Aspect Oriented Software Development

Krzysztof Podlaski

Faculty of Physics and Applied Informatics,
University of Lodz, Poland
e-mail podlaski@uni.lodz.pl

Erasmus visit at:

CESI Bordeaux

20 June 2019, Bordeaux, France

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018

My University, My City, Poland

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century
- Rapid growth in XIX century - cotton industry

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century
- Rapid growth in XIX century - cotton industry
- In result known Secession style architecture - palaces of factory's owners

My University, My City, Poland

My University, My City, Poland

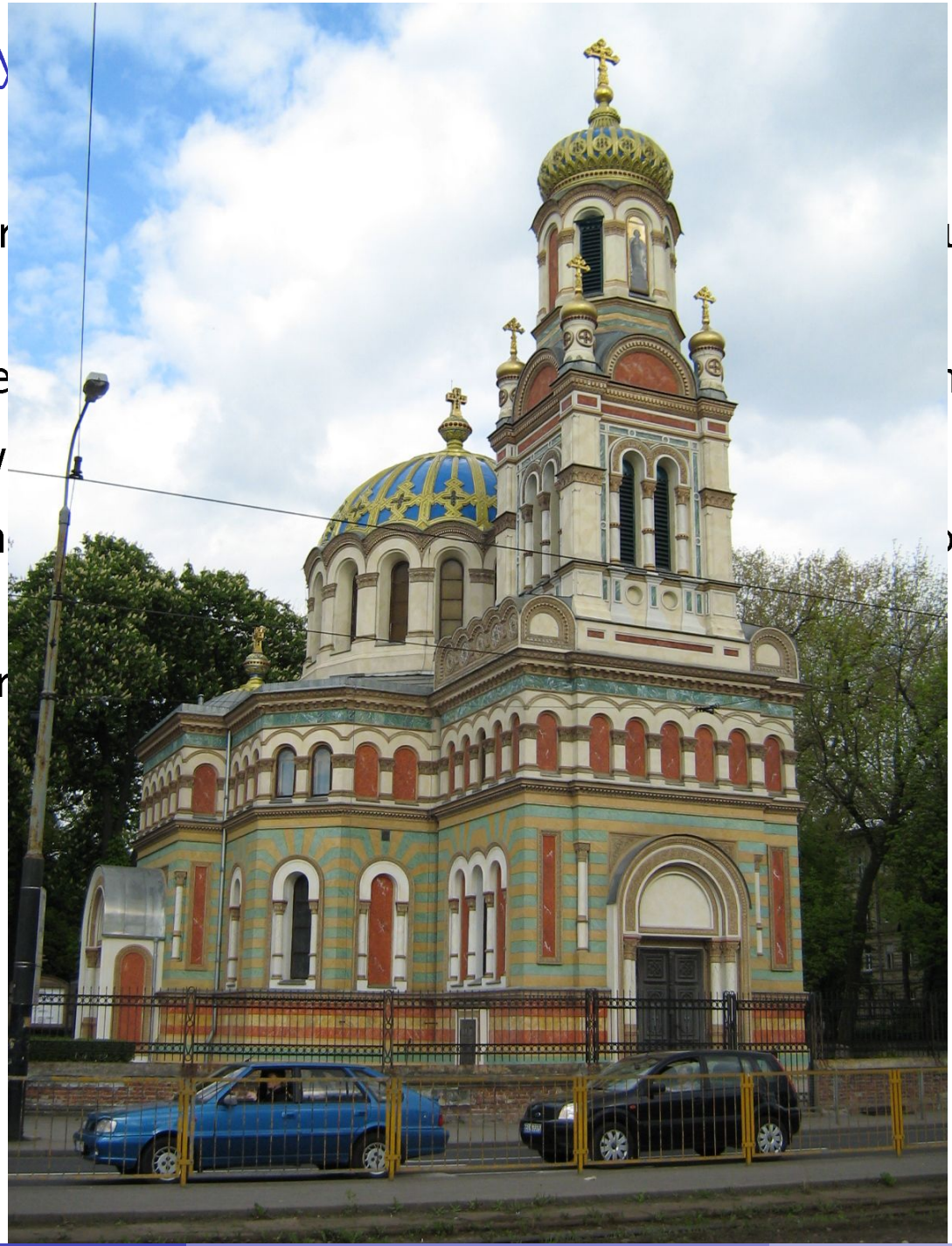
My University, My City, Poland

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century
- Rapid growth in XIX century - cotton industry
- In result known Secession style architecture - palaces of factory's owners
- City of four cultures (polish, german, jewish, russian)

My University

- Łódź is thir ut 685000 in 2018
- City founde ntury
- Rapid grow
- In result kn f factory's owners
- City of four



My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century
- Rapid growth in XIX century - cotton industry
- In result known Secession style architecture - palaces of factory's owners
- City of four cultures (polish, german, jewish, russian)
- Academic City: University, Technical University, Medical University, Academy of Fine Arts, National Film School, Academy of Music,
- about 74 thousands of students every academic year

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century
- Rapid growth in XIX century - cotton industry
- In result known Secession style architecture - palaces of factory's owners
- City of four cultures (polish, german, jewish, russian)
- Academic City: University, Technical University, Medical University, Academy of Fine Arts, National Film School, Academy of Music,
- about 74 thousands of students every academic year
- New ideas, renovation

My University, My City, Poland

My University, My City, Poland

My Unive

- Łódź i 5000 in 2018
- City fo
- Rapid
- In resu tory's owners
- City of
- Acader niversity, Acader Music,
- about
- New id

My University, My City, Poland

- Łódź is third city in Poland by number of citizens about 685000 in 2018
- City founded in XIV as village, city rights since XV century
- Rapid growth in XIX century - cotton industry
- In result known Secession style architecture - palaces of factory's owners
- City of four cultures (polish, german, jewish, russian)
- Academic City: University, Technical University, Medical University, Academy of Fine Arts, National Film School, Academy of Music,
- about 74 thousands of students every academic year
- New ideas, renovation
- Companies of new technologies - IT governs.

Programming paradigms

- High level languages
  - We are not interested in Assembler,
  - At Least Here
- Paradigms

Programming paradigms

- High level languages
  - We are not interested in Assembler,
  - At Least Here
- Paradigms
  - Procedural

Programming paradigms

- High level languages
  - We are not interested in Assembler,
  - At Least Here
- Paradigms
  - Procedural
  - Object Oriented

Programming paradigms

- High level languages
  - We are not interested in Assembler,
  - At Least Here
- Paradigms
  - Procedural
  - Object Oriented
  - Functional

Procedural Programming

- Good in many cases, but...

Procedural Programming

- Good in many cases, but...
  - Mixes all possible things in one

Procedural Programming

- Good in many cases, but...
  - Mixes all possible things in one
  - Context problems

Procedural Programming

- Good in many cases, but...
  - Mixes all possible things in one
  - Context problems
  - Bunch of global variables

Procedural Programming

- Good in many cases, but...
  - Mixes all possible things in one
  - Context problems
  - Bunch of global variables
  - Complicated to maintain

Procedural Programming

- Good in many cases, but...
  - Mixes all possible things in one
  - Context problems
  - Bunch of global variables
  - Complicated to maintain
  - Long unreadable documentation

Procedural Programming

- Good in many cases, but...
  - Mixes all possible things in one
  - Context problems
  - Bunch of global variables
  - Complicated to maintain
  - Long unreadable documentation

Object Oriented Programming

- Classes
  - Elements that encapsulate some information, behavior

Object Oriented Programming

- Classes
  - Elements that encapsulate some information, behavior
- Objects
  - Instances of classes

Object Oriented Programming

- Classes
  - Elements that encapsulate some information, behavior
- Objects
  - Instances of classes
- Program $\Rightarrow$ Set of Objects
- Main Properties
  - Abstraction

Object Oriented Programming

- Classes
  - Elements that encapsulate some information, behavior
- Objects
  - Instances of classes
- Program $\Rightarrow$ Set of Objects
- Main Properties
  - Abstraction
  - Encapsulation

Object Oriented Programming

- Classes
  - Elements that encapsulate some information, behavior
- Objects
  - Instances of classes
- Program $\Rightarrow$ Set of Objects
- Main Properties
  - Abstraction
  - Encapsulation
  - Inheritance

Object Oriented Programming

- Classes
  - Elements that encapsulate some information, behavior
- Objects
  - Instances of classes
- Program $\Rightarrow$ Set of Objects
- Main Properties
  - Abstraction
  - Encapsulation
  - Inheritance
  - Polymorphism

Abstraction

Programming and THE BIG PICTURE

- Assembler (No abstraction at all)
- Procedural (Little abstraction)
  - Programmer is drowned in little cases

Abstraction

Programming and THE BIG PICTURE

- Assembler (No abstraction at all)
- Procedural (Little abstraction)
  - Programmer is drowned in little cases
- OOP
  - First design your (basic application elements)

Abstraction

Programming and THE BIG PICTURE

- Assembler (No abstraction at all)
- Procedural (Little abstraction)
  - Programmer is drowned in little cases
- OOP
  - First design your (basic application elements)
  - Build your program from them

Abstraction

Programming and THE BIG PICTURE

- Assembler (No abstraction at all)
- Procedural (Little abstraction)
  - Programmer is drowned in little cases
- OOP
  - First design your (basic application elements)
  - Build your program from them
  - Is that enough ?

Object Oriented Design

- Well known rules how to proceed
- SOLID
  - Single responsibility principle

Object Oriented Design

- Well known rules how to proceed
- SOLID
  - Single responsibility principle
- GRASP
  - High cohesion

Object Oriented Design

- Well known rules how to proceed
- SOLID
  - Single responsibility principle
- GRASP
  - High cohesion
- K.I.S.S (Keep it simple, stupid)

Object Oriented Design

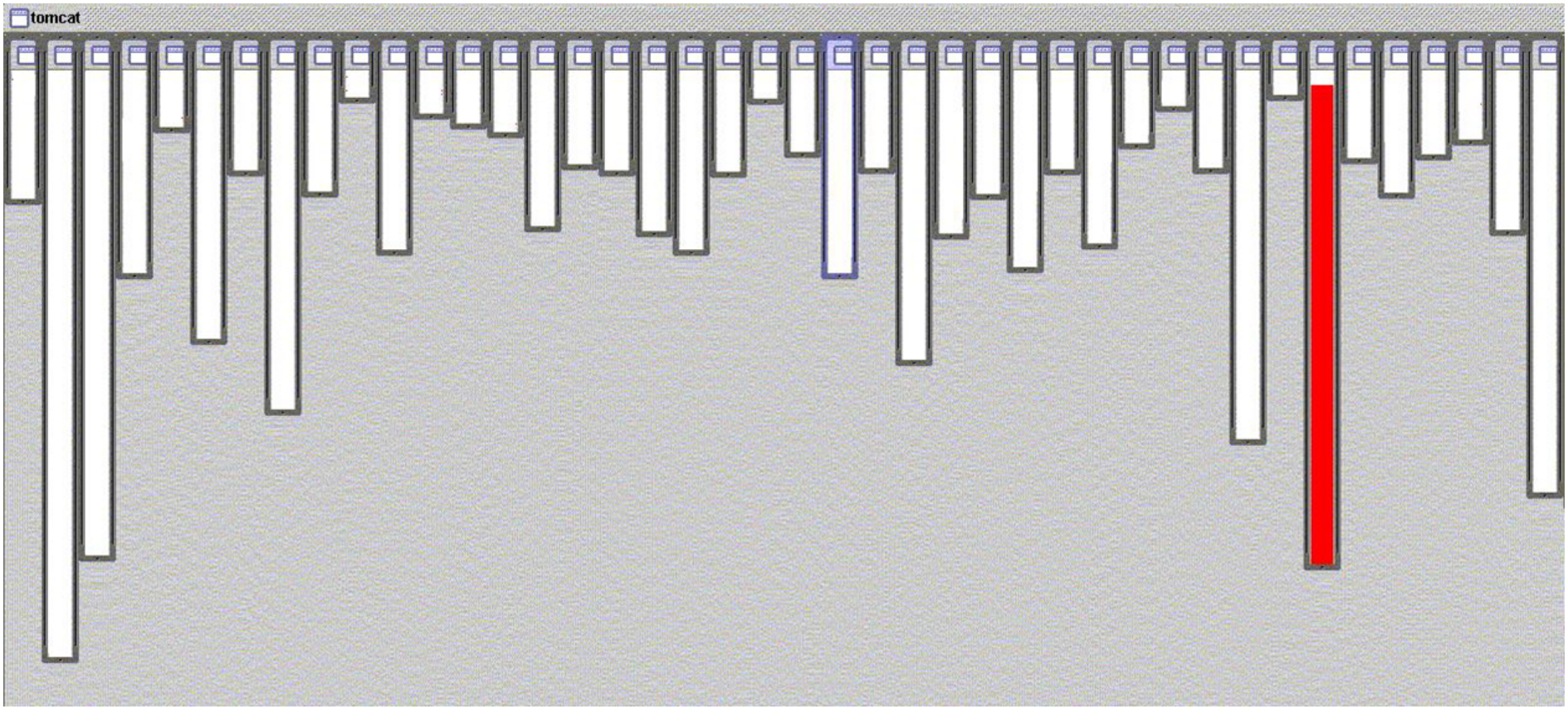
- Well known rules how to proceed
- SOLID
  - Single responsibility principle
- GRASP
  - High cohesion
- K.I.S.S (Keep it simple, stupid)
- DRY (Don't repeat yourself)

Object Oriented Design

- Well known rules how to proceed
- SOLID
  - Single responsibility principle
- GRASP
  - High cohesion
- K.I.S.S (Keep it simple, stupid)
- DRY (Don't repeat yourself)
- and many more rules

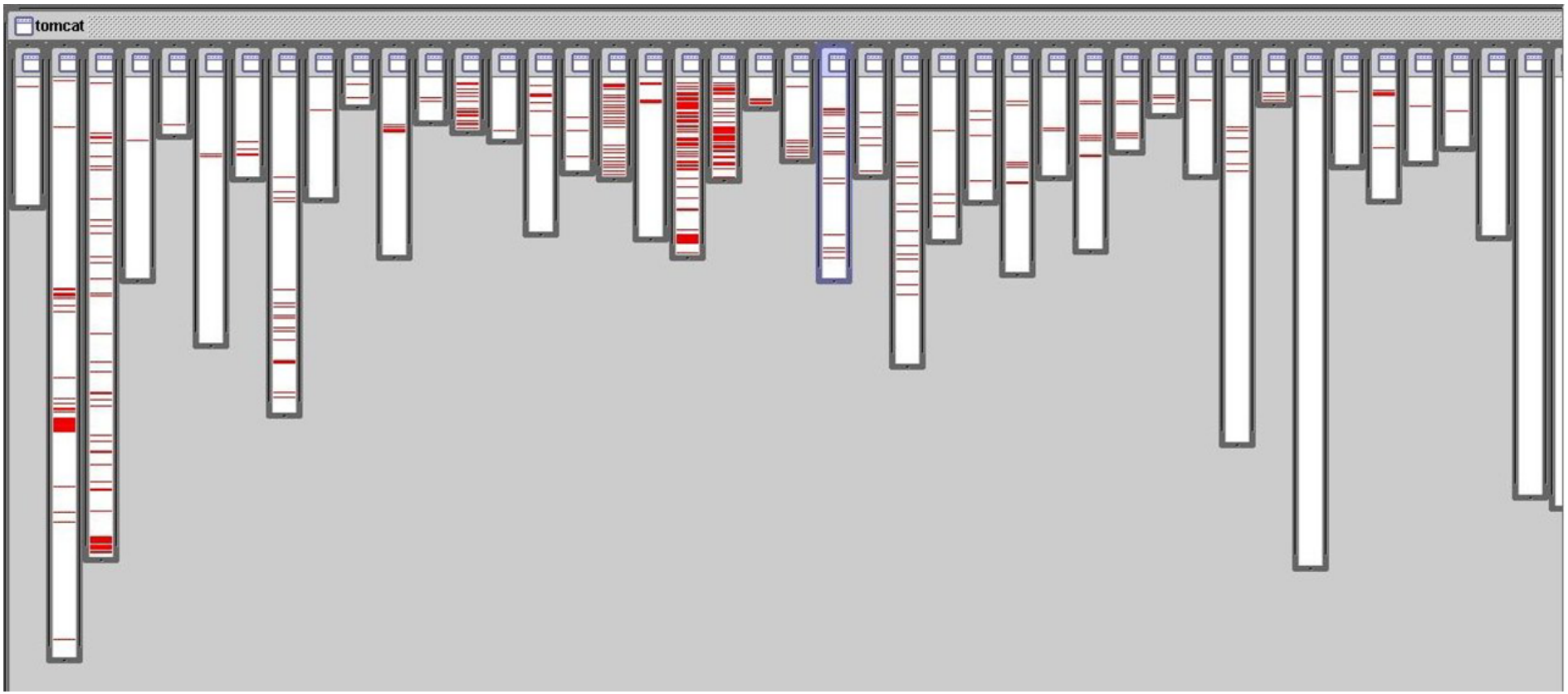
OOP and THE BIG PICTURE

Parsing XML in org.apache.tomcat (source aspecj.org)



OOP and THE BIG PICTURE

Logging in org.apache.tomcat (source aspecj.org)



Abstraction (continuation)

- Domain modeling
  - Used in Web Development

Abstraction (continuation)

- Domain modeling
  - Used in Web Development
  - Split your application in domains of interest

Abstraction (continuation)

- Domain modeling
  - Used in Web Development
  - Split your application in domains of interest
    - ★ Each can have many Classes
    - ★ Interaction between them

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution
    - Read/write to a field

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution
    - Read/write to a field
    - Exception thown out

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution
    - Read/write to a field
    - Exception thown out
  - Pointcut
    - Set of selected join points

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution
    - Read/write to a field
    - Exception thown out
  - Pointcut
    - Set of selected join points
    - Places we'd like to do something

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution
    - Read/write to a field
    - Exception thown out
  - Pointcut
    - Set of selected join points
    - Places we'd like to do something
- Advices

AOP vocabulary

Basic elements

- Cross-cutting concerns
  - Problems related to many (all) classes in application
- Join Points
  - Well defined places in application flow
    - Method call/execution
    - Constructor call/execution
    - Read/write to a field
    - Exception thown out
  - Pointcut
    - Set of selected join points
    - Places we'd like to do something
- Advices
  - Additional tasks to do at joint points

Aspect and programming language

Do we have to change programming language ?

Aspect and programming language

Do we have to change programming language ?

- Just a little

Aspect solutions in known languages

C/C++	-	AspectC++,XWeaver,Aspectc,...
C#	-	LOOmNET, ASpect#,...
JavaScript	-	Ajaxpect, jQUery AOP,...
PHP	-	PHPaspect, Seasar.PHP,...
Java	-	AspectJ, Spring AOP, JBoss AOP,...

How it works

- We have program in preferred language

How it works

- We have program in preferred language
- We add aspects in selected language/frameworkpause
  - Recompile/Rebuild our application
    - ★ Look as native preferred
    - ★ Java + AspectJ => usual bytecode

How it works

- We have program in preferred language
- We add aspects in selected language/frameworkpause
  - Recompile/Rebuild our application
    - Look as native preferred
    - Java + AspectJ => usual bytecode
  - JVM can't see the difference

How it works

- We have program in preferred language
- We add aspects in selected language/frameworkpause
  - Recompile/Rebuild our application
    - Look as native preferred
    - Java + AspectJ => usual bytecode
  - JVM can't see the differenceChanges in fly
    - Change in Class made during Class Loading

Simple Example

- Logging - usually the flag example of Aspects
  - We will do it later
- Simple HelloWord type Java App

Simple Example

- Logging - usually the flag example of Aspects
  - We will do it later
- Simple HelloWord type Java App
- Now aspect elements

Simple Example

- Logging - usually the flag example of Aspects
  - We will do it later
- Simple HelloWord type Java App
- Now aspect elements
  - Define join points

Simple Example

- Logging - usually the flag example of Aspects
  - We will do it later
- Simple HelloWord type Java App
- Now aspect elements
  - Define join points
  - Define advices

Simple Example

- Logging - usually the flag example of Aspects
  - We will do it later
- Simple HelloWord type Java App
- Now aspect elements
  - Define join points
  - Define advices
- All examples are on github
- `https://github.com/kpodlaski/CESI_Bordeaux_2019/tree/AOP`
  - `https://github.com/kpodlaski`
  - Repository : `CESI_Bordeaux_2019`
  - Branch `AOP`

Cut points in details

- Cut-points definitions:

Cut points in details

- Cut-points definitions:
  - execution
    - ★ Fired when method is executed

Cut points in details

- Cut-points definitions:
  - execution
    - ★ Fired when method is executed
  - call
    - ★ Fired when method is called
    - ★ Used often with a constructor

Cut points in details

- Cut-points definitions:
  - execution
    - ★ Fired when method is executed
  - call
    - ★ Fired when method is called
    - ★ Used often with a constructor
  - handler
    - ★ Used with Exceptions

Cut points in details

- Cut-points definitions:
  - execution
    - ★ Fired when method is executed
  - call
    - ★ Fired when method is called
    - ★ Used often with a constructor
  - handler
    - ★ Used with Exceptions
  - this, target, within, cflow
    - ★ This we will cover later

Cut points in details

- Cut-points definitions:
  - execution
    - ★ Fired when method is executed
  - call
    - ★ Fired when method is called
    - ★ Used often with a constructor
  - handler
    - ★ Used with Exceptions
  - this, target, within, cflow
    - ★ This we will cover later
  - logic operators AND, OR

Advices

- We can use the following types of advices:
  - after

Advices

- We can use the following types of advices:
  - after
  - after returning

Advices

- We can use the following types of advices:
  - after
  - after returning
  - after throwing

Advices

- We can use the following types of advices:
  - after
  - after returning
  - after throwing
  - before

Advices

- We can use the following types of advices:
  - after
  - after returning
  - after throwing
  - before
  - around
    - ★ Very special one, can replace real method

Advices

- We can use the following types of advices:
  - after
  - after returning
  - after throwing
  - before
  - around
    - ★ Very special one, can replace real method

Monitoring arguments

- How to know what values of arguments metod has?

Monitoring arguments

- How to know what values of arguments metod has?
  - args(...)

Monitoring arguments

- How to know what values of arguments metod has?
  - args(...)
- How to get to the state of the watched object?

Monitoring arguments

- How to know what values of arguments metod has?
  - args(...)
- How to get to the state of the watched object?
  - target(...)

Monitoring arguments

- How to know what values of arguments metod has?
  - args(...)
- How to get to the state of the watched object?
  - target(...)

Additional useful operation

- thisJoinPoint
  - Reference to joint point in given advice instance

Other Examples

- Instead of the best know aspect example
  - logging example

Other Examples

- Instead of the best know aspect example
  - logging example
- Let's try create access policy.

Other Examples

- Instead of the best know aspect example
  - logging example
- Let's try create access policy.
- ...
- Performance restriction rules

What we can do with aspects

- Almost everything:
  - Change the classes/objects

What we can do with aspects

- Almost everything:
  - Change the classes/objects
    - ★ Add fields, methods

What we can do with aspects

- Almost everything:
  - Change the classes/objects
    - ★ Add fields, methods
    - ★ Change hierarchy of inheritance

What we can do with aspects

- Almost everything:
  - Change the classes/objects
    - ★ Add fields, methods
    - ★ Change hierarchy of inheritance
  - Add additional behaviour

What we can do with aspects

- Almost everything:
  - Change the classes/objects
    - ★ Add fields, methods
    - ★ Change hierarchy of inheritance
  - Add additional behaviour
  - Change methods behaviour

What we can do with aspects

- Almost everything:
  - Change the classes/objects
    - ★ Add fields, methods
    - ★ Change hierarchy of inheritance
  - Add additional behaviour
  - Change methods behaviour
  - Create new types/classes

AOP in Software Development

- Design
  - Slight impact of aspects on design

AOP in Software Development

- Design
  - Slight impact of aspects on design
- Coding
  - Just code program + aspects

AOP in Software Development

- Design
  - Slight impact of aspects on design
- Coding
  - Just code program + aspects
- Testing
  - We can use logging, bug tracing aspects etc.
  - We can/should test aspects too

AOP in Software Development

- Design
  - Slight impact of aspects on design
- Coding
  - Just code program + aspects
- Testing
  - We can use logging, bug tracing aspects etc.
  - We can/should test aspects too
- Maintain
  - Less work with changes in cross-cutting concerns

AOP in real life

- Spring Framework
  - used in background, developer do not have to know it

AOP in real life

- Spring Framework
  - used in background, developer do not have to know it
- Instrumentation of the code
  - Performance live measurements
  - Faults/flaws detection

The End?

- No
  - Just tip of Iceberg

The End?

- No
  - Just tip of Iceberg
- Literature:
  - Gregor Kichales et al., Aspect-oriented programming, (1997)
  - Jacobson, Ivar, and Pan-Wei Ng, Aspect-oriented software development with use cases, Addison-Wesley Professional, 2004.
  - Colyer, Adrian, et al. Eclipse aspectj: aspect-oriented programming with aspectj and the eclipse aspectj development tools, Addison-Wesley Professional, 2004.
  - Laddad, Ramnivas, Aspectj in action: enterprise AOP with spring applications. Manning Publications Co., 2009.
  - Alshareef, Sohil F., et al., Aspect-oriented requirements engineering: approaches and techniques, Proceedings of the First International Conference on Data Science, E-learning and Information Systems. ACM, 2018.
  - Articles in the Internet

Thank you for your attention