O OBJETIVO DA MODELAGEM DE SISTEMAS DE SOFTWARE É:

1. Gerenciar a complexidade envolvida em um cenário de negócio no qual temos de desenvolver uma solução de software.
2. Possibilitar a comunicação e o entendimento entre as pessoas envolvidas no processo de desenvolvimento do software.
3. Conseguir estimar tempo e custo envolvido no desenvolvimento do software.
4. Possibilitar a modelagem do comportamento das partes do sistema.

PONTOS MAIS IMPORTANTES SOBRE A MODELAGEM DE SISTEMAS DE SOFTWARE:

1. **Gerenciar a Complexidade**: A modelagem ajuda a lidar com a complexidade dos sistemas, facilitando a compreensão e o desenvolvimento de soluções de software.
2. **Comunicação e Entendimento**: Permite uma comunicação clara e um entendimento uniforme entre todos os envolvidos no processo de desenvolvimento.
3. **Estimativa de Tempo e Custo**: Auxilia na estimativa precisa do tempo e custo necessários para o desenvolvimento do software.
4. **Modelagem de Comportamento**: Facilita a modelagem do comportamento das diferentes partes do sistema, garantindo que todos os aspectos essenciais sejam considerados.

Esses pontos são fundamentais para garantir que o desenvolvimento de sistemas de software seja eficiente e eficaz, atendendo às necessidades do negócio e dos usuários.

A MODULARIZAÇÃO MELHORA O DESENVOLVIMENTO DE SISTEMAS DE VÁRIAS MANEIRAS:

1. **Divisão em Módulos**: Divide o sistema em subsistemas, módulos, componentes, serviços, microsserviços ou classes, facilitando a gestão e o desenvolvimento.
2. **Compilação Separada**: Permite que os módulos sejam compilados separadamente, o que pode acelerar o processo de desenvolvimento e testes.
3. **Conexões entre Módulos**: Embora os módulos sejam independentes, eles mantêm conexões com outros módulos, garantindo a coesão do sistema.
4. **Coesão e Acoplamento**: Promove a criação de um conjunto coeso e fracamente acoplado de módulos, o que melhora a manutenção e a escalabilidade do sistema.

Esses aspectos tornam o desenvolvimento de sistemas mais organizado, eficiente e adaptável a mudanças.

ABSTRAÇÃO:

1. **Definição**: Abstração é o exame seletivo de certos aspectos de um problema, focando nas características essenciais e ignorando detalhes irrelevantes.
2. **Lidar com a Complexidade**: É uma forma fundamental de lidar com a complexidade, permitindo a criação de modelos que representam as partes essenciais de um sistema.
3. **Semelhanças e Diferenças**: Surge do reconhecimento de semelhanças entre objetos, situações ou processos, concentrando-se nessas semelhanças e ignorando as diferenças.
4. **Características Essenciais**: Denota as características essenciais de um objeto que o distinguem de outros, proporcionando clareza e definindo limites conceituais.
5. **Pilar da Orientação a Objetos**: É um dos pilares do paradigma orientado a objetos, organizando o software como uma coleção de objetos distintos com estrutura de dados e comportamento.

A abstração facilita a comunicação e o entendimento entre os envolvidos no desenvolvimento de sistemas, gerenciando a complexidade e permitindo a criação de soluções de software eficientes.

ENCAPSULAMENTO:

1. **Ocultamento de Informações**: O encapsulamento trata de esconder os detalhes internos da implementação de um objeto, expondo apenas os aspectos externos que outros objetos podem acessar, protegendo sua estrutura interna.
2. **Separação entre Interface e Implementação**: Segundo Booch (2007), o encapsulamento separa a interface contratual (o que o objeto oferece) da sua implementação (como ele faz isso), garantindo que a lógica interna fique isolada.
3. **Complementaridade com Abstração**: Encapsulamento e abstração trabalham juntos — enquanto a abstração destaca o comportamento observável de um objeto, o encapsulamento foca na implementação que possibilita esse comportamento.
4. **Barreiras Explícitas**: O encapsulamento cria limites claros entre diferentes abstrações, promovendo uma separação de preocupações (interesses), o que facilita a manutenção e a independência entre partes do sistema.

5. **Proteção da Implementação**: Como Booch (2007) afirma, “encapsulamento esconde detalhes da implementação de um objeto”, permitindo que mudanças internas sejam feitas sem afetar os objetos que interagem com ele.

MODULARIZAÇÃO:

1. **Divisão do Sistema**: A modularização consiste em dividir um sistema em partes menores, como subsistemas, módulos, componentes, serviços, microsserviços ou classes, simplificando sua estrutura geral.
2. **Módulos Independentes**: Esses módulos podem ser compilados separadamente, permitindo que cada parte do sistema seja trabalhada de forma independente durante o desenvolvimento ou manutenção.
3. **Conexões entre Módulos**: Apesar de serem independentes, os módulos mantêm conexões com outros módulos, garantindo que juntos formem um sistema funcional e integrado.
4. **Coesão**: A modularidade busca criar módulos coesos, ou seja, cada módulo tem uma função bem definida e clara, focando em uma responsabilidade específica.
5. **Baixo Acoplamento**: O sistema modular é fracamente acoplado, o que significa que as dependências entre os módulos são minimizadas, facilitando alterações e aumentando a flexibilidade.