ser desenvolvido.
- A etapa de projeto toma a modelagem iniciada na fase de análise e a enriquece com profundos acréscimos, melhorias e detalhamentos.
- Enquanto na análise foram identificadas as funcionalidades necessárias ao software e suas restrições, na fase de projeto será estabelecido como essas funcionalidades deverão realizar o que foi solicitado.

# Projeto

- **É nesse momento que será selecionada a linguagem de programação a ser utilizada, o sistema gerenciador de banco de dados a ser empregado, como será a interface final do sistema e até mesmo como o software será distribuído fisicamente na empresa, especificando o hardware necessário para a sua implantação e funcionamento correto.**

# Manutenção

- **Alguns autores afirmam que muitas vezes a manutenção de um software pode representar de 40% a 60% do custo total do projeto.**
- Alguém poderá, então, dizer que a modelagem é necessária para diminuir os custos com a manutenção – se a modelagem estiver correta, o sistema não apresentará erros e, então, não precisará sofrer manutenções.
- Embora um dos objetivos de modelar um software seja realmente diminuir a necessidade de mantê-lo, a modelagem não serve apenas para isso.
- **Na verdade, na maioria dos casos, a manutenção de um software é inevitável.**

# Manutenção

- É bastante provável que um sistema de informação, por mais bem modelado que esteja, precise sofrer manutenções.
- **Nesse caso, a modelagem não serve apenas para diminuir a necessidade de futuras manutenções, mas também para facilitar a compreensão do sistema por quem tiver que o manter.**
- Uma modelagem correta, aliada a uma documentação completa e atualizada de um sistema de informação, torna mais rápido o processo de manutenção e impede que erros sejam cometidos, já que é muito comum que, depois de manter uma rotina, método ou função de um software, outras rotinas ou funções do sistema que antes funcionavam perfeitamente passem a apresentar erros ou simplesmente deixem de funcionar.

# Manutenção

- Tais erros são conhecidos como “efeitos colaterais” da manutenção.
- **Além disso, qualquer manutenção a ser realizada em um software deve ser também modelada e documentada, para não desatualizar a documentação do sistema e prejudicar futuras manutenções, já que muitas vezes uma documentação desatualizada pode ser mais prejudicial à manutenção do sistema do que nenhuma documentação.**

# Documentação Histórica

- Refere-se à documentação histórica dos projetos anteriores já concluídos pela empresa. É por meio dessa documentação histórica que a empresa pode responder a perguntas como:
  - **A empresa está evoluindo?**
  - **O processo de desenvolvimento tornou-se mais rápido?**
  - **As metodologias hoje adotadas são superiores às práticas aplicadas anteriormente?**
  - **As estimativas de prazos e custos estão mais próximas da realidade?**
  - **A qualidade do software produzido está melhorando?**

# Documentação Histórica

- Uma empresa ou setor de desenvolvimento de software necessita de um registro detalhado de cada um de seus sistemas de informação antes desenvolvidos para poder determinar, entre outros, fatores como:
  - **Qual é a média de custo e de tempo gasto na análise de um software?**
  - **Qual é a média de custo e de tempo gasto no projeto?**
  - **Qual é a média de custo e de tempo para codificar e testar um sistema?**
  - **Quantos profissionais são necessários envolver, em média, em cada fase de desenvolvimento do software?**
  - **A média de manutenções que um sistema sofre dentro de um determinado período de tempo.**



# Documentação Histórica

- **Essas informações são computadas nos orçamentos de desenvolvimento de novos softwares e são de grande auxílio no momento de determinar prazos e custos mais próximos da realidade.**
- Além disso, a documentação pode ser muito útil em outra área: a Reusabilidade.
- Uma das formas de agilizar o processo de desenvolvimento é a reutilização de rotinas, funções e algoritmos previamente desenvolvidos em outros sistemas.

# Documentação Histórica

- Nesse caso, a documentação correta do sistema pode auxiliar a sanar questões como:
  - **Onde as rotinas se encontram?**
  - **Para que foram utilizadas?**
  - **Em que projetos estão documentadas?**
  - **São adequadas ao software atualmente em desenvolvimento?**
  - **Qual é o nível necessário de adaptação dessas rotinas para serem utilizadas na construção do sistema atual?**

# 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 completude da modelagem, permitindo que cada diagrama complemente os outros.
- Cada diagrama da UML analisa o sistema, ou parte dele, sob uma determinada óptica.

# Por que tantos Diagramas?

- **A utilização de diversos diagramas permite que falhas sejam descobertas, diminuindo a possibilidade da ocorrência de erros futuros.**
- Os diversos diagramas fornecidos pela UML permitem analisar o sistema em diferentes níveis, podendo focar a organização estrutural do sistema, o comportamento de um processo específico, a definição de um determinado algoritmo ou até mesmo as necessidades físicas para implantar o software.