VOLTAR

Diferença entre as abordagens análise estruturada, orientada a objetos e orientada a aspectos

Apresentar as principais características das abordagens estruturada, orientada a objetos e orientada a aspectos, de maneira a ressaltar suas diferenças.

NESTE TÓPICO

- Abordagem estruturada
- Abordagem orientada a objetos
- Abordagem orientada a aspectos
- Referências

Abordagem estruturada

Independente do paradigma utilizado para o desenvolvimento de software, as fases de análise e projeto sempre estão presentes.

Os principais objetivos dessas fases são:

- Redução da complexidade.
- Descoberta da estrutura do problema.
- Elaboração de esboço para a solução.

Podemos utilizar abordagens nessas fases que utilizam técnicas de detalhamento top-down (do todo para as partes) e bottom-up (inverso).

- Abordagem top-down: processo de etapas que se inicia com o objeto mais geral (sistema, programa, módulo, estrutura de dados) e, então, divide esse objeto em partes menores e reaplica sucessivamente a divisão nos objetos menores.
- Abordagem bottom-up: começa com componentes de nível mais baixo do sistema (ex.: formato de uma tabela). Esses componentes são agrupados para se ter uma visão um pouco menos detalhada do sistema. O processo se

repete até que se chegue ao nível mais alto de abstração. (Mais adequada para quando já existe um protótipo do sistema e se conhece bem a funcionalidade de cada módulo).

Na etapa de análise é realizada a definição dos requisitos para a solução de um problema. Principais atividades:

- Exame das necessidades dos usuários.
- Definição das propriedades que o sistema deve possuir para atender às necessidades do usuário.
- Definição das funções que o sistema deve desempenhar.
- Estabelecimento das restrições e desempenho desejado para o sistema.

Já na etapa de projeto é realizado o processo de planejamento da elaboração do sistema, que consiste das seguintes atividades:

- Determinação dos módulos do sistema.
- Dados que serão necessários pelos módulos.
- Maneira que módulos serão reunidos de forma a formar o sistema que está sendo desenvolvido.
- Definição de algoritmos para descrever o processamento dos dados por cada módulo.

O resultado da análise é utilizado como "entrada" para a fase de projeto.

O resultado da etapa de análise é a **especificação do sistema**,a qual descreve os componentes do sistema e suas interfaces.

Quando utilizamos a abordagem estruturada, a etapa de análise produz:

- A especificação funcional do sistema (representada pelo Diagrama de Fluxo de Dados – DFD).
- A especificação dos dados (representada pelo Diagrama de Entidade Relacionamento – DER).
- A especificação do comportamento do sistema (representada pelo Diagrama de Transição de Estados – DTE).

A especificação do sistema deve ser precisa a fim de possibilitar verificação e ter alto grau de formalidade.

No caso da etapa de projeto, utilizando a abordagem estruturada, o principal produto gerado é a **especificação do projeto** em que devem estar representadas:

- As informações sobre estruturas de dados.
- A estrutura de programas.
- As características das interfaces.
- Os detalhes dos procedimentos dos módulos.

Normalmente, essa fase é realizada entre as de análise e implementação visando à estruturação do sistema. A principal ferramenta de representação utilizada nessa fase é o Diagrama de Estrutura de Software – DES.

Abordagem orientada a objetos

Essa abordagem permite que o software seja construído de objetos que tenham um comportamento especifico. Os próprios objetos podem ser construídos a partir de outros.

A fase de análise é feita analisando-se os objetos e os eventos que interagem com esses objetos. Já a fase de projeto é feita reusando-se classes de objetos existentes e, quando necessário, construindo-se novas classes.

As técnicas orientadas a objeto podem ser usadas para simplificar o projeto de sistemas complexos. Eles podem ser visualizados como uma coleção de objetos, estando cada um dos objetos em um determinado estado.

Dentre os conceitos básicos utilizados na abordagem orientada a objetos, podemos citar os objetos, classes, métodos, herança e encapsulamento.

Um objeto é qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e os métodos que os manipulam. Um objeto é uma instância de uma classe.

Os métodos especificam a maneira pela qual os dados de um objeto são manipulados. São similares às funções e procedures do universo da programação estruturada.

Já o encapsulamento protege os dados do objeto do uso arbitrário e não intencional. Ele é o resultado (ou ato) de ocultar do usuário os detalhes da implementação de um objeto. Além disso, ele se torna importante porque separa a maneira como um objeto se comporta da maneira como ele é implementado.

Uma classe especifica uma estrutura de dados e os métodos operacionais permissíveis, que se aplicam a cada um de seus objetos.

É comum haver similaridades entre diferentes classes. Frequentemente, duas ou mais classes irão compartilhar os mesmos atributos e/ou métodos.

Para evitar ter que reescrever várias vezes o mesmo código, é interessante usar algum mecanismo que possa tirar proveito dessas similaridades. A herança é esse mecanismo. Por intermédio dela é possível reutilizar rotinas e dados já existentes.

Abordagem orientada a aspectos

A necessidade de software de qualidade motivou o uso da programação orientada a objetos em busca de maiores níveis de reuso e manutenção, aumentando a produtividade e o suporte a mudanças de requisitos. Porém, o paradigma orientado a objetos apresenta limitações, tais como o entrelaçamento e o espalhamento de código por meio de vários requisitos.

Por exemplo, código de distribuição entrelaçado com código de negócio e de interface com o usuário e código de controle de concorrência espalhado por diversos módulos do software. Sabemos que o software é composto pelos seus requisitos funcionais e não funcionais.

A programação orientada a objetos mistura os requisitos funcionais e os requisitos não funcionais do sistema, sendo que esse entrelaçamento dificulta a reutilização do código.

A programação orientada a aspecto busca separar os requisitos funcionais e não funcionais, ou seja, dividir os interesses (concerns), para depois uni-los (weaving), formando um sistema completo.

Cada vez mais se nota a vantagem e a necessidade do uso do desenvolvimento e da programação orientada a aspectos, que supera as deficiências da orientação a objetos.

O desenvolvimento orientado a aspectos (ASOD) objetiva desenvolver um sistema pela utilização de um novo conceito, o aspecto, visando obter alta modularidade, separando código que implementa funções específicas que afetam diferentes partes do sistema, chamadas "interesses ortogonais" (crosscutting concern). O aspecto é a estrutura que os encapsulam.

A princípio, o conceito de aspecto foi introduzido no nível de implementação por meio da programação orientada a aspecto (AOP).

AOSD agora não é só AOP, porque ele engloba várias técnicas para obter melhor modularidade. Portanto, AOSD focaliza sobre a modularidade dos interesses (crosscutting concerns) em todas as etapas do ciclo de vida de desenvolvimento do software.

O AOSD inicia seu processo com a fase denominada engenharia de requisitos orientada a aspectos, que se concentra na separação das propriedades funcionais e não funcionais dos crosscutting, que, do contrário, poderiam ficar espalhadas e entrelaçadas com outros requisitos.

A seguir, é realizada a fase de arquitetura orientada a aspectos a qual se concentra na localização e especificação dos crosscutting concerns no projeto arquitetural. Essa fase utiliza modelagem e projeto orientados a aspecto para descrever a arquitetura do sistema.

O projeto orientado a aspecto se baseia no fato que o espalhamento e entrelaçamento encontrados em outras abordagens podem ser modularizados. Nessa abordagem, os requisitos são utilizados como entrada para produzir um modelo de projeto, o qual representa os interesses (concerns) separadamente e seus relacionamentos. Aqui são utilizados "constructs" que podem descrever os elementos representados no projeto e seus relacionamentos.

Para a fase de implementação é utilizada a AOP que inclui técnicas e ferramentas para a modularização dos crosscuting concerns no nível de código. Nesse processo, elementos separados e independentes são implementados.

Finalmente, são utilizadas técnicas de testes orientadas a aspectos para testar o sistema. O objetivo dos testes é mostrar e rever se os aspectos implementados realmente incorporaram características de crosscutting concerns ao sistema para satisfazer os requisitos do cliente.

Agora que você já estudou esta aula, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

EXERCÍCIOS (https://ead.uninove.br/ead/disciplinas/impressos/_g/pdsoft80_100/a06e

Referências

PRESSMAN, R. S. *Engenharia de software.* 7ª ed. São Paulo: McGraw-Hill, 2010.

SOMMERVILLE, Ian. *Engenharia de software.* São Paulo: Addison-Wesley, 2007.

Avalie este tópico

ANTERIOR
Modelos de desenvolvimento: espiral e incremental

Índice

PRÓXIMO
Engenharia orientada a serviços

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.un idCurso=)