

seminários abertos em computação

Identificação de Interesses Transversais: Uma Visão Geral

Paulo Afonso Parreira Júnior

paulojunior@jatai.ufg.br

Agenda

- Breve Histórico sobre Desenvolvimento de Software
- Separação de Interesses
- Orientação a Aspectos
- Identificação de Interesses Transversais
- Pesquisas: Mestrado e Doutorado
- Ideias para Projetos
- Comentários & Dúvidas

Agenda

- Breve Histórico sobre Desenvolvimento de Software
- Separação de Interesses
- Orientação a Aspectos
- Identificação de Interesses Transversais
- Pesquisas: Mestrado e Doutorado
- Ideias para Projetos
- Comentários & Dúvidas

Breve Histórico sobre Desenvolvimento de Software

- O que é linguagem?
 - Conjunto de palavras que podem ser geradas a partir de um alfabeto.
 - Alfabeto da língua portuguesa: A, B, C, D ...
 - Algumas palavras: laranja, porta, cadeira, universidade...

Breve Histórico sobre Desenvolvimento de Software

- Para que serve uma linguagem?



Resposta: para se comunicar.

Breve Histórico sobre Desenvolvimento de Software

- Linguagem de Programação:
 - conjunto de palavras derivadas de um alfabeto e utilizadas para expressar nossas ideias e intenções de forma clara e objetiva ao computador.

Breve Histórico sobre Desenvolvimento de Software

- Alguns tipos de linguagens:
 - Imperativas: o programador conhece a solução para o problema que ele deve resolver e “diz” ao computador como resolvê-lo.
 - Declarativas: o programador simplesmente descreve o problema que ele deve resolver e o computador o resolve.

Breve Histórico sobre Desenvolvimento de Software

- Instruções comuns em linguagens imperativas:
 - comandos de entrada e saída de dados, estruturas condicionais e de repetição, procedimentos, funções...

```
scanf, printf, if..else,  
while,do..while, switch,...
```

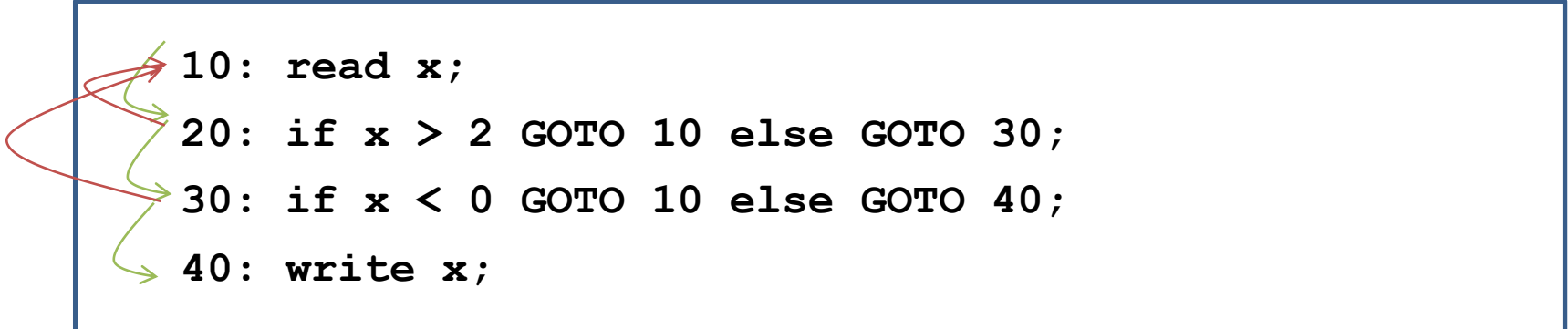

Breve Histórico sobre Desenvolvimento de Software

- Programação Estruturada vs. Programação Estruturada



Breve Histórico sobre Desenvolvimento de Software

- Programação desestruturada:
 - uso indiscriminado do comando GOTO (desvio incondicional);
 - dificulta o entendimento e a manutenção do código do programa.



```
10: read x;  
20: if x > 2 GOTO 10 else GOTO 30;  
30: if x < 0 GOTO 10 else GOTO 40;  
40: write x;
```

Breve Histórico sobre Desenvolvimento de Software

- Programação estruturada:
 - evita-se o uso do comando GOTO;
 - o desvio de fluxo passa a ser realizado apenas por meio das estruturas condicionais.

```
if ((x >= 0) and (x <= 2)) write x;
```

Breve Histórico sobre Desenvolvimento de Software

- Programação estruturada:
 - o uso de GOTO pode continuar acontecendo, “por baixo dos panos” → código compilado;
 - porém, o programador não precisará entender, nem manter o código compilado.

```
if ((x >= 0) and (x <= 2)) write x;
```

Compilação

```
10: read x;  
20: if x > 2 GOTO 10 else GOTO 30;  
30: if x < 0 GOTO 10 else GOTO 40;  
40: write x;
```

Breve Histórico sobre Desenvolvimento de Software

- A **Programação Estruturada** trouxe avanços consideráveis para organização do código fonte de um programa.
- Mas ainda havia um problema central: o foco estava nas funções e não nos dados.
- Os dados não eram protegidos contra acessos indevidos, por isso, havia problemas de garantia de **integridade** dos mesmos.

Breve Histórico sobre Desenvolvimento de Software

- Como saber quando o valor de uma variável dentro de um programa abaixo sofreu alguma alteração?

```
int idade;  
char nome[80];  
  
int main() {  
    idade = 26;  
}
```

- Agravante: qualquer módulo do sistema poderia ter acesso direto às variáveis **idade** e **nome**.

Breve Histórico sobre Desenvolvimento de Software

- Programação Orientada a Objetos (POO):
 - novo paradigma de desenvolvimento de software;
 - o mundo real é composto de objetos, portanto, a POO trouxe uma perspectiva mais humana de observação da realidade, para o contexto da programação;
 - um dos princípios básicos da POO: **separação de interesses.**

Agenda

- Breve Histórico sobre Desenvolvimento de Software
- **Separação de Interesses**
- Orientação a Aspectos
- Identificação de Interesses Transversais
- Pesquisas: Mestrado e Doutorado
- Ideias para Projetos
- Comentários & Dúvidas

Separação de Interesses

- Separação de Interesses (*Separation of Concerns* - SoC, Dijkstra, 1976).



*“(...) **the separation of concerns** (...) is what I mean by "focusing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant.”*

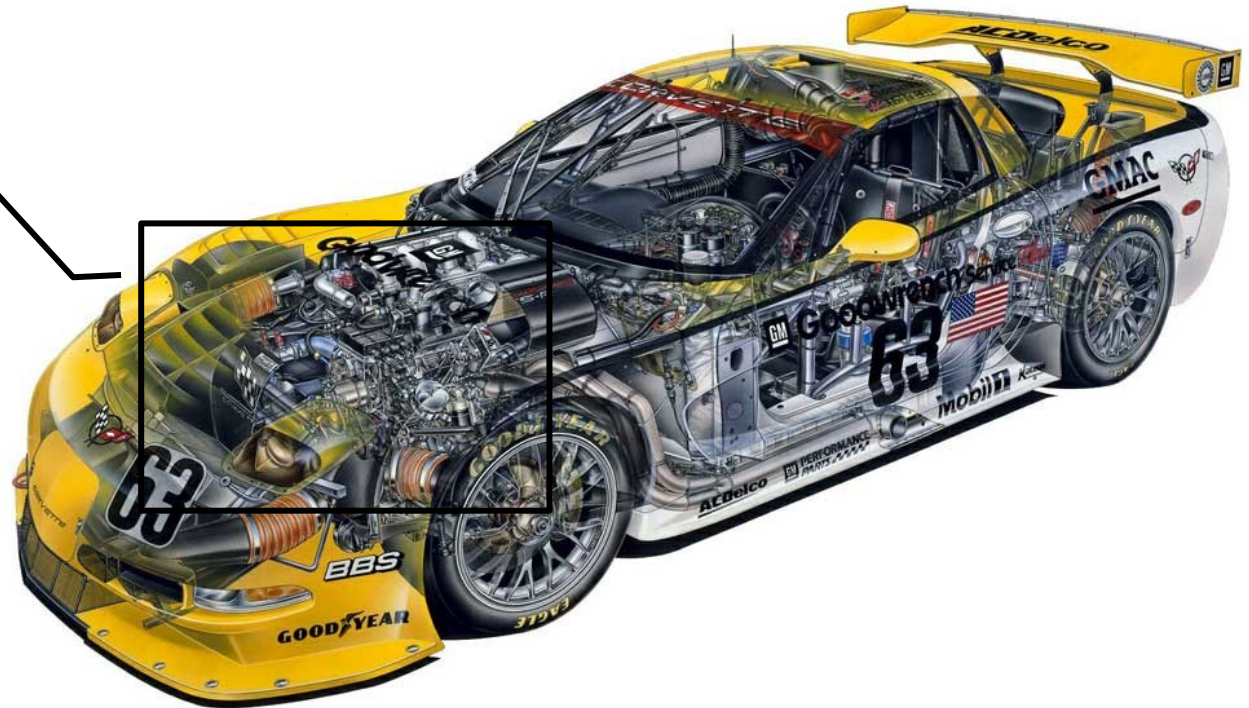
Separação de Interesses

- O que são interesses?
 - são as diferentes preocupações que se tem ao desenvolver um software.
 - Exemplos: requisitos funcionais, persistência, distribuição, segurança, desempenho, entre outros.

Separação de Interesses

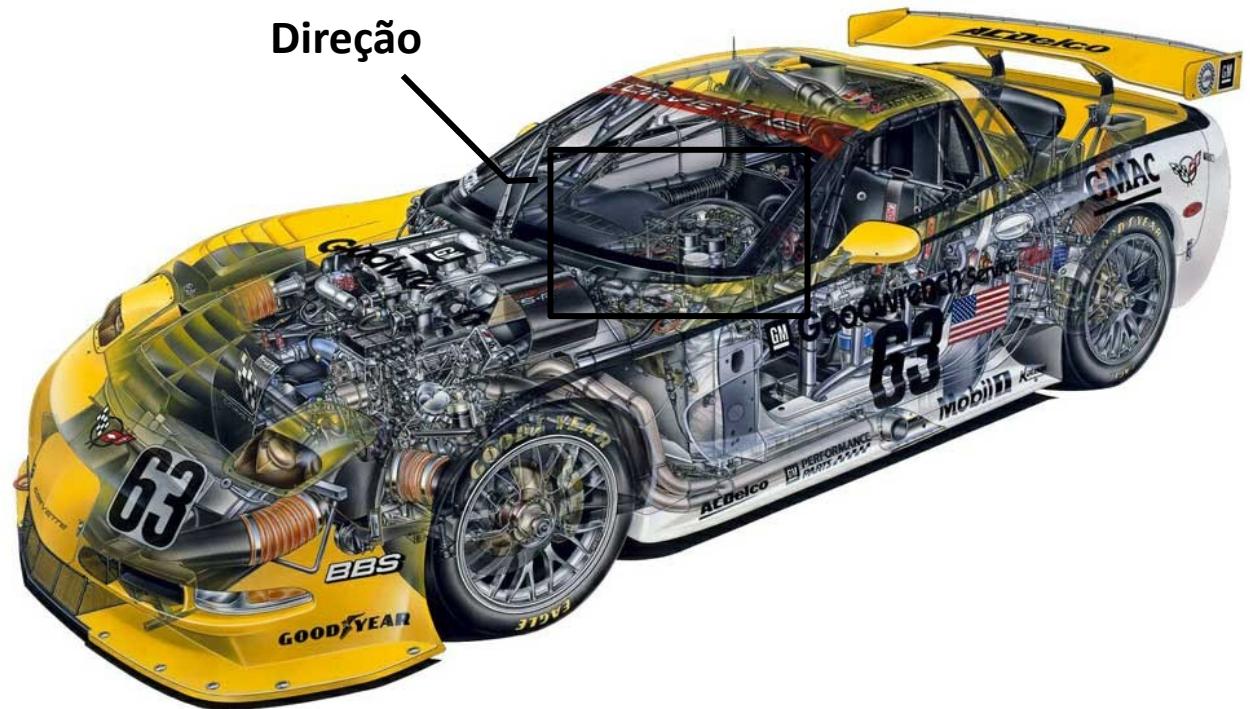
- Exemplo automóvel...

Motor



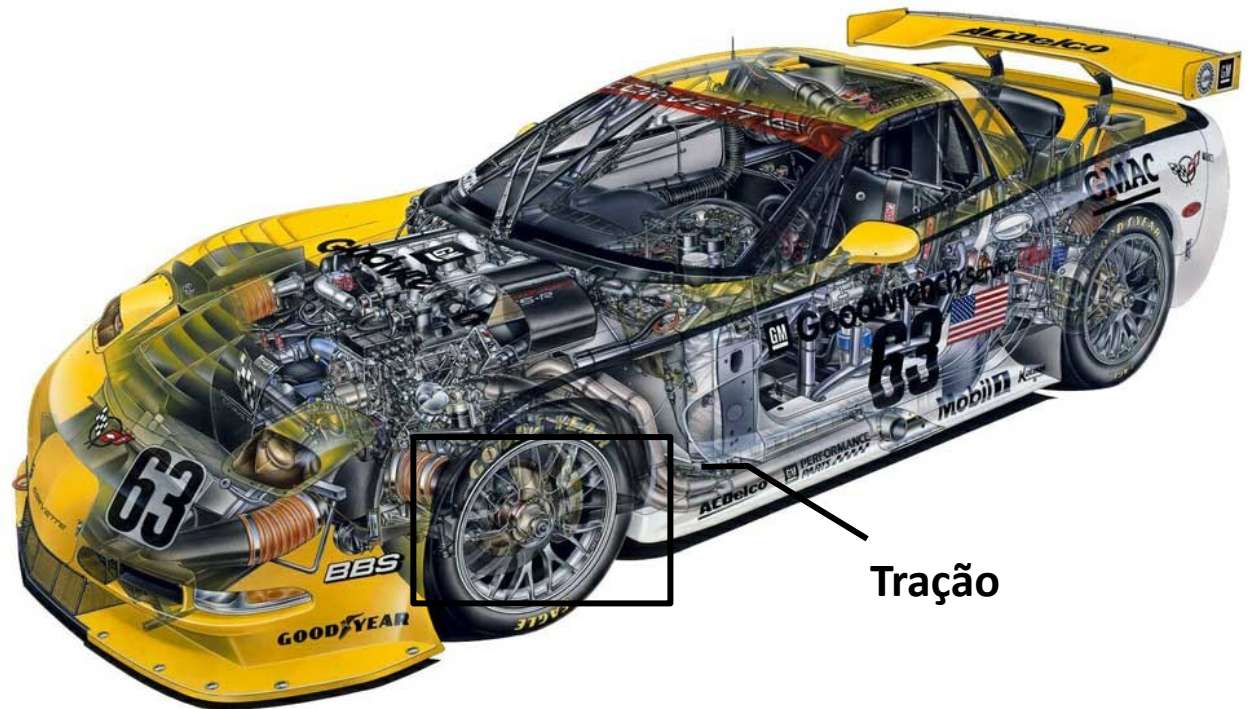
Separação de Interesses

- Exemplo automóvel...



Separação de Interesses

- Exemplo automóvel...

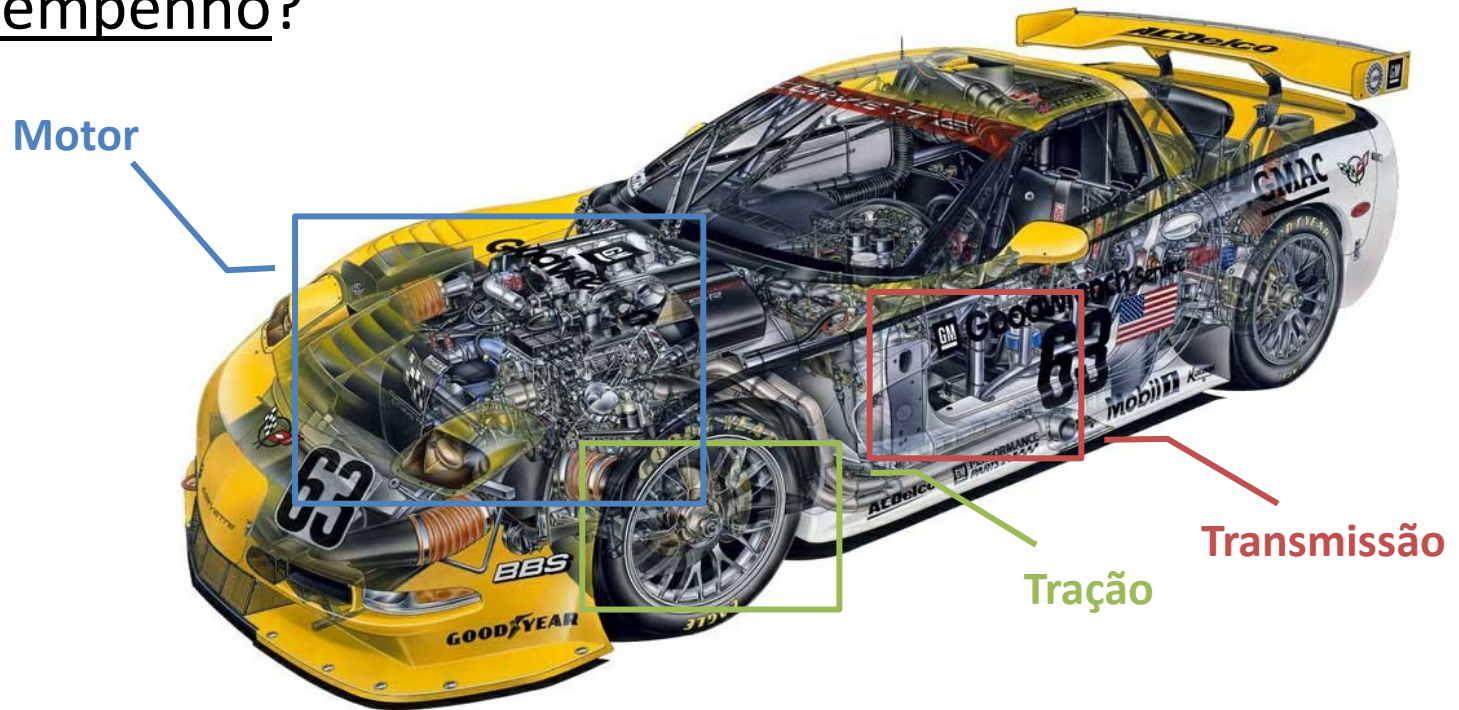


Separação de Interesses

- Como a POO visa a contemplar esse princípio?
 - Por meio dos mecanismos de abstração disponíveis na linguagem: encapsulamento e visibilidade, classes, objetos, entre outros.
 - Entretanto, problemas começam a surgir quando tentamos separar alguns tipos específicos de interesses.

Separação de Interesses

- Qual o componente responsável pelo desempenho?



Separação de Interesses

```
public class Conta {  
    private double saldo;  
    private int numero;  
  
    public void sacar(double valor) {  
        saldo = saldo - valor;  
    }  
}
```

Legenda	
Cor	Interesses *
	Lógica de Negócios

* Interesses a serem implementados no software.

Separação de Interesses

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;

    public void sacar(double valor) {
        saldo = saldo - valor;
        log.exitLog("Saque efetuado na conta: " + numero);
    }
}
```

Legenda	
Cor	Interesses *
■	Lógica de Negócios
■	<i>Logging</i>

* Interesses a serem implementados no software.

Separação de Interesses

```
public class Conta {  
    private double saldo;  
    private int numero;  
    private Logger log;  
    private BD bd;  
  
    public void sacar(double valor) {  
        saldo = saldo - valor;  
        log.exitLog("Saque efetuado na conta: " + numero);  
        bd.salvar(this);  
    }  
}
```

Legenda	
Cor	Interesses *
■	Lógica de Negócios
■	<i>Logging</i>
■	Persistência

* Interesses a serem implementados no software.

Separação de Interesses

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;
    private BD bd;

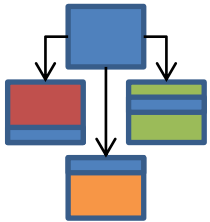
    public void sacar(double valor) throws SaldoInsuficienteException {
        if (saldo >= valor) {
            saldo = saldo - valor;
            log.exitLog("Saque efetuado na conta: " + numero);
            bd.salvar(this);
        } else {
            throw new
                SaldoInsuficienteException(this);
        }
    }
}
```

Legenda	
Cor	Interesses *
■	Lógica de Negócios
■	Logging
■	Persistência
■	Tratamento de Exceções

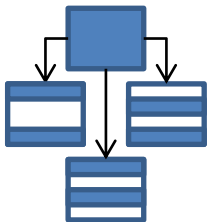
* Interesses a serem implementados no software.

Separação de Interesses

- O que acontece no exemplo anterior é amplamente conhecido como: entrelaçamento e espalhamento de interesses.



- **Entrelaçamento de interesses** é o resultado da inserção de código de um determinado interesse em módulos relacionados a outros tipos de interesses.



- **Espalhamento de interesses** é o resultado da inserção de código de um determinado interesse em vários pontos do software.

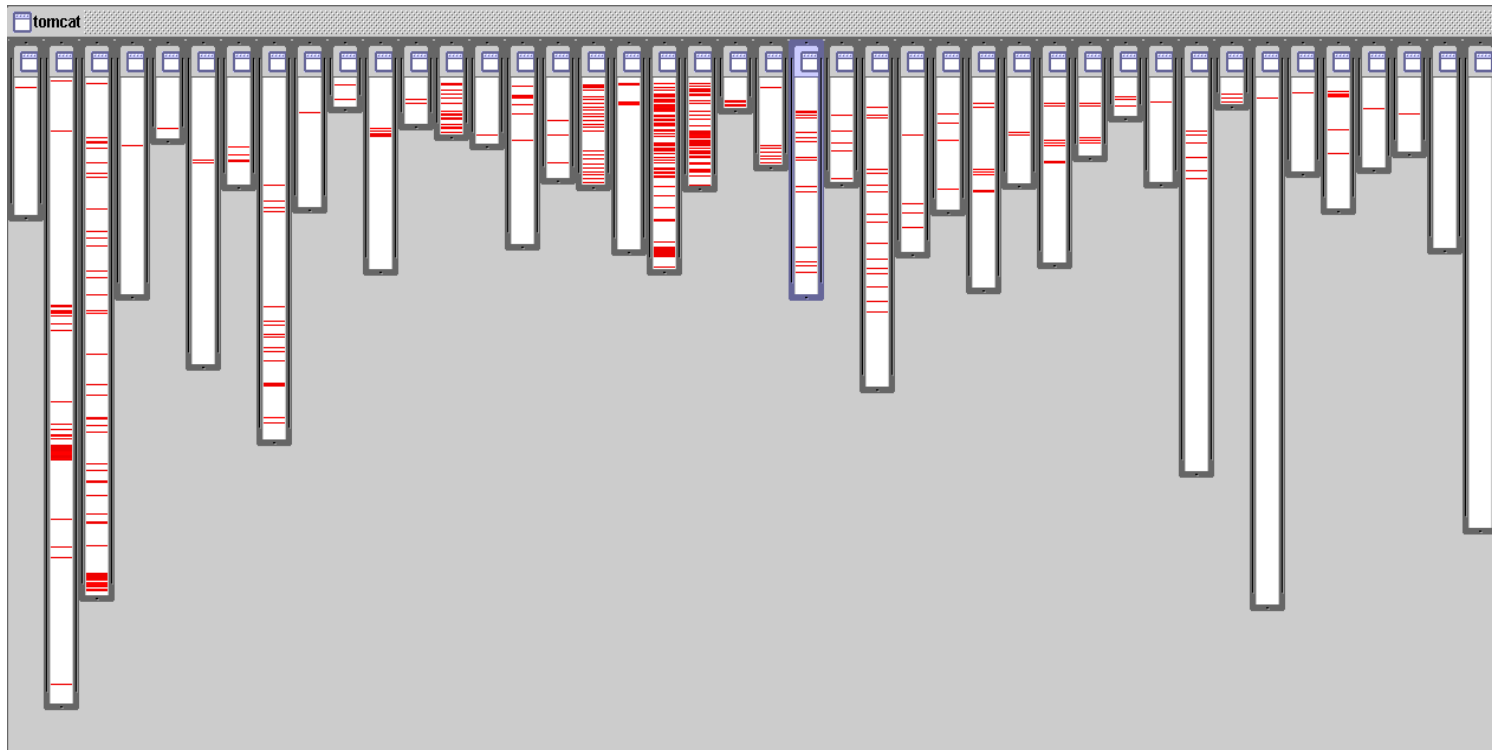
Separação de Interesses

- Os interesses apresentados no exemplo anterior (*logging*, persistência e tratamento de exceções) são conhecidos como Interesses Transversais.

Interesse Transversal (*crosscutting concern*) é o nome dado aos interesses que apresentam-se entrelaçados e espalhados com outros interesses do sistema.

Separação de Interesses

- Entrelaçamento e espalhamento em um software real.



Registro de *Logging* do servidor de aplicações *Tomcat*.

Separação de Interesses

- Quais impactos o entrelaçamento e espalhamento de interesses pode trazer?
 - Redução da legibilidade do software;
 - Aumento do acoplamento e redução da coesão entre os módulos;
 - Redução da manutenibilidade e reusabilidade do software.

Agenda

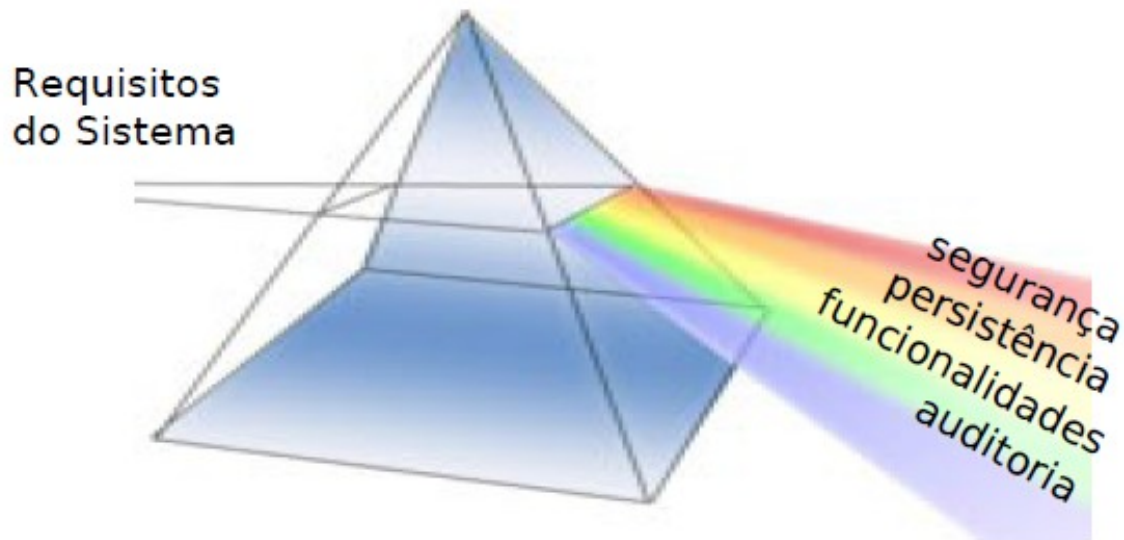
- Breve Histórico sobre Desenvolvimento de Software
- Separação de Interesses
- **Orientação a Aspectos**
- Identificação de Interesses Transversais
- Pesquisas: Mestrado e Doutorado
- Ideias para Projetos
- Comentários & Dúvidas

Orientação a Aspectos

- Solução para minimizar tais impactos negativos: encapsular interesses transversais em módulos independentes.
- Uma alternativa é utilizar **Orientação a Aspectos (OA)**.

Orientação a Aspectos

- A OA foi proposta por Gregor Kiczales em 1997.
- Objetivo: encapsular *Interesses Transversais* em módulos fisicamente separados, denominados **aspectos**, de outros módulos do software.



Orientação a Aspectos

- Como ficaria o código abaixo em OA?

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;
    private BD bd;

    public void sacar(double valor) throws SaldoInsuficienteException {
        if (saldo >= valor) {
            saldo = saldo - valor;
            log.exitLog("Saque efetuado na conta: " + numero);
            bd.salvar(this);
        } else {
            throw new
                SaldoInsuficienteException(this);
        }
    }
}
```

Orientação a Aspectos

```
public class Conta {
    private double saldo;
    private int numero;

    public void sacar(double valor) {
        saldo = saldo - valor;
    }
}

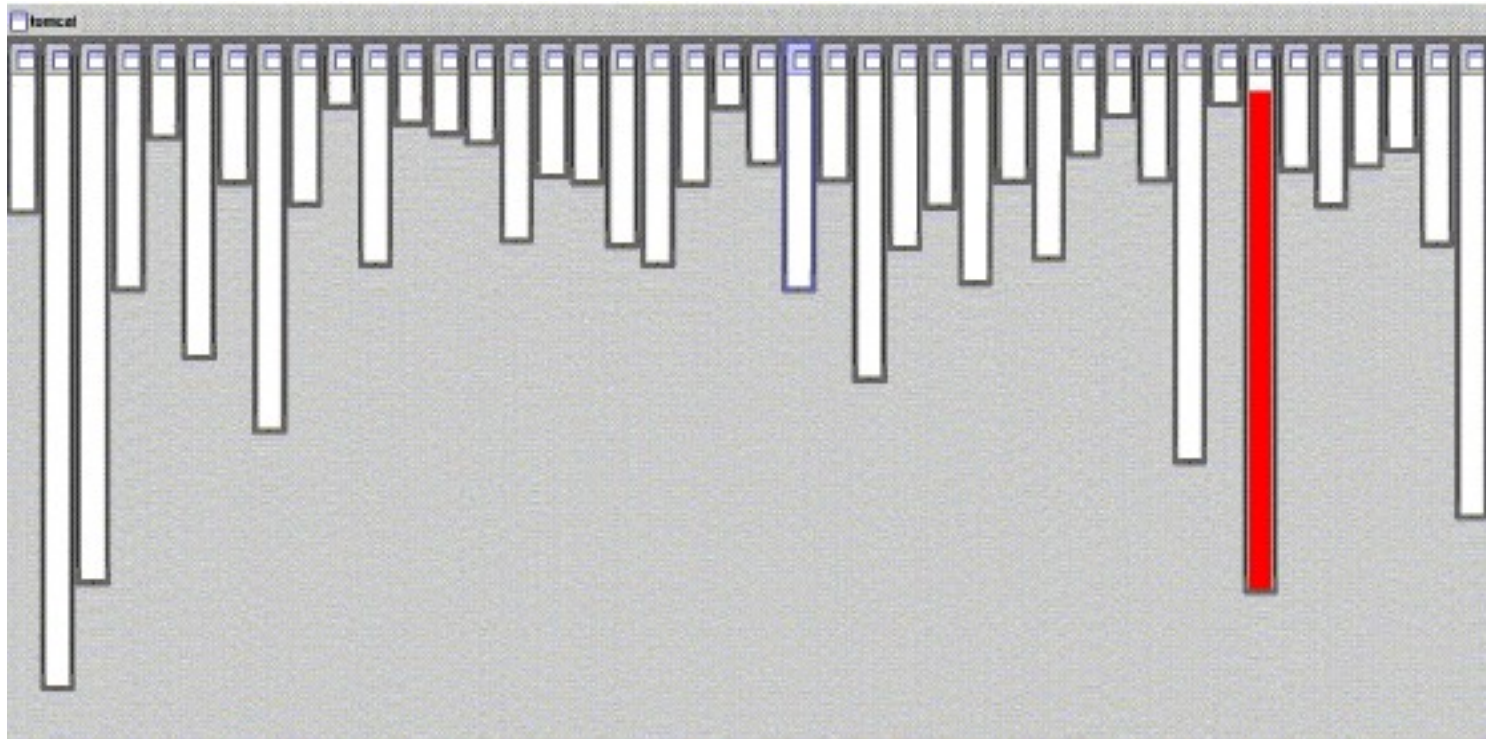
public aspect AspectoLogging {
    // código relacionado ao logging.
}

public aspect AspectoPersistencia {
    // código relacionado à persistência.
}

public aspect AspectoExcecao {
    // código relacionado ao tratamento
    // de exceções.
}
```

Legenda	
Cor	Interesses *
■	Lógica de Negócios
■	<i>Logging</i>
■	Persistência
■	Tratamento de Exceções

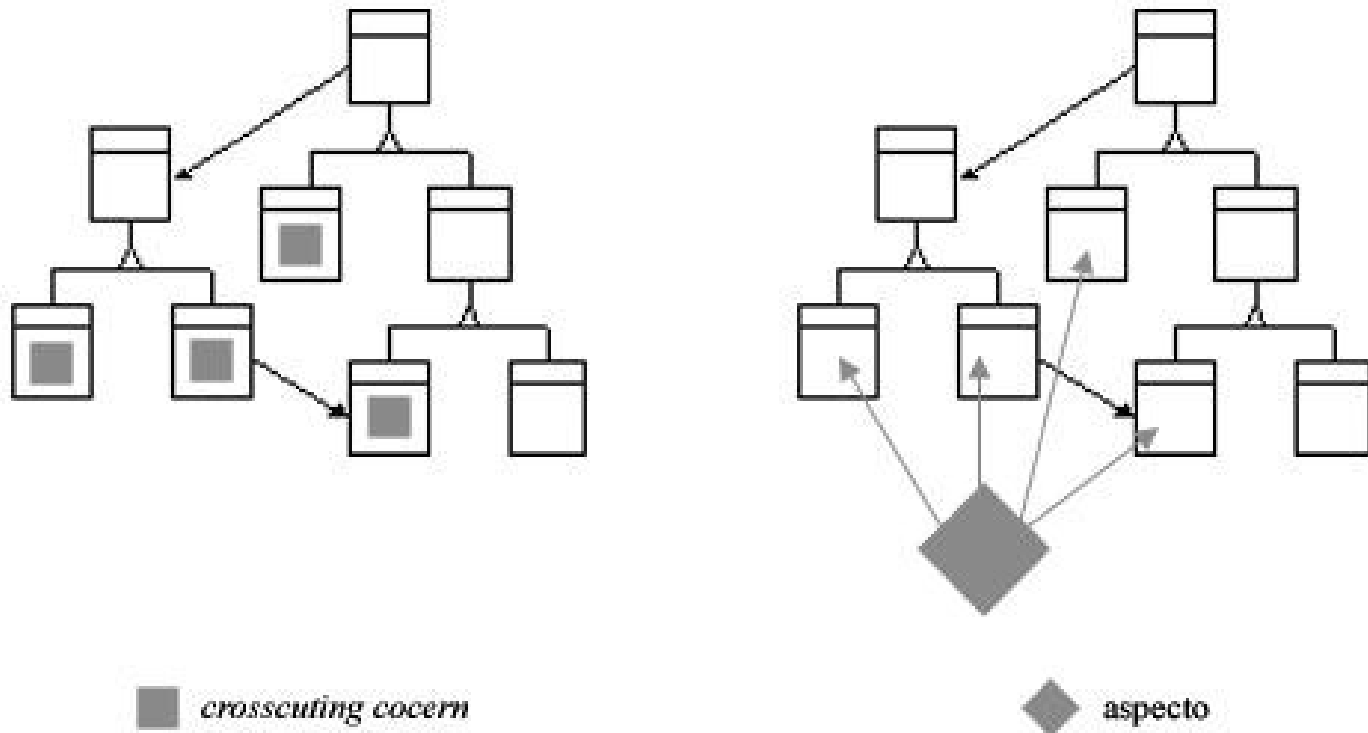
Orientação a Aspectos



Registro de *Logging* do servidor de aplicações *Tomcat* implementado em OA.

Orientação a Aspectos

- Sem Separação de Interesses vs. Com Separação de Interesses



Orientação a Aspectos

- [Exemplo de código em AspectJ](#): uma extensão da linguagem Java para Orientação a Aspectos.
- Veja como configurar o ambiente para executar o exemplo acima em: <http://www.eclipse.org/ajdt/>

Orientação a Aspectos

- Na realidade, “por baixo dos panos”, os interesses continuam entrelaçados e espalhados.
- Porém, mais uma vez isso é abstraído do trabalho do programador.

Orientação a Aspectos

Ponto de Vista do Programador

```
public class Conta {
    private double saldo;
    private int numero;

    public void sacar(double valor) {
        saldo = saldo - valor;
    }
}

public aspect AspectoLogging {
    // código relacionado ao logging.
}

public aspect AspectoPersistencia {
    // código relacionado à persistência.
}

public aspect AspectoExcecao {
    // código relacionado ao tratamento
    // de exceções.
}
```

Ponto de Vista da Máquina

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;
    private BD bd;
    public void sacar(double valor)
        throws SaldoInsuficienteException {
        if (saldo >= valor) {
            saldo = saldo - valor;
            log.exitLog("Saque efetuado
                na conta: " + numero);
            bd.salvar(this);
        } else {
            throw new
                SaldoInsuficienteException(this);
        }
    }
}
```

Orientação a Aspectos

- Onde OA está sendo utilizada?
 - **No meio acadêmico:**
 - AOSD (*International Conference on Aspect-Oriented Software Development*)
 - LA-WASP (*Latin American Workshop on Aspect-Oriented Software Development*)
 - AOM (*Aspect Oriented Modeling*)
 - **Na indústria:**
 - JBOSS/AOP
 - Spring/AOP
 - *Framework Demoiselle*

Agenda

- Breve Histórico sobre Desenvolvimento de Software
- Separação de Interesses
- Orientação a Aspectos
- **Identificação de Interesses Transversais**
- Pesquisas: Mestrado e Doutorado
- Ideias para Projetos
- Comentários & Dúvidas

Identificação de Interesses Transversais

- Antes de separar interesses transversais, o engenheiro de software precisa saber:
 - Quais são os interesses transversais existentes no software?
 - Onde eles se encontram implementados?
 - Como identificá-los?

Identificação de Interesses Transversais

- O problema é que, no mundo real, o código legado apresenta-se assim:

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;
    private BD bd;
    public void sacar(double valor) throws SaldoInsuficienteException {
        if (saldo >= valor) {
            saldo = saldo - valor;
            log.exitLog("Saque efetuado na conta: " + numero);
            bd.salvar(this);
        } else {
            throw new
                SaldoInsuficienteException(this);
        }
    }
}
```

Identificação de Interesses Transversais

- Nesse contexto aparece a área denominada **Identificação (ou Mineração) de Interesses Transversais**.
- Consiste em identificar trechos de código que contribuem para implementação de determinados interesses transversais.

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;
    private BD bd;
    public void sacar(double valor)
        throws SaldoInsuficienteException {
        if (saldo >= valor) {
            saldo = saldo - valor;
            log.exitLog("Saque efetuado
                na conta: " + numero);
            bd.salvar(this);
        } else {
            throw new
                SaldoInsuficienteException(this);
        }
    }
}
```



Identificação de Interesses Transversais

```
public class Conta {
    private double saldo;
    private int numero;
    private Logger log;
    private BD bd;
    public void sacar(double valor)
        throws SaldoInsuficienteException {
        if (saldo >= valor) {
            saldo = saldo - valor;
            log.exitLog("Saque efetuado
                na conta: " + numero);
            bd.salvar(this);
        } else {
            throw new
                SaldoInsuficienteException(this);
        }
    }
}
```

Identificação de Interesses Transversais

- Técnicas para Identificação de Interesses Transversais:
 - **Baseada em Texto (*tokens*):** por exemplo, encontrar todas as classes cujos nomes iniciam com “SQL”.
 - **Baseada em Tipos:** por exemplo, encontrar todas as classes que utilizam algum objeto do tipo “java.util.Connection”.
 - **Baseadas em Métricas:** por exemplo, encontrar todos os métodos cujo valor da métrica *X* seja maior do que 3.

Agenda

- Breve Histórico sobre Desenvolvimento de Software
- Separação de Interesses
- Orientação a Aspectos
- Identificação de Interesses Transversais
- **Pesquisas: Mestrado e Doutorado**
- Ideias para Projetos
- Comentários & Dúvidas

Pesquisas: Mestrado e Doutorado

- Filiação:

– *Advanced research group on Software Engineering (AdvanSE)*



– Programa de Pós-Graduação em Ciência da Computação (PPGCC)



– Universidade Federal de São Carlos (UFSCar)



Pesquisas: Mestrado e Doutorado

- Mestrado:
 - Uma das atividades desenvolvidas: adaptação de uma ferramenta para identificação de interesses transversais em código fonte Java.
 - ComSCId: *Computational Support for Concern Identification*.
 - *Plug-in* do Eclipse.
 - Permite o gerenciamento de regras para identificação de interesses transversais.

Pesquisas: Mestrado e Doutorado

- Mestrado:
 - ComSCId: resultado da identificação de interesses transversais.

Os trechos de código identificados são anotados com o símbolo de uma “prancheta”.

```
package mod
import java

public

double balance = 0;
Connection conn;

public void withdraw(double value) throws Ex
Class.forName("com.mysql.jdbc.Driver");
conn = DriverManager.getConnection("url"
balance -= value;
conn.close();

public void deposit(double value) throws Exc
Class.forName("com.mysql.jdbc.Driver");
conn = DriverManager.getConnection("url"
balance += value;
conn.close();
```

Esses são os indícios identificados a partir do cadastramento da interface “Connection” no conjunto de regras do ComSCId.

Pesquisas: Mestrado e Doutorado

- Doutorado:
 - Desenvolvimento de um processo para identificação e classificação de interesses em nível de requisitos.
 - No contexto da Engenharia de Requisitos Orientada a Aspectos (EROA), interesses transversais são conhecidos como “early aspects”.

Pesquisas: Mestrado e Doutorado

- Doutorado:
 - A EROA objetiva promover melhorias quanto à Separação de Interesses durante as fases iniciais do desenvolvimento do software.

“quanto mais tarde no ciclo de desenvolvimento do software se encontrar um erro, maior será o custo de sua correção”



“quanto mais tarde ocorrer a identificação dos interesses transversais, maior será o esforço necessário para sua modularização”

Pesquisas: Mestrado e Doutorado

- Doutorado:
 - Desafios:
 - baixa qualidade do documento de requisitos (incompleto, ambíguo, inconsistente);
 - baixa qualidade dos conjuntos de palavras-chaves utilizados para identificação dos interesses; e
 - dificuldade de se identificar certos tipos de interesses, por meio de palavras-chaves no texto. Exemplo: interesse de concorrência.

Agenda

- Breve Histórico sobre Desenvolvimento de Software
- Separação de Interesses
- Orientação a Aspectos
- Identificação de Interesses Transversais
- Pesquisas: Mestrado e Doutorado
- **Ideias para Projetos**
- Comentários & Dúvidas

Ideias para Projetos

- Propor novas estratégias e ferramentas para identificação de interesses transversais.
 - Utilização de repositórios de projetos anteriores;
 - Utilização de técnicas da área de Inteligência Artificial;
 - Entre outros;
- Realização de estudos comparativos de estratégias já existentes na literatura.

Comentários & Dúvidas



seminários abertos em computação

Obrigado!

Paulo Afonso Parreira Júnior

paulojunior@jatai.ufg.br