



# Comparative Analysis Of Strategies For Applying AOP On Legacy Code

---

Forth Turkish Aspect-Oriented Software  
Development Workshop(Bilkent Ankara 12/2009)

***Duygu Sarıkaya***  
***Can Ufuk Hantaş***  
***Dilek Demirbaş***



# Outline

---

- Introduction
- Case Study
- Strategies
  - Direct Application Of Aspects To Legacy Code Without Refactoring
  - Dealing crosscutting concerns by Object-Oriented re-factoring without AOP
  - Applying AOP After Object-Oriented Re-factoring
- Conclusion



# Problem Definition

---

- Object Oriented Programming
- Design Patterns
- Aspect Oriented Programming



# Problem Definition

---

- Problems with Existing Code



# The Goal

---

- Investigate the Usage of AOP in existing code.



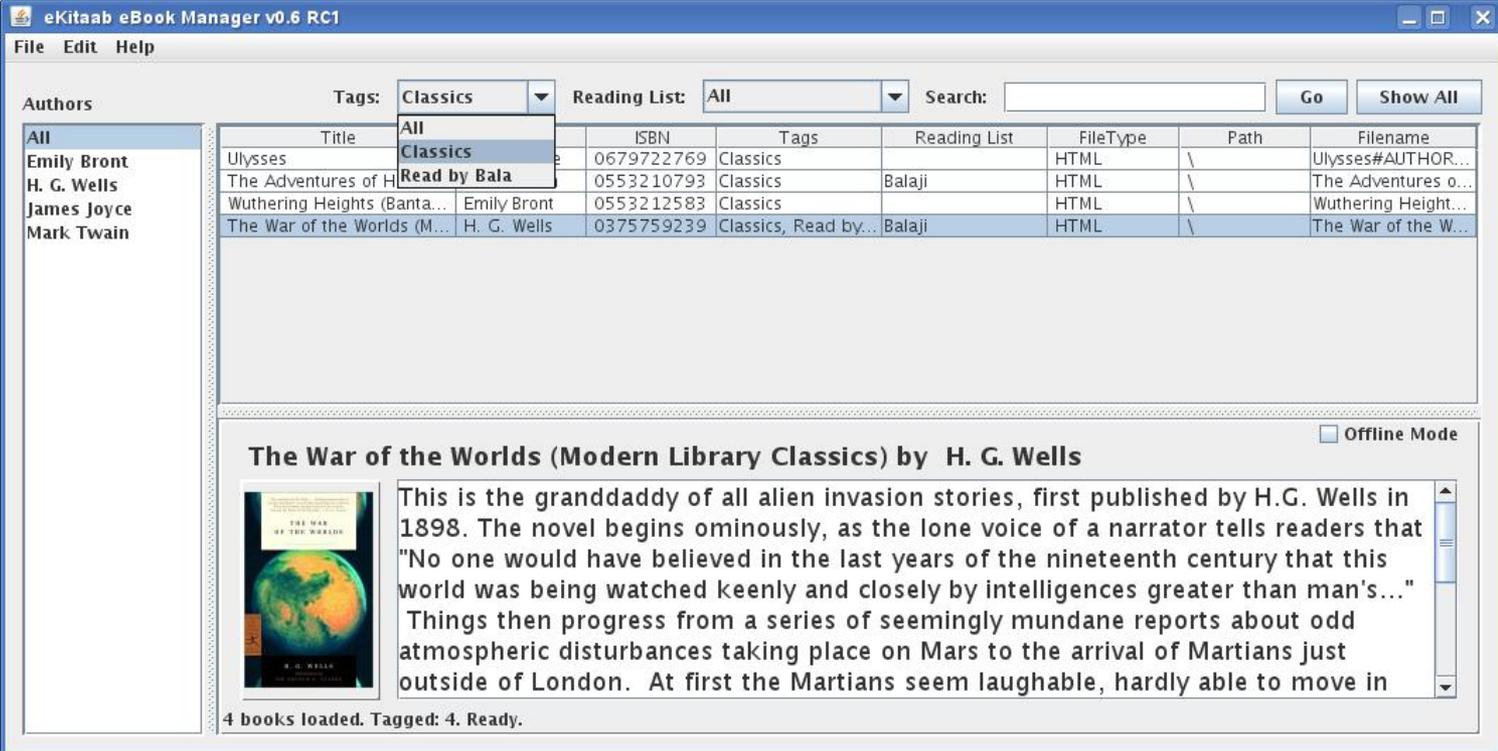
# Our Approach

---

- Use Aspect Oriented Approach
  - Not only as complementary to the shortages of OOP programming
  - But also solely to create solutions in a fast manner
    - Which saves
      - Time
      - Effort
      - Cost

# Case Study

## ○ E-Book Manager Application



The screenshot shows the eKitaab eBook Manager v0.6 RC1 application window. The interface includes a menu bar (File, Edit, Help), a search bar, and a table of books. The 'Tags' dropdown is set to 'Classics', and the 'Reading List' is set to 'All'. The table lists books such as 'Ulysses', 'The Adventures of Huckleberry Finn', 'Wuthering Heights', and 'The War of the Worlds'. The 'The War of the Worlds' entry is selected, and its details are shown in a preview pane below the table. The preview pane includes the book's cover and a description.

**Authors:** All, Emily Bront, H. G. Wells, James Joyce, Mark Twain

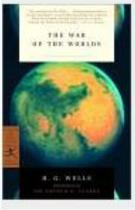
**Tags:** Classics (dropdown menu open showing: All, Classics, Read by Bala)

**Reading List:** All (dropdown menu)

**Search:** [ ] **Go** **Show All**

Title	ISBN	Tags	Reading List	FileType	Path	Filename
Ulysses	0679722769	Classics		HTML	\	Ulysses#AUTHOR...
The Adventures of H...	0553210793	Classics	Balaji	HTML	\	The Adventures o...
Wuthering Heights (Banta...)	0553212583	Classics		HTML	\	Wuthering Height...
The War of the Worlds (M...)	0375759239	Classics, Read by...	Balaji	HTML	\	The War of the W...

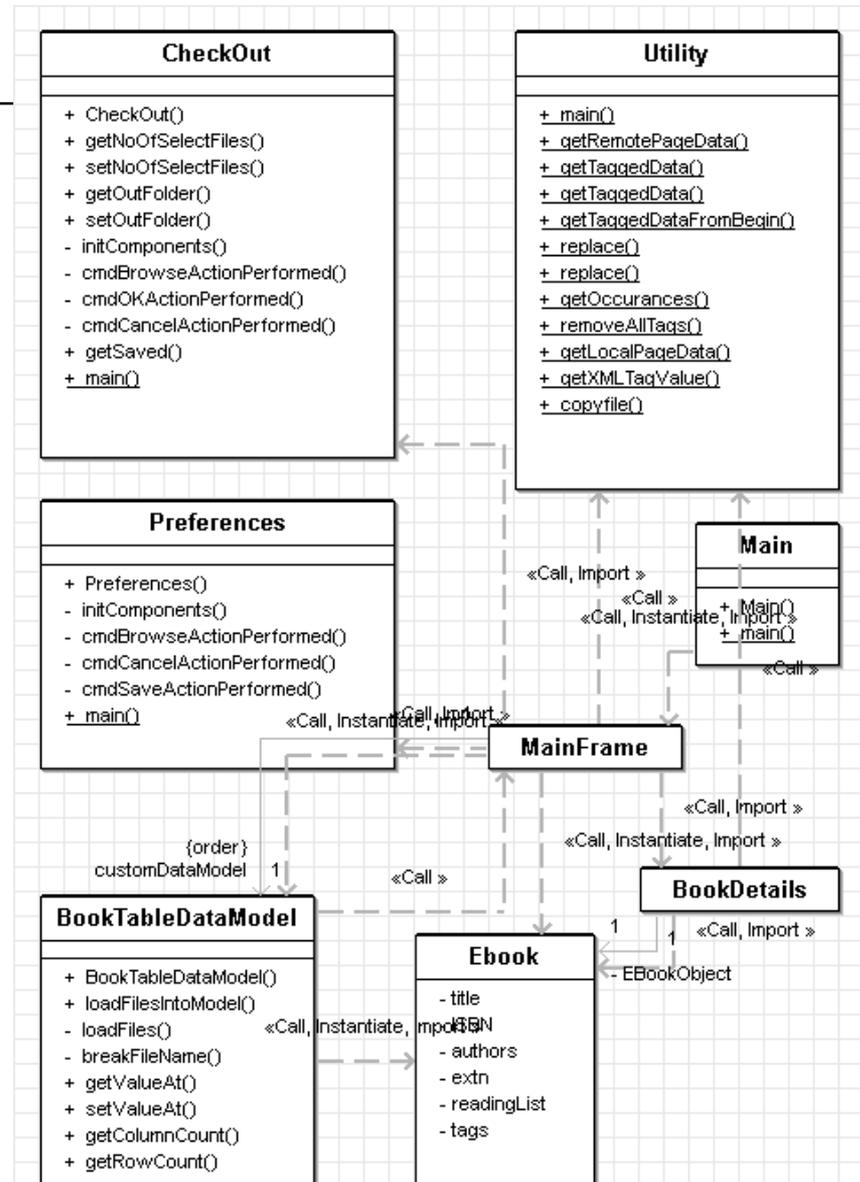
**The War of the Worlds (Modern Library Classics) by H. G. Wells**  Offline Mode

 This is the granddaddy of all alien invasion stories, first published by H.G. Wells in 1898. The novel begins ominously, as the lone voice of a narrator tells readers that "No one would have believed in the last years of the nineteenth century that this world was being watched keenly and closely by intelligences greater than man's..." Things then progress from a series of seemingly mundane reports about odd atmospheric disturbances taking place on Mars to the arrival of Martians just outside of London. At first the Martians seem laughable, hardly able to move in

4 books loaded. Tagged: 4. Ready.

# Case Study: E-Book Manager Application

## ○ Class Diagram





# Our Strategy

---

- Direct Application Of Aspects To Legacy Code Without Refactoring
  - Existing Concerns
  - New Concerns
- Dealing crosscutting concerns by Object-Oriented re-factoring without AOP
  - Existing Concerns
  - New Concerns
- Applying AOP After Object-Oriented Re-factoring
  - Existing Concerns
  - New Concerns



## Our Strategies: Direct Application Of Aspects To Legacy Code Without Refactoring

---

- Existing Concerns
  - Detail Storage
  - Checkout
  - Web Service
- New Concerns
  - Internationalization
  - Look&Feel

# Existing Concerns

## :Detail Storage

---

- By Using Aspect Oriented Approach
  - 3 point cuts – 3 advices
    - Save Details
      - Saving Ebook objects with file streams
    - Load Details
      - Loading files and casting them to Ebook objects
    - Copy File
      - Copying detail storage files along with Ebook files

# Existing Concerns

## :Detail Storage

---

```
public aspect DetailStorage {  
    pointcut saveBook(Ebook book, int i, MainFrame mf) : call(* MainFrame.SaveBook(Ebook , int)) &&  
        args(book,i) && target(mf);  
    pointcut loadBook(String fileName, BookTableDataModel btdm) : call(*  
        BookTableDataModel.breakFileName(String)) && args(fileName) && target(btdm);  
    pointcut copyFile(String src, String dst): call(* Utility.copyfile(..)) && args(src, dst) &&  
        this(FileCopyCheckOut);  
    declare parents: Ebook implements Serializable; //use serializable interface to store book information  
    //Advice for overriding method calls to breakFileName method to get boot information from file name and  
    create book object  
    Ebook around(String s, BookTableDataModel btdm) : loadBook(s, btdm){  
        ...  
    }  
    ...  
}
```



# Existing Concerns :Checkout

---

- By Using Aspect Oriented Approach
  - Menu integration
  - Implementation

# Existing Concerns

## :Checkout

---

```
public aspect CheckOutManager {

    pointcut checkOut(MainFrame mf) : call(* MainFrame.CheckoutFiles()) && target(mf);
    pointcut checkOutFileMenuFunction(MainFrame mf): call(* MainFrame.mnuCheckoutActionPerformed(*)) &&
        target(mf);
    pointcut menuCreation(MainFrame mf): call(* MainFrame.initComponents()) && this(mf);

    //override checkOut function so that it will work according to selected check out manager
    void around(MainFrame mf) : checkOut(mf) {

        ...
    }

    //Create send-by-email menu
    after(final MainFrame mf): menuCreation(mf){

        ...
    }

    //Set manager to fileCopyManager before menu action performed
    before(MainFrame mf):checkOutFileMenuFunction(mf){

        ...
    }
}
```



# Existing Concerns

:Web Service

---

:Aspect code of Web Service



# Our Strategies: New Concerns

---

- Dealing with crosscutting concerns that come with adding related functionalities
  - Internatinalization
  - Look and Feel
  - Agent Based Communication



# New Concerns: Internatinalization

---

- Swing and AWT elements
  - setText() method
  - Language files

# New Concerns:Internationalization

---

**public aspect** internationalization

{

**pointcut** setText(String s,JComponent comp):**call**(\* \*.setText(String)) && **args**(s) && **target**(comp) &&  
**!this**(internationalization);

**void around**(String s, JComponent  
comp): setText(s, comp) {

//some variable declarions

...

locale = **new** Locale("tr","TR");

rb =ResourceBundle.getBundle("Messages"  
, locale);

//getting locale value according to keys

...

**if**(comp **instanceof** JMenuItem){

JMenuItem jm=(JMenuItem)comp;

jm.setText(s);

}

// check for other types

...

}



# New Concerns: Look and Feel

---

- Swing look & feels
- OS dependent visual style
- Component type specific style

# New Concerns: Look and Feel

---

```
public aspect lookandfeel {  
pointcut catchButton(Jbutton j): set(JButton *.* ) && args(j);  
after(Jbutton j): catchButton(j){  
int osType =DetailStorage.osType();  
switch (osType){  
case 0://Unknown OS  
break;  
case 1://Windows  
j.setUI(new  
javax.swing.plaf.basic.BasicButtonUI());  
break;  
case 2://Mac  
...  
break;  
case 3://Unix, Linux  
j.setUI(new javax.swing.plaf.metal.MetalButtonUI());  
break;
```

# New Concerns: Agent Based Communication

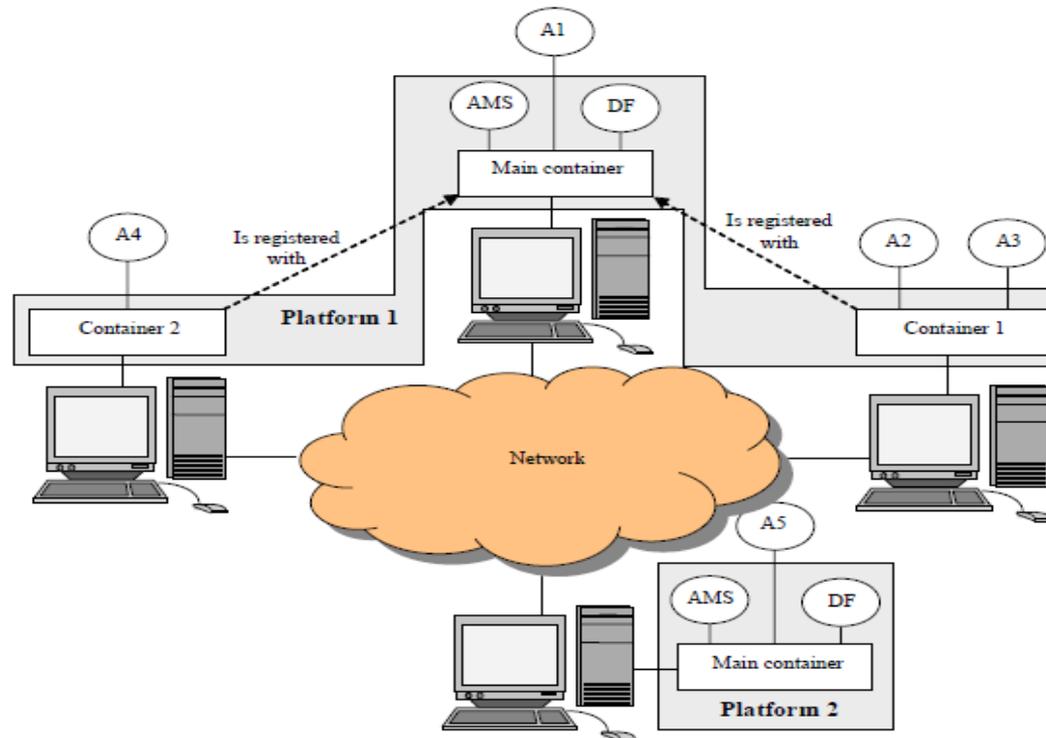


Figure 1 Containers and Platforms

# New Concerns: Agent Based Communication

---

```
public aspect DistributedSellerAgent {

    pointcut loadFilesIntoModel(BookTableDataModel btdm) : call(*
        BookTableDataModel.loadFilesIntoModel(..) && target(btdm);

    //Advice to create a seller agent after execution of the pointcut
    after(BookTableDataModel btdm) : loadFilesIntoModel(btdm){
        ...
    }
}
```



## Our Strategies: Dealing crosscutting concerns by Object-Oriented re-factoring without AOP

---

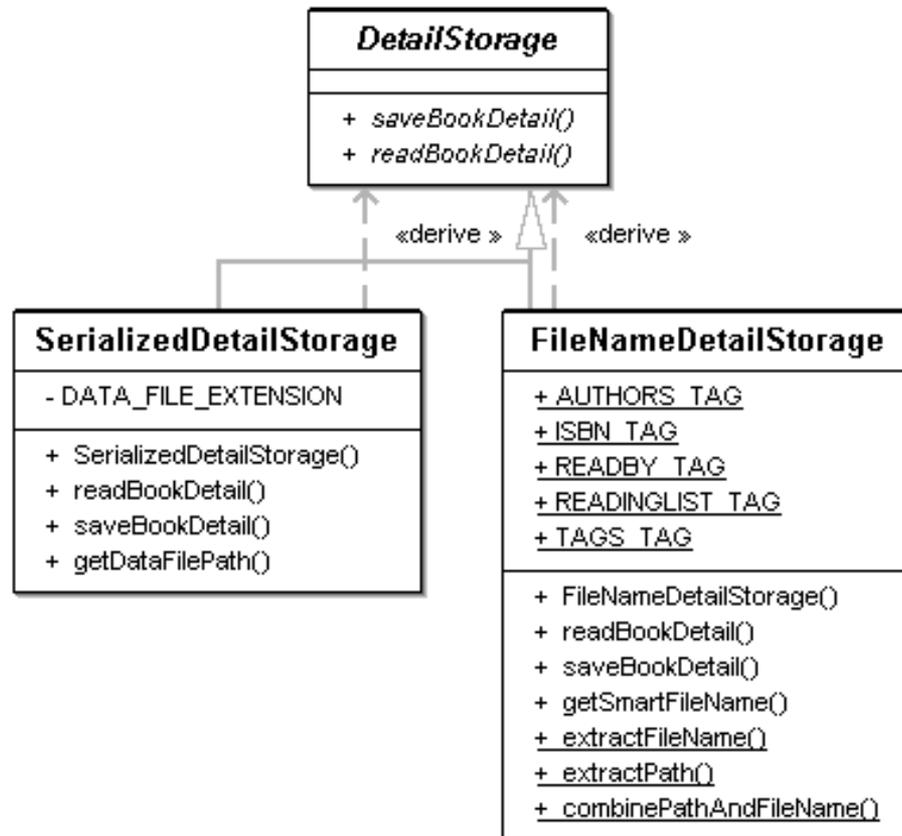
- Existing Concerns
  - Detail Storage
  - Checkout
  - Web Service
- New Concerns
  - Internationalization
  - Look&Feel

# Existing Concerns

## :Detail Storage

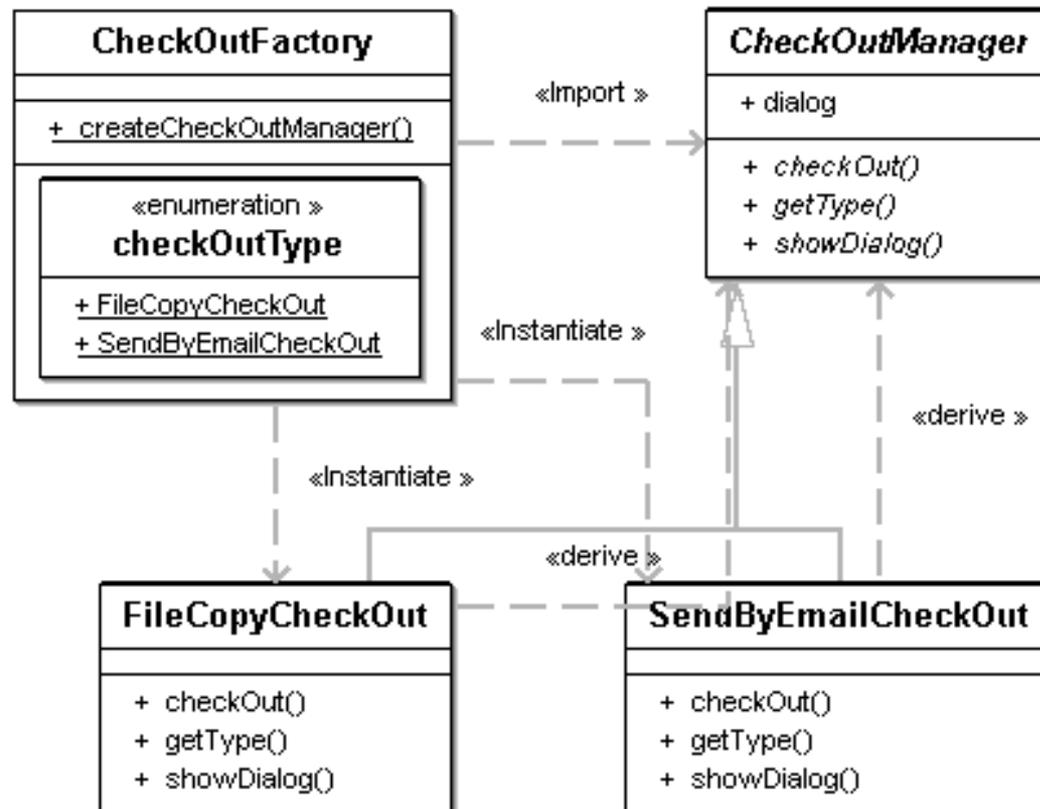
---

- By rearranging the Code



# Existing Concerns :Checkout

- By rearranging the Code





# New Concerns

---

- Internationalization
  - require many changes to be done on the existing code
- Look& Feel
- Agent Based Communication



## Our Strategies: Applying AOP After Object-Oriented Re-factoring

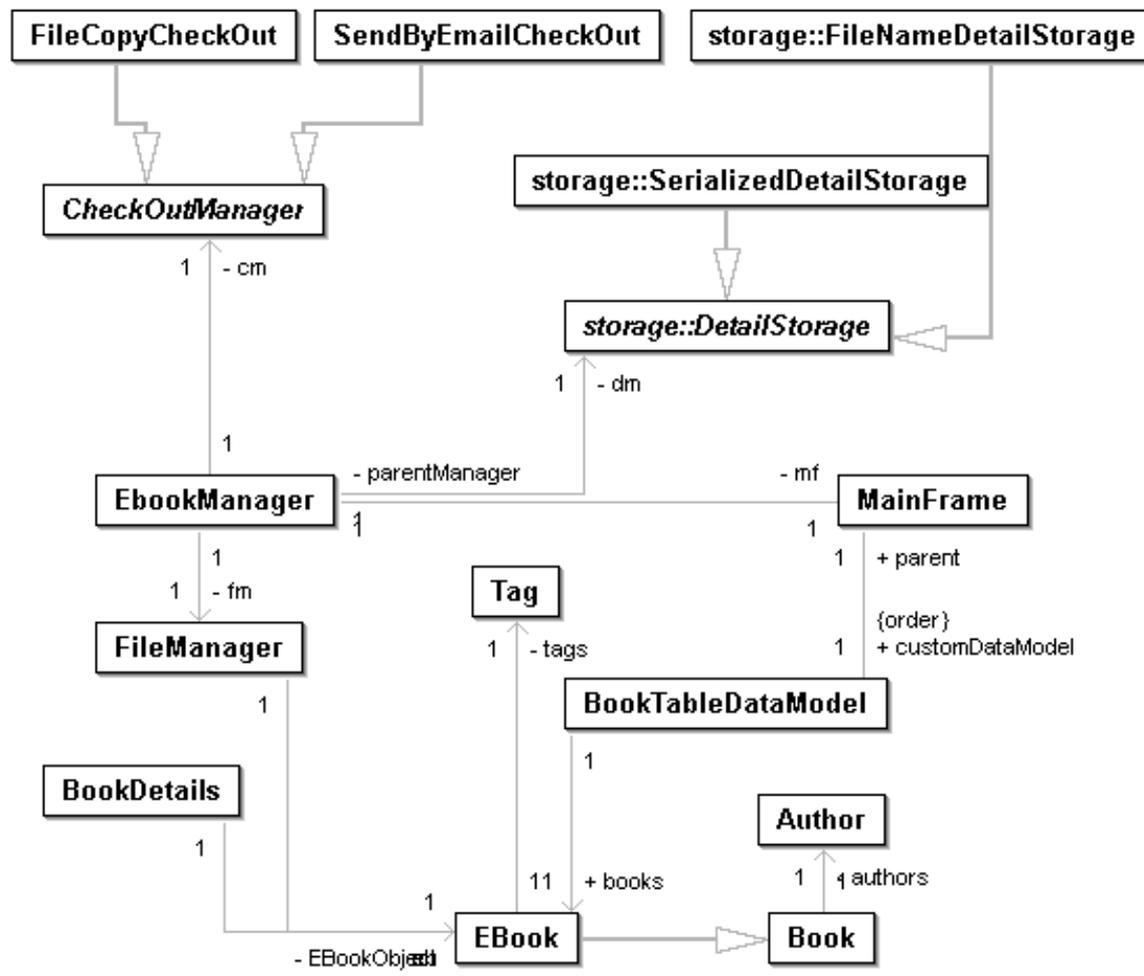
---

- Existing Concerns
  - Detail Storage
  - Checkout
  - Web Service
- New Concerns
  - Internationalization
  - Look&Feel

# Existing Concerns

## :Detail Storage

- By rearranging the Code





# Existing Concerns:Checkout

---

- Refactorization strategy
- Adding new functionality to menu



# Existing Concerns: Web Service

---

- Changed the code in place
- Could be done in a separate class



# Our Strategies: New Concerns

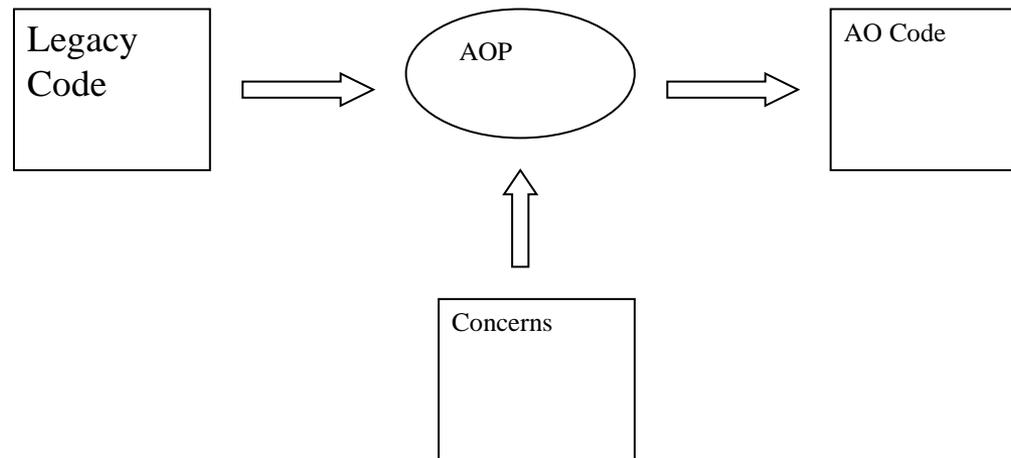
---

- Internationalization
- Look&Feel
- Agent Based Communication

# Discussion

---

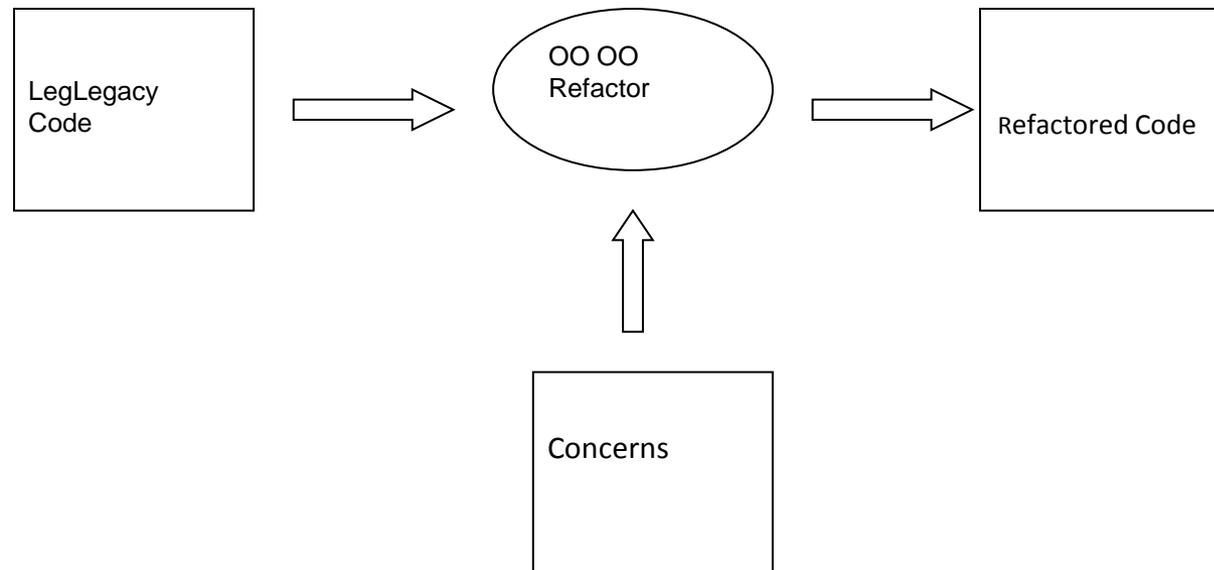
- Direct application of aspects to legacy code without re-factoring



# Discussion

---

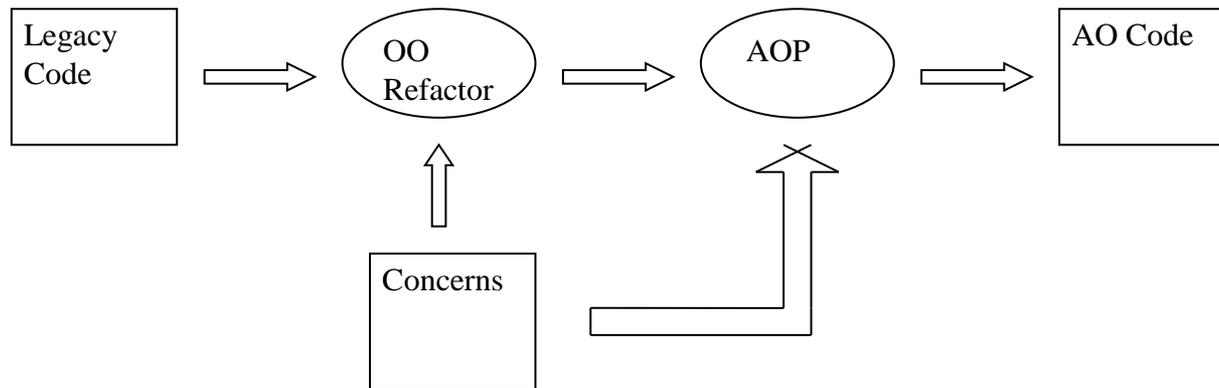
- Dealing crosscutting concerns by Object-Oriented re-factoring without AOP



# Discussion

---

## Applying AOP after Object-Oriented refactoring



# Discussion

	Existing Concerns	New Concerns
Directly Applying AOP	<ul style="list-style-type: none"> <li>• Easy to implement</li> <li>⌚ Does not require understanding all parts of code.</li> <li>⌚ No need to make change on existing code.</li> </ul>	<ul style="list-style-type: none"> <li>⌚ Easy to implement</li> <li>⌚ Do not require wide understanding on existing code.</li> <li>⌚ Could be applied to the project easily</li> <li>⌚ Improves extendibility for future aspectual concerns.</li> </ul>
Re-factorization	<ul style="list-style-type: none"> <li>⌚ Decreases crosscutting concerns</li> <li>⌚ In some cases may not be possible.</li> <li>⌚ Hard to understand existing scattered code</li> <li>⌚ Time consuming</li> <li>⌚ Increases code reuse-ability</li> </ul>	<ul style="list-style-type: none"> <li>⌚ For some concerns not possible to solve by re-factorization</li> <li>⌚ Requires hard coding within existing classes</li> </ul>
Re-factorization and AOP	<ul style="list-style-type: none"> <li>⌚ Could deal with crosscutting concerns which OOP re-factorization could not solve itself</li> <li>⌚ Improves modularity</li> </ul>	<ul style="list-style-type: none"> <li>⌚ Prevents crosscutting concerns</li> <li>⌚ Improves modularity</li> </ul>



# Demo

---

Thank You...

Questions are Welcomed...