Fatec
Araras

DESENVOLVIMENTO WEB II

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Orientação a Objetos

Orientação a Objetos

Fatec Araras

Nos primórdios da tecnologia de objetos, muitas pessoas foram inicialmente apresentadas ao "OO" por meio de linguagens de programação. Eles descobriram o que essas novas linguagens poderiam fazer por eles e tentaram aplicá-las na prática para resolver problemas do mundo real. Com o passar do tempo, as linguagens melhoraram, as técnicas de desenvolvimento evoluíram, surgiram melhores práticas e foram criadas metodologias formais orientadas a objetos.

Complexidade do software

Fatec Araras

Percebemos que alguns sistemas de software não são complexos. Esses são os aplicativos amplamente esquecíveis que são especificados, construídos, mantidos e usados pela mesma pessoa, geralmente o programador amador ou o desenvolvedor profissional trabalhando isoladamente.

Isto não quer dizer que todos esses sistemas sejam grosseiros e deselegantes, nem pretendemos menosprezar os seus criadores. Tais sistemas tendem a ter uma finalidade muito limitada e uma vida útil muito curta.

Complexidade do software

Fatec Araras

Essa complexidade externa geralmente surge da “lacuna de comunicação” que existe entre os usuários de um sistema e seus desenvolvedores: os usuários geralmente acham muito difícil expressar com precisão suas necessidades de uma forma que os desenvolvedores possam entender.

Em alguns casos, os usuários podem ter apenas ideias vagas sobre o que desejam em um sistema de software. Isso não é tanto culpa dos usuários ou dos desenvolvedores de um sistema; em vez disso, ocorre porque cada grupo geralmente carece de experiência no domínio do outro.

Complexidade do software

Usuários e desenvolvedores têm perspectivas diferentes sobre a natureza do problema e fazem suposições diferentes sobre a natureza da solução.

Uma complicação adicional é que os requisitos de um sistema de software mudam frequentemente durante o seu desenvolvimento, em grande parte porque a própria existência de um projeto de desenvolvimento de software altera as regras do problema.

A dificuldade no processo de desenvolvimento

Fatec Araras

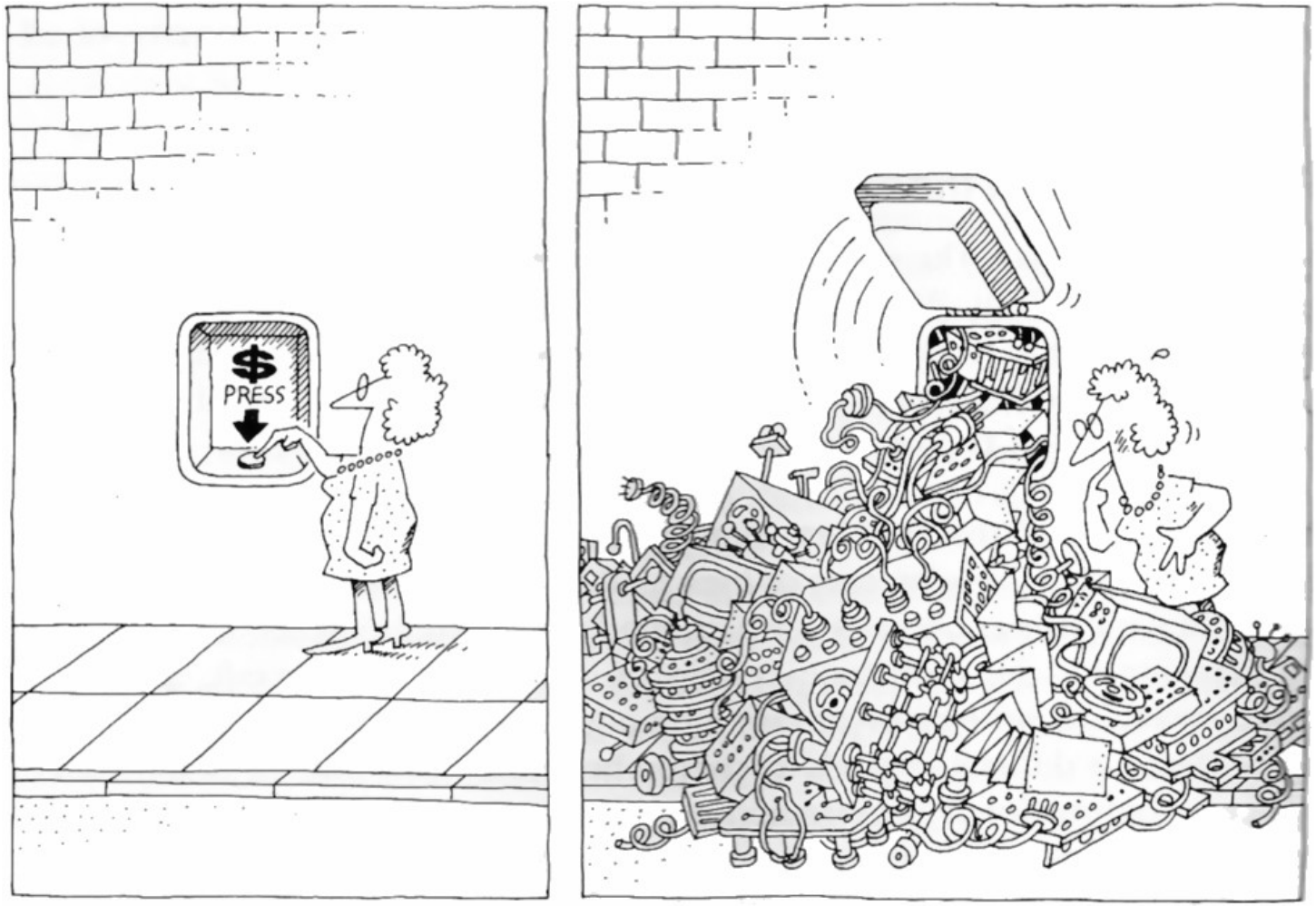
A tarefa fundamental da equipe de desenvolvimento de software é criar a ilusão de simplicidade – proteger os usuários dessa vasta e muitas vezes arbitrária complexidade externa.

Certamente, o tamanho não é uma grande virtude num sistema de software.

Nós nos esforçamos para escrever menos código, inventando mecanismos inteligentes e poderosos que nos dão essa ilusão de simplicidade, bem como reutilizando estruturas de designs e códigos existentes.

A dificuldade no processo de desenvolvimento

Fatec Araras



Prof. Me. Orlando Saraiva Júnior

Decomposição do software

O papel da decomposição

Fatec Araras

“A técnica de dominar a complexidade é conhecida desde os tempos antigos: divide et impera (dividir para governar)”.

Ao projetar um sistema de software complexo, é essencial decompô-lo em partes cada vez menores, cada uma das quais podemos refinar de forma independente.

Desta forma, satisfazemos a restrição muito real que existe na capacidade de canal da cognição humana: para compreender qualquer nível de um sistema, precisamos apenas compreender algumas partes (em vez de todas as partes) de uma só vez.

Decomposição Orientada a Objetos

Fatec Araras

Os objetos fazem coisas e pedimos que façam o que fazem, enviando-lhes mensagens. Como nossa decomposição é baseada em objetos e não em algoritmos, chamamos isso de decomposição orientada a objetos.

Qual é a maneira correta de decompor um sistema complexo – por algoritmos ou por objetos? Na verdade, esta é uma pergunta capciosa porque a resposta certa é que ambas as visões são importantes: a visão algorítmica destaca a ordem dos eventos, e a visão orientada a objetos enfatiza os agentes que causam a ação ou são os sujeitos sobre os quais essas operações atuam.

Surgimento Programação Orientada a Objetos

Fatec Araras

O desenvolvimento orientado a objetos não se gerou espontaneamente a partir das cinzas dos incontáveis projetos de software fracassados que usaram tecnologias anteriores. Não se trata de um afastamento radical das abordagens anteriores. Na verdade, baseia-se nas melhores ideias de tecnologias anteriores.

Linguagens da Primeira e Segunda geração

Fatec Araras

Na figura a seguir, vemos a topologia da maioria das linguagens de programação de primeira e segunda geração.

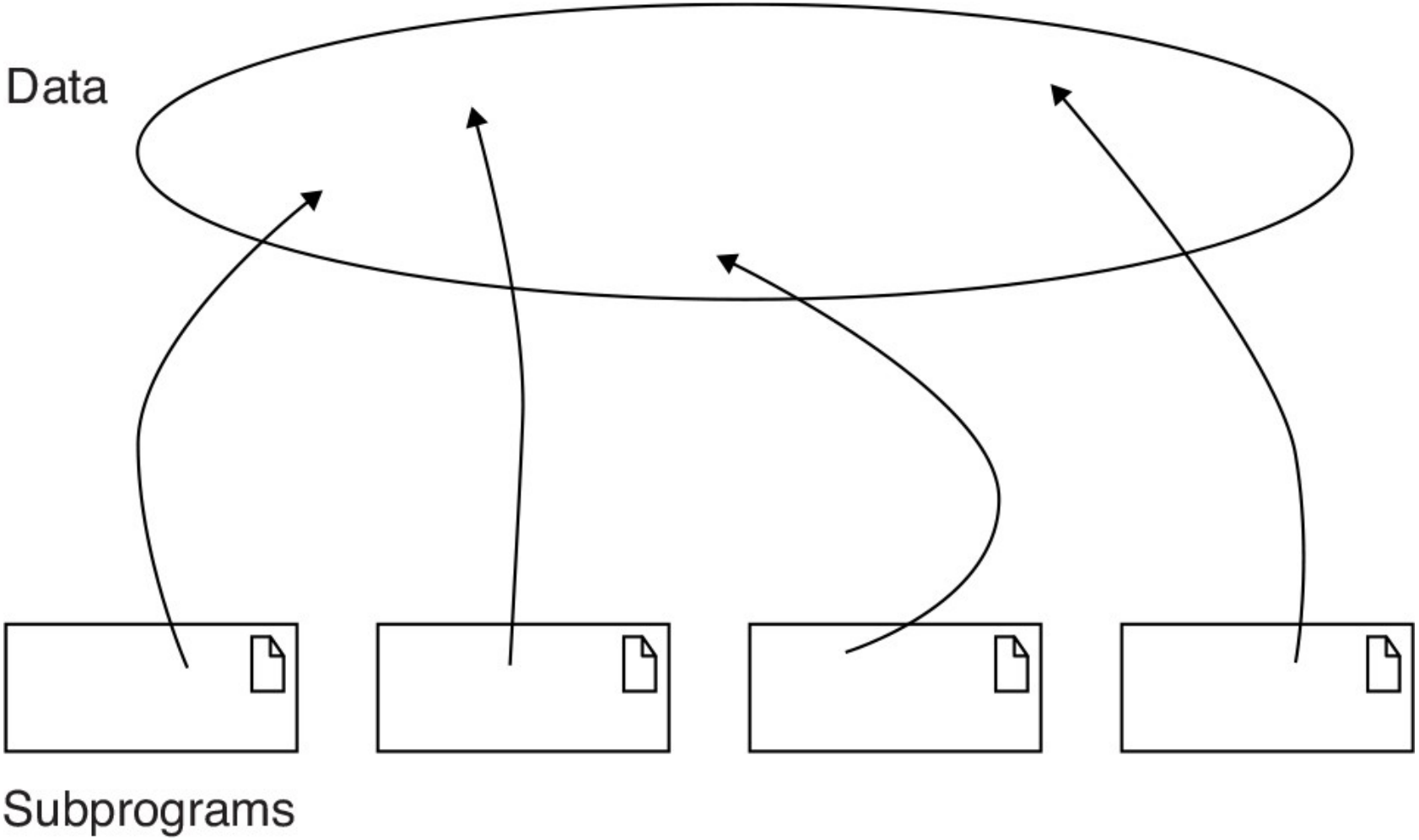
Por topologia, queremos dizer os blocos físicos básicos da linguagem e como essas partes podem ser conectadas.

As aplicações escritas nessas linguagens exibem uma estrutura física relativamente plana, consistindo apenas de dados globais e subprogramas. As setas nesta figura indicam dependências dos subprogramas em vários dados.

Durante o projeto, é possível separar logicamente diferentes tipos de dados uns dos outros, mas há pouco nessas linguagens que possa impor essas decisões de projeto.

A topologia das primeiras linguagens de programação

Fatec Araras



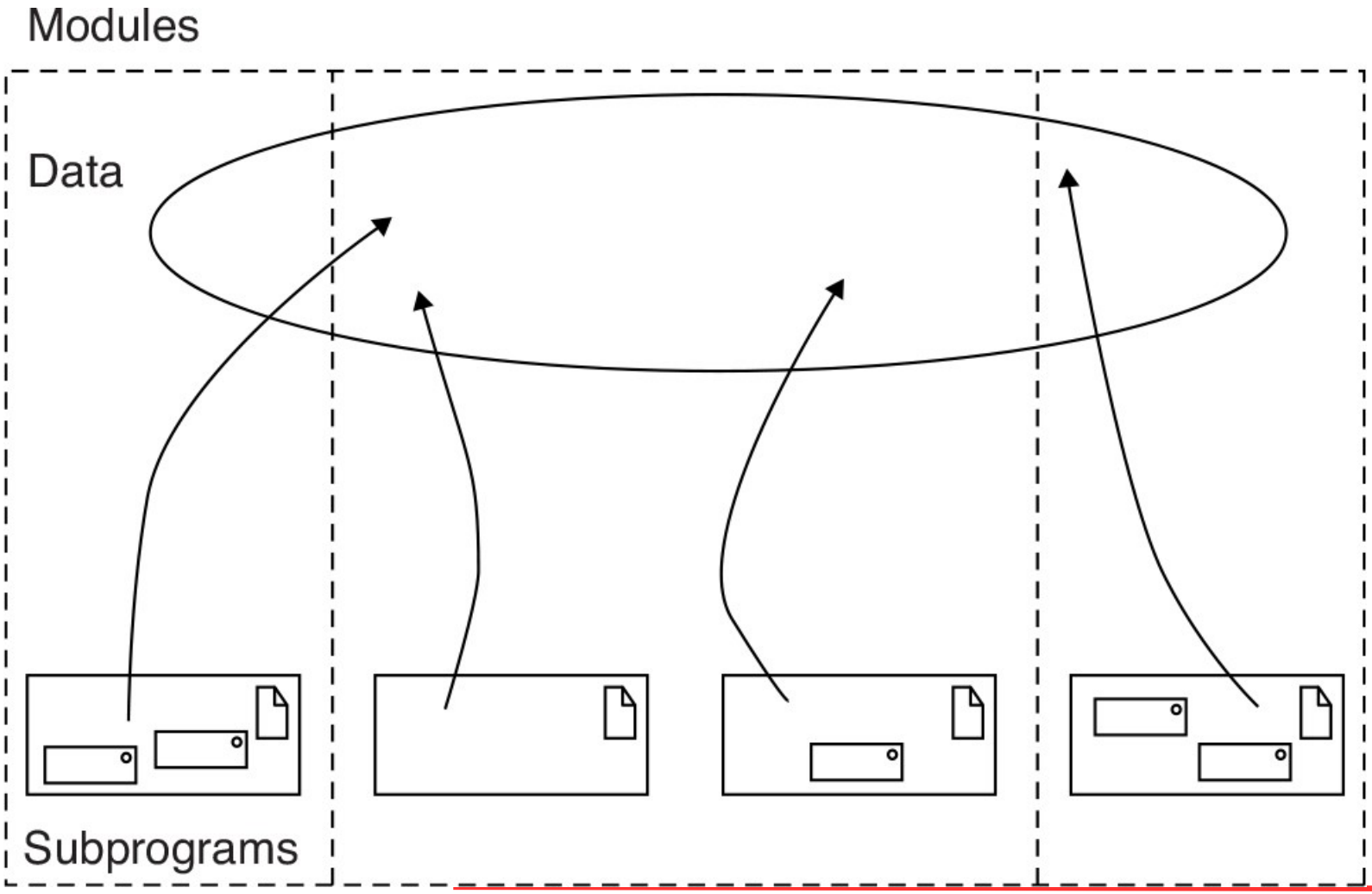
Linguagens da Terceira geração

Fatec Araras

Posteriormente, embora suportassem algum tipo de estrutura modular, tinham poucas regras que exigiam consistência semântica entre as interfaces dos módulos.

Infelizmente, a maioria das linguagens dessa época tinha um péssimo suporte para abstração de dados e tipagem forte.

A topologia das linguagens de terceira geração



A topologia das linguagens baseadas em OO

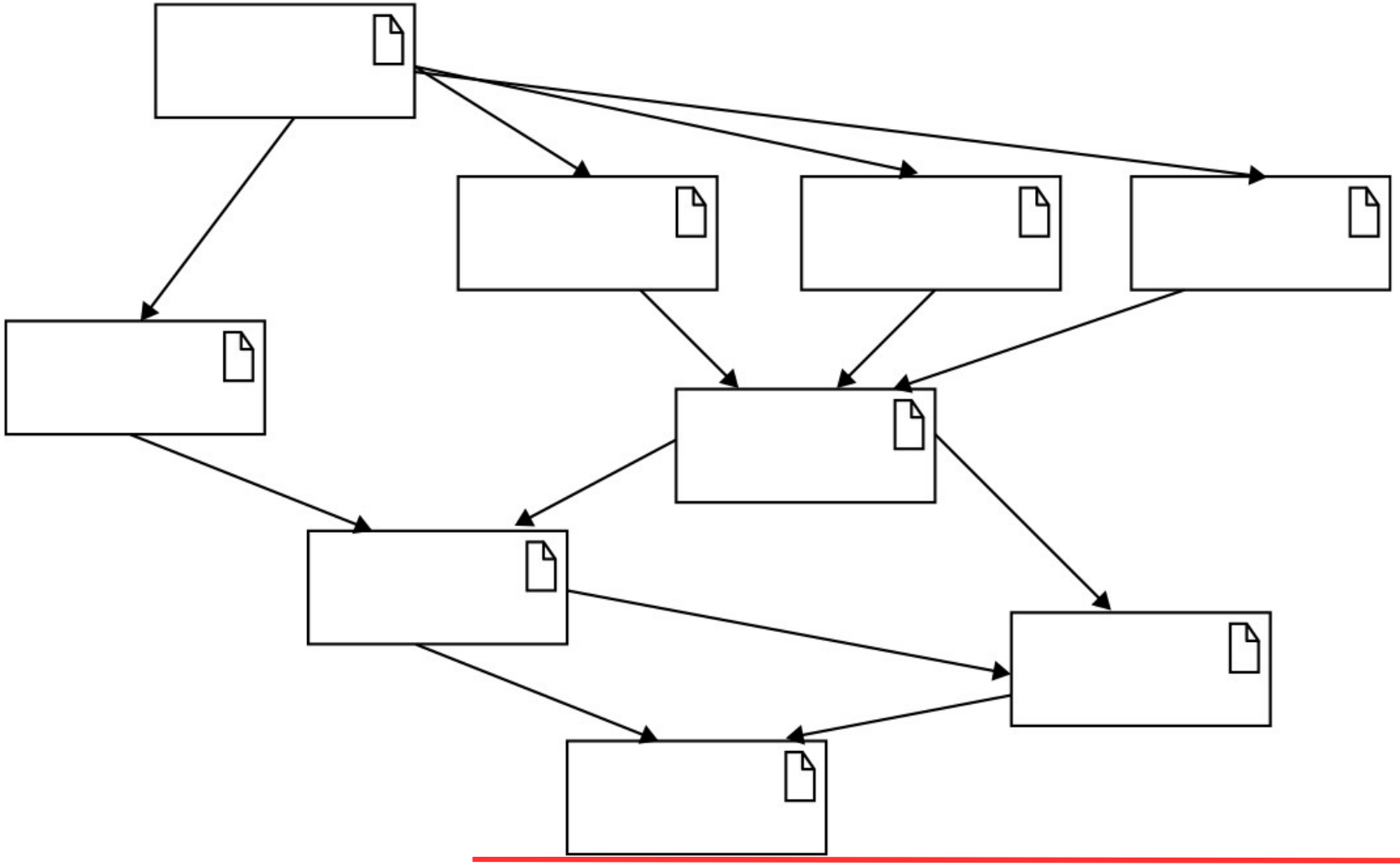
Fatec Araras

O bloco de construção físico nessas linguagens é o módulo, que representa uma coleção lógica de classes e objetos em vez de subprogramas, como nas linguagens anteriores.

Dito de outra forma: "Se procedimentos e funções são verbos e dados são substantivos, um programa orientado a procedimentos é organizado em torno de verbos, enquanto um programa orientado a objetos é organizado em torno de substantivos"

A topologia das linguagens baseadas em OO

Fatec Araras



A topologia das linguagens baseadas em OO

Fatec Araras

Por esse motivo, a estrutura física de uma aplicação orientada a objetos de tamanho pequeno a moderado aparece como um gráfico, e não como uma árvore, o que é típico de linguagens orientadas por algoritmos.

Além disso, há poucos ou nenhum dado global. Em vez disso, os dados e as operações estão unidos de tal forma que os blocos de construção lógicos fundamentais dos nossos sistemas já não são algoritmos, mas sim classes e objetos.

Orientação a Objetos

O que é ?

Fatec Araras

A programação orientada a objetos é um método de implementação no qual os programas são organizados como coleções cooperativas de objetos, cada um dos quais representa uma instância de alguma classe, e cujas classes são todas membros de uma hierarquia de classes unidas por meio de relacionamentos de herança.

O que é ?

Fatec Araras

[Uma] linguagem é orientada a objetos se e somente se satisfizer os seguintes requisitos:

Ele suporta objetos que são abstrações de dados com uma interface de operações nomeadas e um estado local oculto.

Os objetos possuem um tipo associado [classe].

Tipos [classes] podem herdar atributos de supertipos [superclasses].

O que é ?

Fatec Araras

Para uma linguagem apoiar herança significa que é possível expressar relações "é um" entre tipos, por exemplo, uma rosa vermelha é um tipo de flor e uma flor é um tipo de planta.

Se uma linguagem não fornece suporte direto para herança, então ela não é orientada a objetos.

Cardelli e Wegner distinguem essas linguagens chamando-as de baseadas em objetos em vez de orientadas a objetos. Sob esta definição, Smalltalk, Object Pascal, C++, Eiffel, CLOS, C# e Java são todos orientados a objetos, e Ada83 é baseado em objetos.

O que é ?

Fatec Araras

Cada estilo de programação é baseado em sua própria estrutura conceitual.

Cada um requer uma mentalidade diferente, uma maneira diferente de pensar sobre o problema. Para todas as coisas orientadas a objetos, a estrutura conceitual é o modelo de objetos. Existem quatro elementos principais deste modelo:

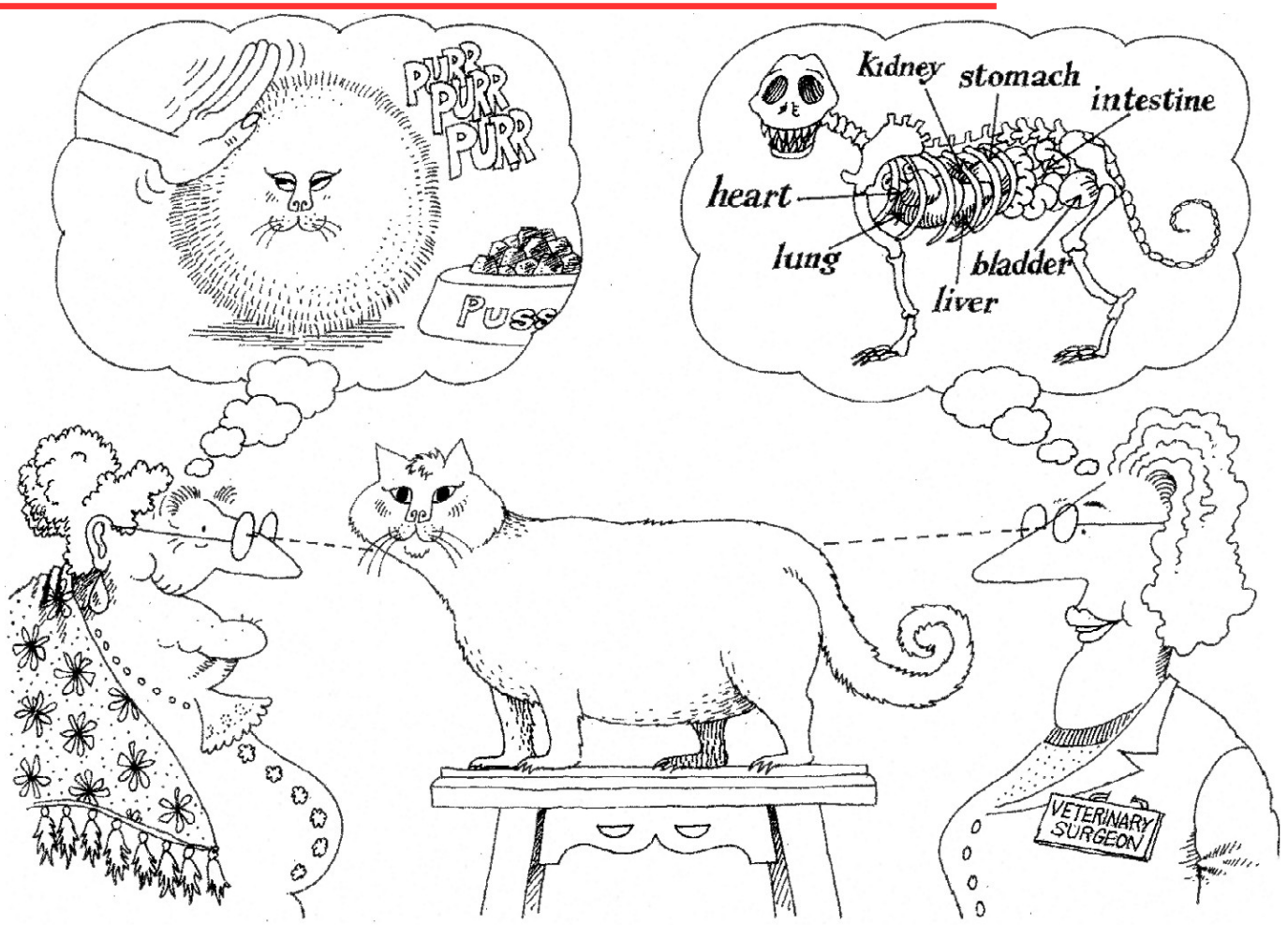
- Abstração
- Encapsulamento
- Modularidade
- Hierarquia

O significado da abstração

A abstração é uma das maneiras fundamentais pelas quais nós, como humanos, lidamos com a complexidade. Dahl, Dijkstra e Hoare sugerem que "a abstração surge do reconhecimento de semelhanças entre certos objetos, situações ou processos no mundo real, e da decisão de se concentrar nessas semelhanças e de ignorar por enquanto as diferenças"

Shaw define uma abstração como "uma descrição simplificada, ou especificação, de um sistema que enfatiza alguns detalhes ou propriedades do sistema enquanto suprime outros. Uma boa abstração é aquela que enfatiza detalhes que são significativos para o leitor ou usuário e suprime detalhes que são , pelo menos no momento, imaterial ou diversivo"

O significado da abstração



A abstração concentra-se nas características essenciais de algum objeto, em relação à perspectiva do observador.

O significado de encapsulamento

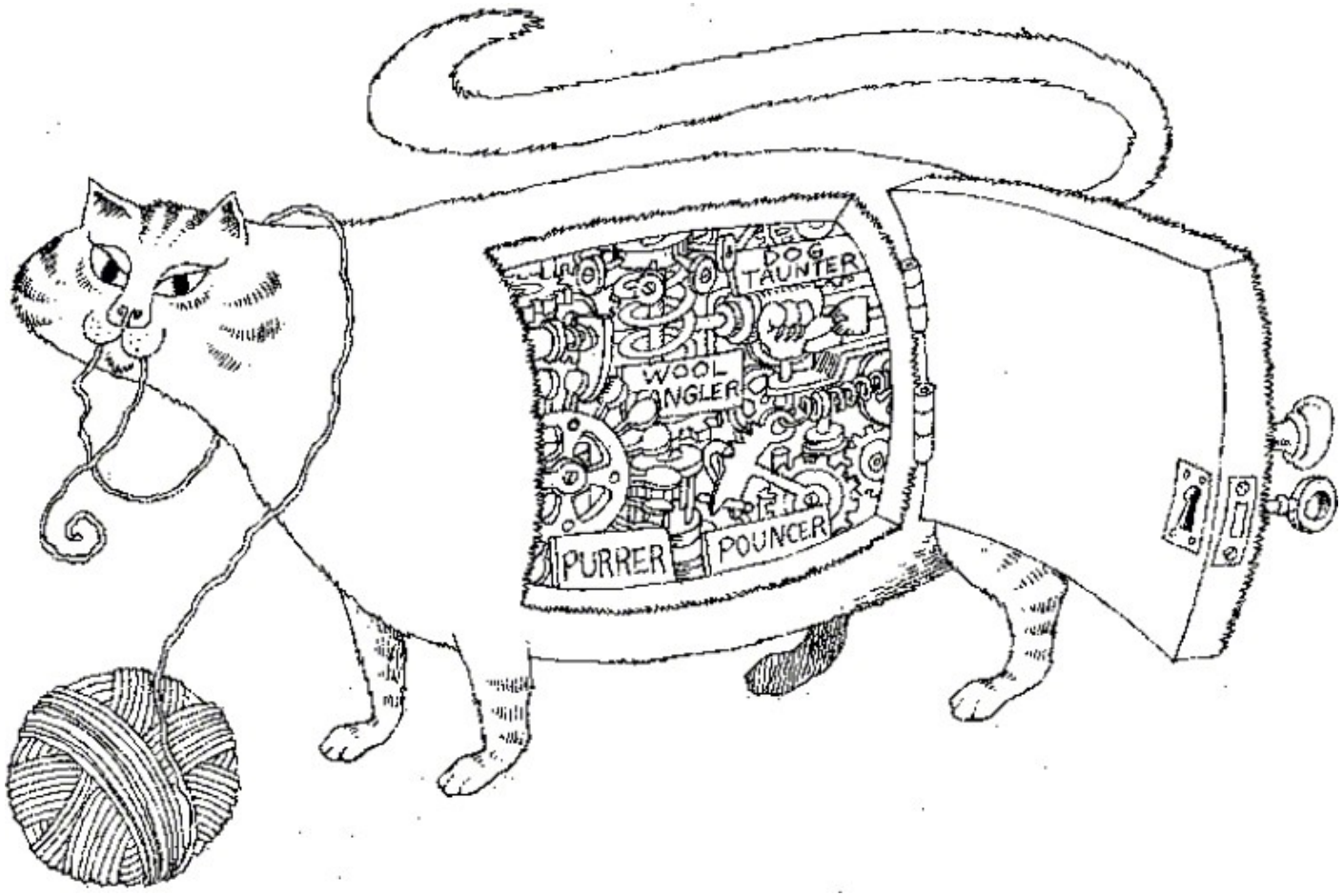
Fatec Araras

Abstração e encapsulamento são conceitos complementares: a abstração concentra-se no comportamento observável de um objeto, enquanto o encapsulamento concentra-se na implementação que dá origem a esse comportamento.

O encapsulamento é mais frequentemente conseguido através da ocultação de informações (não apenas da ocultação de dados), que é o processo de ocultar todos os segredos de um objeto que não contribuem para suas características essenciais; normalmente, a estrutura de um objeto fica oculta, assim como a implementação de seus métodos. “Nenhuma parte de um sistema complexo deve depender dos detalhes internos de qualquer outra parte”.

Enquanto a abstração “ajuda as pessoas a pensar sobre o que estão fazendo”, o encapsulamento “permite que mudanças no programa sejam feitas de forma confiável e com esforço limitado”

O significado de encapsulamento



O encapsulamento oculta os detalhes da implementação de um objeto.

O significado de Modularidade

Fatec Araras

"O ato de particionar um programa em componentes individuais pode reduzir sua complexidade até certo ponto... Embora o particionamento de um programa seja útil por esse motivo, uma justificativa mais poderosa para particionar um programa é que ele cria uma série de componentes bem definidos, limites documentados dentro do programa Esses limites, ou interfaces, são inestimáveis na compreensão do programa" .

Em algumas linguagens, como Smalltalk, não existe o conceito de módulo, portanto a classe forma a única unidade física de decomposição. Java possui pacotes que contêm classes.

Em muitas outras linguagens, incluindo Object Pascal, C++ e Ada, o módulo é uma construção de linguagem separada e, portanto, garante um conjunto separado de decisões de design.

Nessas linguagens, classes e objetos formam a estrutura lógica de um sistema; colocamos essas abstrações em módulos para produzir a arquitetura física do sistema. Principalmente para aplicações maiores, nas quais podemos ter muitas centenas de classes, o uso de módulos é essencial para ajudar a gerenciar a complexidade.

O significado de encapsulamento

A modularidade empacota abstrações em unidades discretas.

O significado de Modularidade

Fatec Araras

“A modularização consiste em dividir um programa em módulos que podem ser compilados separadamente, mas que possuem conexões com outros módulos. Usaremos a definição de Parnas: 'As conexões entre os módulos são as suposições que os módulos fazem uns sobre os outros'”.

A maioria das linguagens que suportam o módulo como um conceito separado também distinguem entre a interface de um módulo e sua implementação.

Assim, seria justo dizer que modularidade e encapsulamento andam de mãos dadas.

O significado de Hierarquia

Fatec Araras

A abstração é uma coisa boa, mas em todas as aplicações, exceto nas mais triviais, podemos encontrar muito mais abstrações diferentes do que podemos compreender ao mesmo tempo.

O encapsulamento ajuda a gerenciar essa complexidade, ocultando a visão interna de nossas abstrações.

A modularidade também ajuda, nos dando uma maneira de agrupar abstrações relacionadas logicamente.

Ainda assim, isso não é suficiente. Um conjunto de abstrações geralmente forma uma hierarquia e, ao identificar essas hierarquias em nosso projeto, simplificamos bastante nossa compreensão do problema.

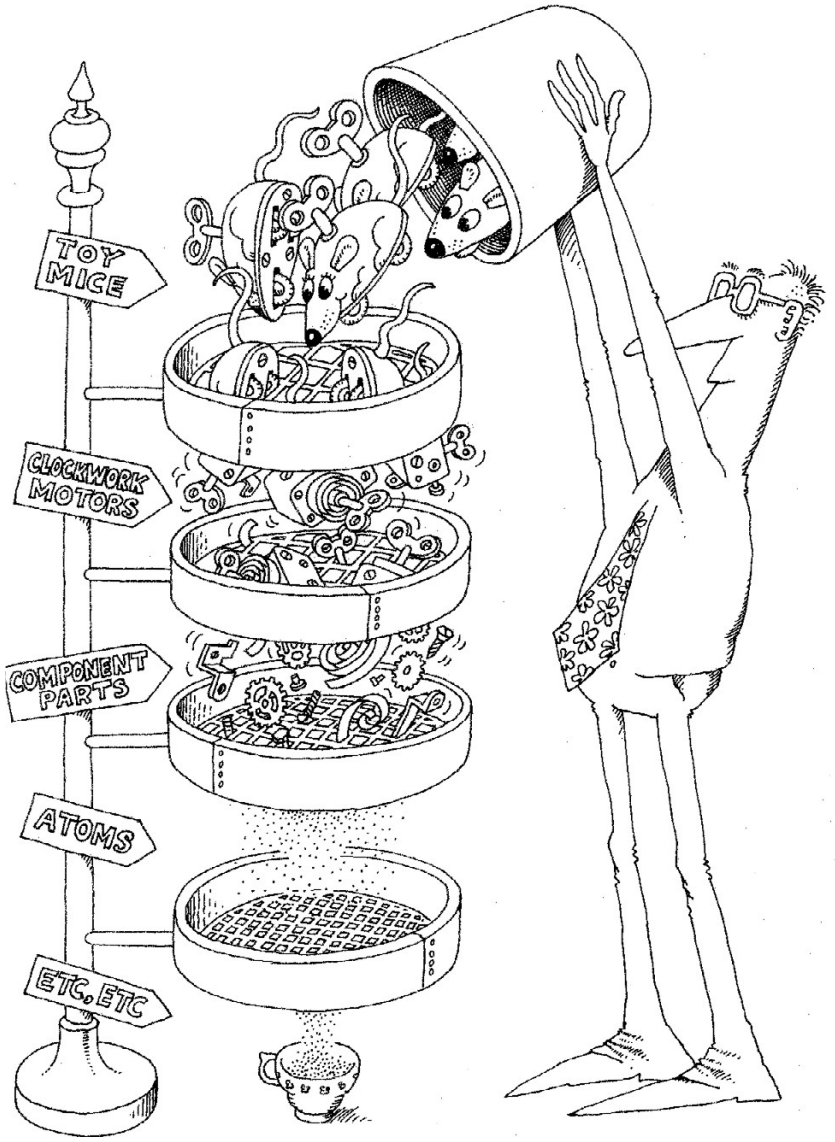
O significado de Hierarquia

Definimos hierarquia da seguinte forma:

Hierarquia é uma classificação ou ordenação de abstrações.

As duas hierarquias mais importantes em um sistema complexo são sua estrutura de classes (a hierarquia "é uma") e sua estrutura de objetos (a hierarquia "parte de").

O significado de hierarquia



Abstrações formam uma hierarquia.

Prof. Me. Orlando Saraiva Júnior

Vantagens do Modelo de Objeto

O modelo de objeto

Fatec Araras

o modelo de objeto é fundamentalmente diferente dos modelos adotados pelos métodos mais tradicionais de análise estruturada, projeto estruturado e programação estruturada.

Isso não significa que o modelo de objeto abandone todos os princípios e experiências sólidos desses métodos mais antigos. Em vez disso, introduz vários elementos novos que se baseiam nestes modelos anteriores.

Assim, o modelo de objetos oferece uma série de benefícios significativos que outros modelos simplesmente não oferecem.

O modelo de objeto

Fatec Araras

O uso do modelo de objetos nos ajuda a explorar o poder expressivo das linguagens de programação baseadas e orientadas a objetos.

Como Stroustrup aponta: "Nem sempre é claro qual a melhor forma de aproveitar as vantagens de uma linguagem como C++. Melhorias significativas na produtividade e na qualidade do código foram consistentemente alcançadas usando C++ como 'um C melhor' com um pouco de abstração de dados incluída onde é claramente útil No entanto, melhorias adicionais e visivelmente maiores foram alcançadas aproveitando-se das hierarquias de classes no processo de design. Isso geralmente é chamado de design orientado a objetos e é aqui que foram encontrados os maiores benefícios do uso de C++. 82].

Sem a aplicação dos elementos do modelo de objeto, os recursos mais poderosos de linguagens como Smalltalk, C++, Java e assim por diante são ignorados ou muito mal utilizados.

O modelo de objeto

O uso do modelo de objeto incentiva a reutilização não apenas de software, mas de projetos inteiros, levando à criação de estruturas de aplicativos reutilizáveis (com uso de frameworks).

Descobrimos que os sistemas orientados a objetos são frequentemente menores do que implementações equivalentes não orientadas a objetos.

O modelo de objeto

O uso do modelo de objeto produz sistemas construídos em formas intermediárias estáveis, mais resilientes à mudança. Isto também significa que tais sistemas podem evoluir ao longo do tempo, em vez de serem abandonados ou completamente redesenhados em resposta à primeira grande mudança nos requisitos.

Fatec
Araras

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Bibliografia

Fatec Araras