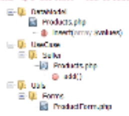
[illegible]

[illegible]

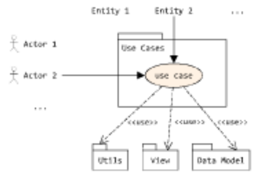
[illegible]

[illegible]

Traceability of use cases in source code

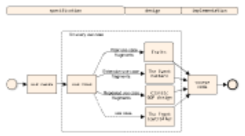


- DataModel
  - Products.php
    - insert(array $values)
- UseCase
  - Sells
    - Products.php
      - add()
- Utils
  - Forms
    - ProductForm.php



> Folder structure:

- DataModel
- ExtUseCase
- Libs
- UseCase
- Utils
- View



[illegible]

Summary

[illegible]

An Opportunistic Approach to Retaining Use Cases in Object-Oriented Source Code

Ján Greppel and Valentino Vranić

Institute of Informatics and Software Engineering

STU FIIT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

jangreppel@gmail.com

vranic@stuba.sk
fiit.sk/~vranic

ECBS-EERC 2015

Brno, August 27, 2015

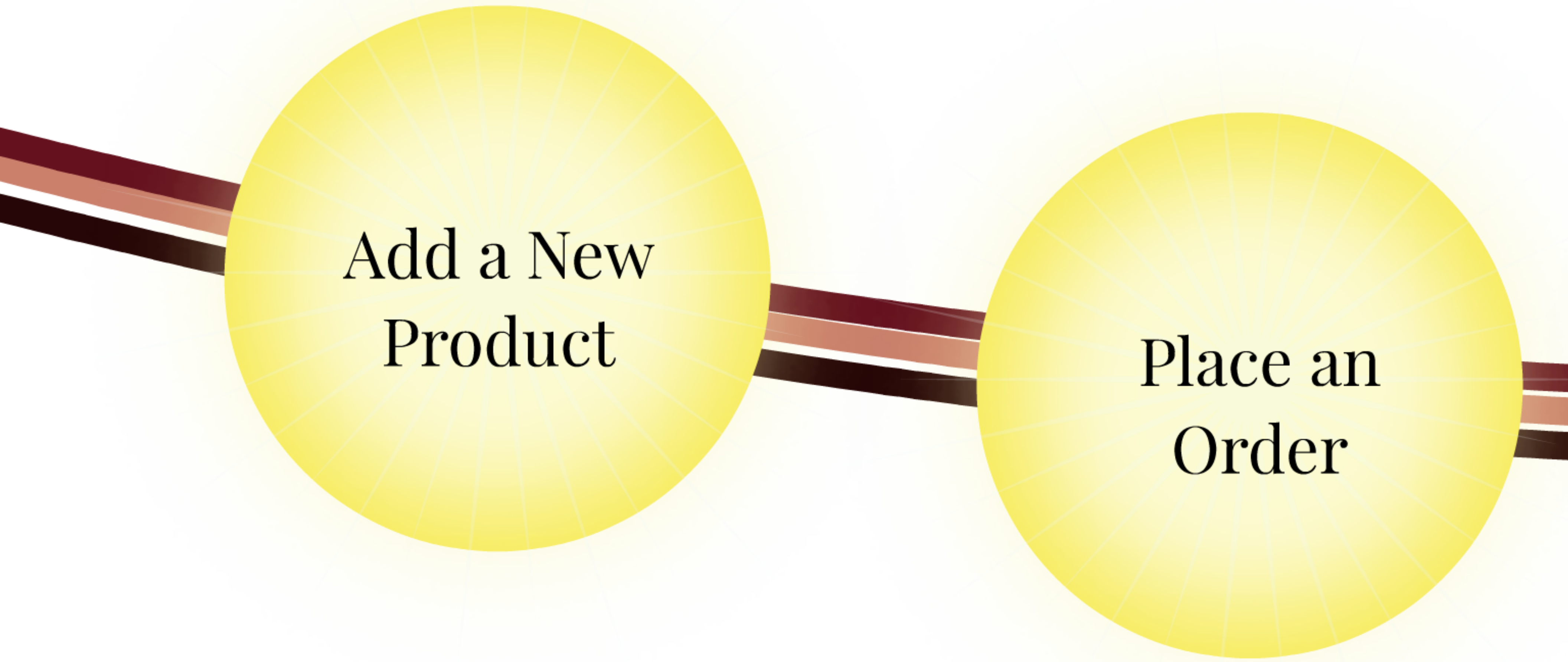
What is a use case and where is its place in the overall software system design?

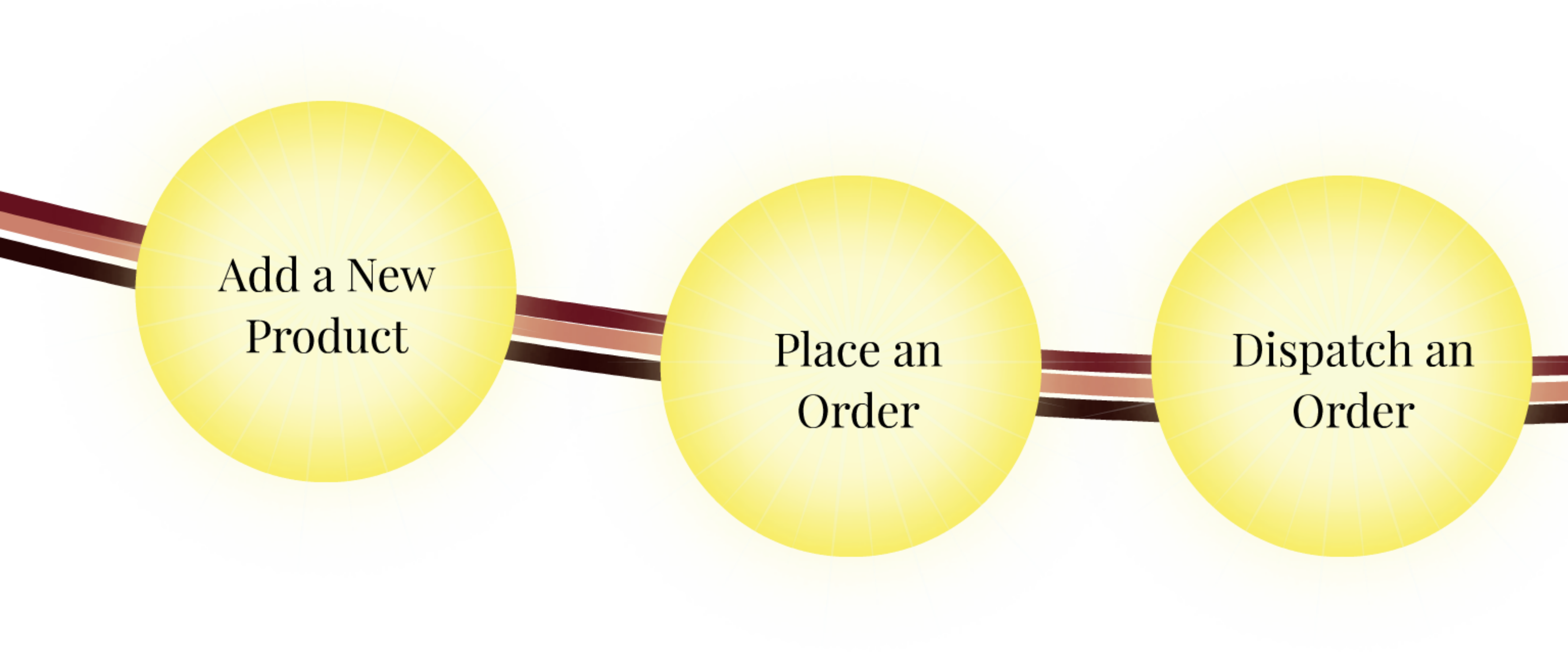
Add a New Product

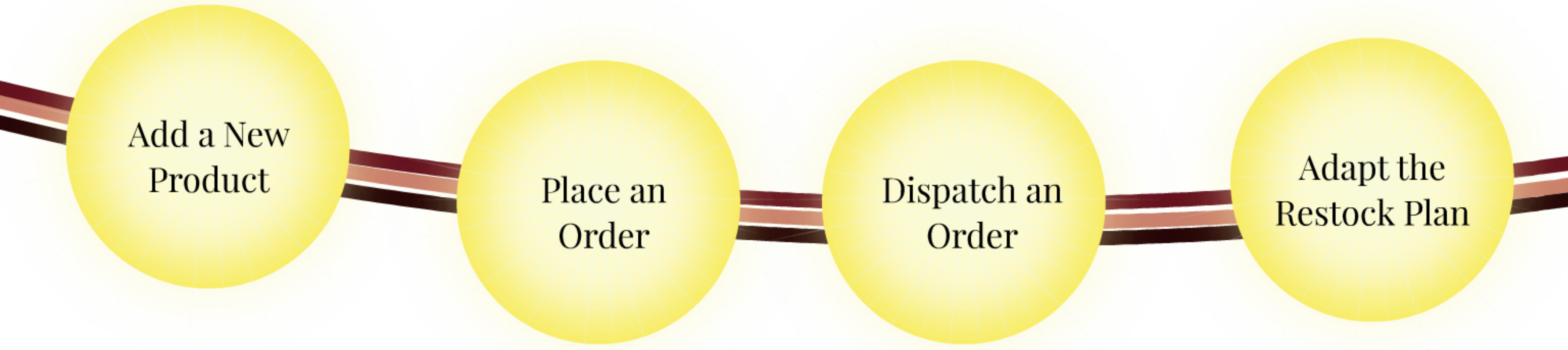
User: seller
Precondition: The user is logged in as a seller.

1. The user selects to add a new product.
2. The system prompts the user to fill the necessary information.
3. The user fills in the information and submits it.
4. The system:
    a) validates the information
    b) creates the new product
    c) notifies user about the creation of a new product
    d) shows the list of all products added by the current user
5. The use case ends.

Alternative scenario:
(if the filled in information is empty or in a wrong format)
4. The system
    a) validates the information
    b) displays the error message
    c) (step 3 again)


Add a New Product
Place an Order


Add a New Product
Place an Order
Dispatch an Order


Add a New Product
Place an Order
Dispatch an Order
Adapt the Restock Plan

> A use case as a bead of behavior on the string of the basic functionality and underlying data

What the system is

vs.

What the system does

> Use cases are a variable part of a software system: can be added or removed, but also can change

> The underlying structure may change, too, but far less frequently

> Use cases are comprehensible to all stakeholders, including the users

> But once translated into code, a use case model quickly becomes outdated

> A need to retain/preserve use cases in the code itself

> What can be retained out of a use case in code?

> Something is always retained, but some approaches aim explicitly at preserving use cases in code

> DCI (Data, Context and Interaction; Reenskaug and Coplien): a fairly complex approach that manages to isolate use cases into roles

> Aspect-oriented software development with use cases (Jacobson and Ng): requires aspect-oriented programming

> Preserving use case flows in source code (Bystrický and Vranić)

> What of a use case can be retained in OOP in an opportunistic manner?

> Common OOP preserves only use case fragments as methods and the include relationship as method call

> No direct support for the extend relationship and peer use cases

Add a New Product

User: seller
Precondition: The user is logged in as a seller.

1. The user selects to add a new product.
2. The system prompts the user to fill the necessary information.
3. The user fills in the information and submits it.
4. The system:
    a) validates the information
    b) creates the new product
    c) notifies user about the creation of a new product
    d) shows the list of all products added by the current user
5. The use case ends.

Alternative scenario:
(if the filled in information is empty or in a wrong format)
4. The system
    a) validates the information
    b) displays the error message
    c) (step 3 again)

Add a New Product

User: seller
Precondition: The user is logged in as a seller.

1. The user selects to add a new product.
2. The system prompts the user to fill the necessary information.
3. The user fills in the information and submits it.
4. The system:
    a) validates the information
    b) creates the new product
    c) notifies user about the creation of a new product
    d) shows the list of all products added by the current user
5. The use case ends.

Alternative scenario:
(if the filled in information is empty or in a wrong format)
4. The system
    a) validates the information
    b) displays the error message
    c) (step 3 again)

```
class Products {
  function add() {
    $form = new ProductForm();
    $form->setData($this->getPost());

    // Validate the information
    if ($form->isValid()) {
      // Create the new product
      ProductsDM::insert($this->getPost());

      // Notify the user about
      // the creation of a new product
      Messenger::getInstance()->
        addMessage('Product added');

      // Show the list of all products
      // added by the current user
      $this->dispatch('Products',
        'showListOfCurrentUser');
      return;
    }
    // Show the form (prompts the user
    // to fill the necessary information)
    $this->view = $form->render();
  }
  function showListOfCurrentUser() {
    // ...
  }
}
```

Add a New Product

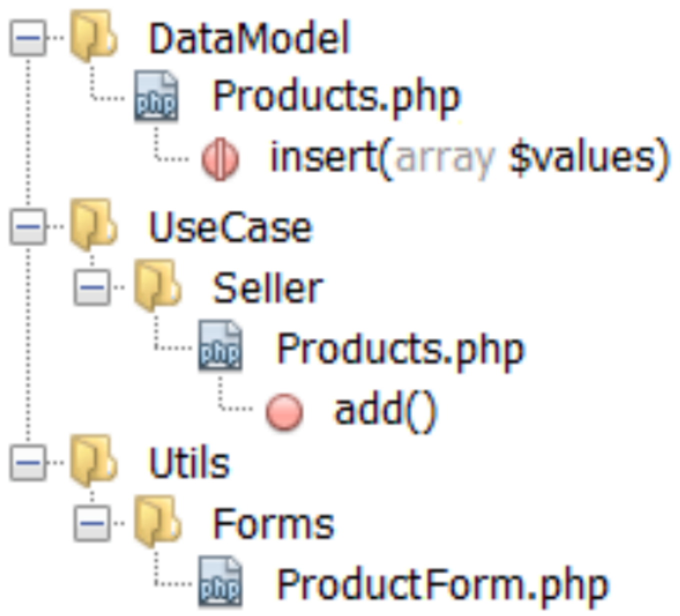
User: seller
Precondition: The user is logged in as a seller.

1. The user selects to add a new product.
2. The system prompts the user to fill the necessary information.
3. The user fills in the information and submits it.
4. The system:
   a) validates the information
   b) creates the new product
   c) notifies user about the creation of a new product
   d) shows the list of all products added by the current user
5. The use case ends.

Alternative scenario:
(if the filled in information is empty or in a wrong format)
4. The system
   a) validates the information
   b) displays the error message
   c) (step 3 again)

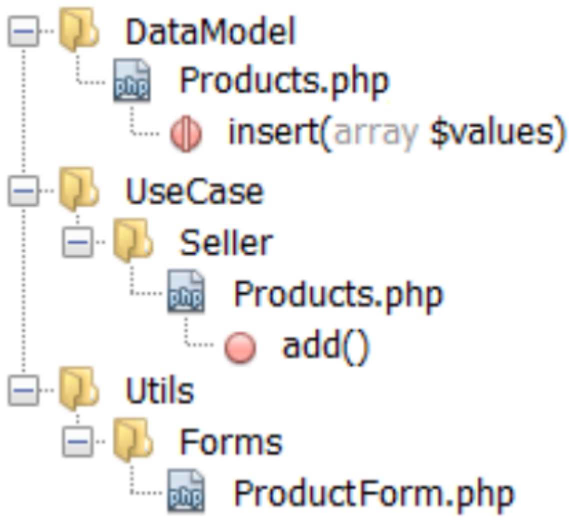
```
class Products {
  function add() {
    $form = new ProductForm();
    $form->setData($this->getPost());

    // Validate the information
    if ($form->isValid()) {
      // Create the new product
      ProductsDM::insert($this->getPost());

      // Notify the user about
      // the creation of a new product
      Messenger::getInstance()->
        addMessage('Product added');

      // Show the list of all products
      // added by the current user
      $this->dispatch('Products',
        'showListOfCurrentUser');
      return;
    }
    // Show the form (prompts the user
    // to fill the necessary information)
    $this->view = $form->render();
  }
  function showListOfCurrentUser() {
    // ...
  }
}
```

Traceability of use cases in source code

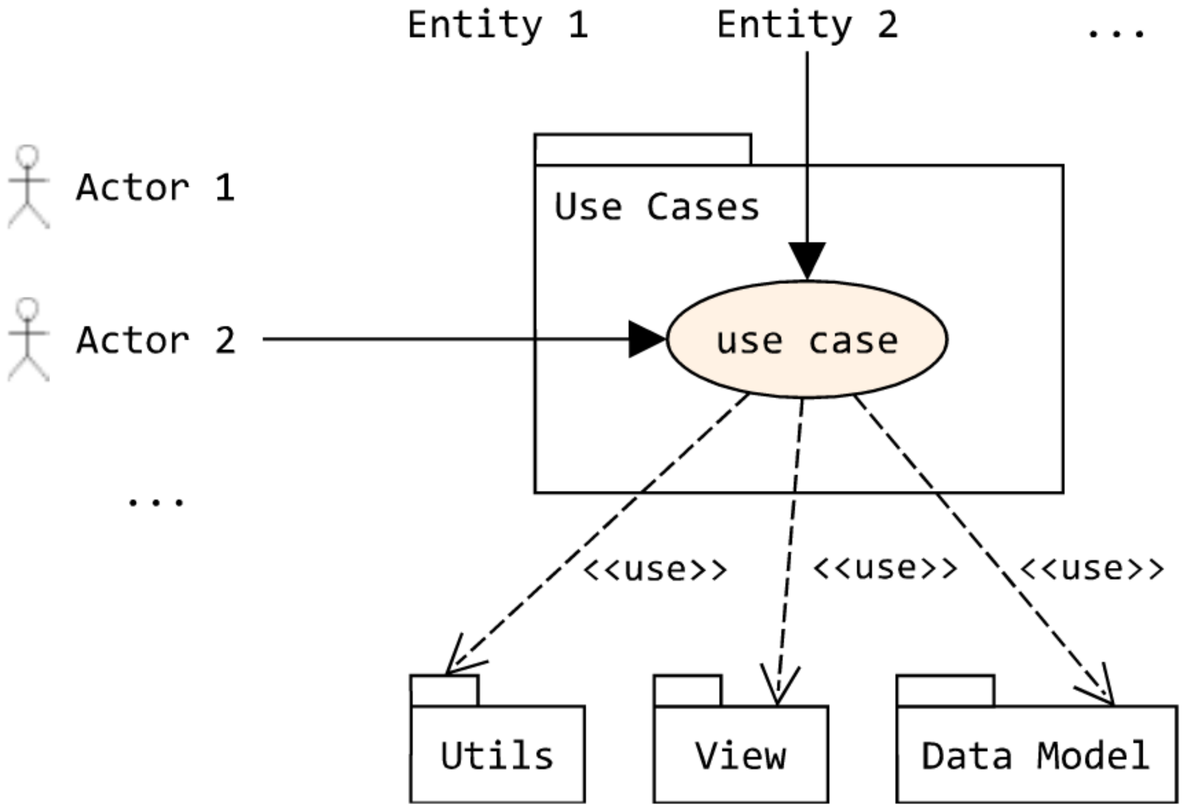


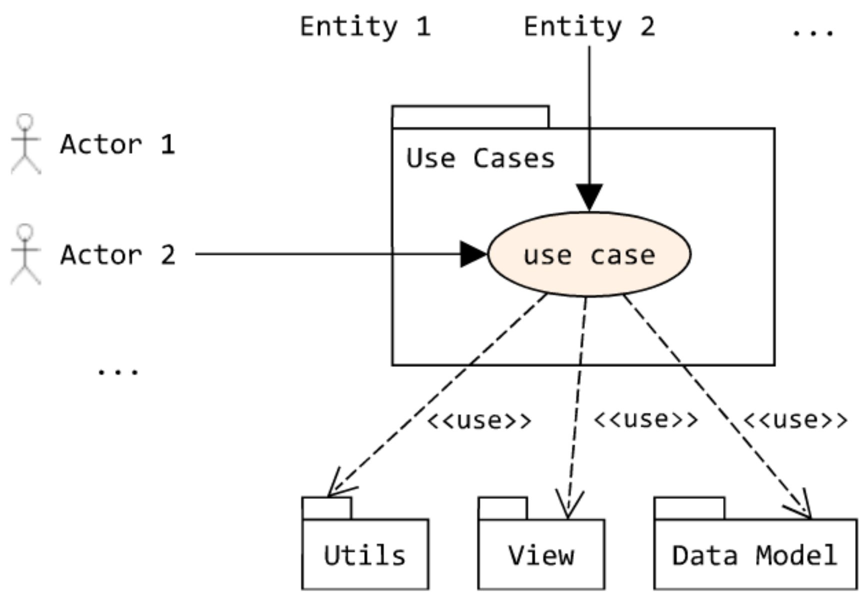
```
DataModel
  Products.php
    insert(array $values)
UseCase
  Seller
    Products.php
      add()
Utils
  Forms
    ProductForm.php
```

Traceability of use cases in source code



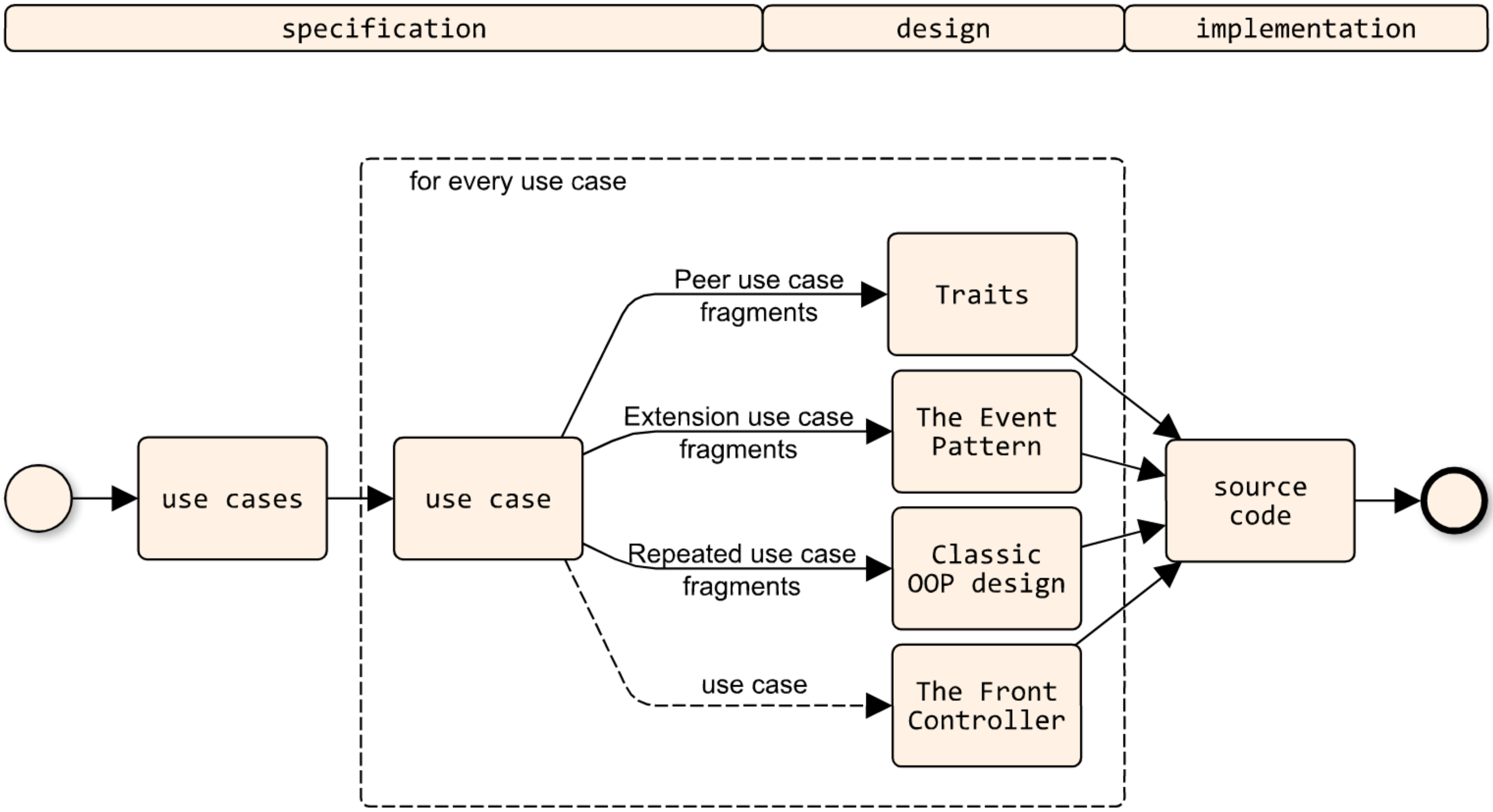
```
DataModel
    Products.php
        insert(array $values)
UseCase
    Seller
        Products.php
            add()
Utils
    Forms
        ProductForm.php
```





> Folder structure:

- DataModel
- ExtUseCase
- Libs
- UseCase
- Utils
- View


specification
design
implementation
for every use case
use cases
use case
Peer use case fragments
Extension use case fragments
Repeated use case fragments
use case
Traits
The Event Pattern
Classic OOP design
The Front Controller
source code

> Change requests are expressed in the application domain terms: the language of use cases

> With respect to use cases, any change request can be seen as a set of the following actions:

- Add a use case
- Remove a use case
- Alter a use case

> The evaluation of the approach has been performed qualitatively on the online shop application in terms of these actions

> The resulting changes to the code are well localized:

- Typically, only a few modules have to be changed
- In case of removal, modules are mostly removed as a whole

Summary

> An opportunistic approach to retaining use cases in source code by object-oriented means that employs:

- Traits
- The Event pattern
- The Front Controller pattern

> With only a moderate effort, use cases are quite easily located and manipulated in code

> The ability to discern different parts of the use case and implement it in appropriate places of source code is critical

> Targeting the client-server architecture and interactive enterprise systems

> Continuous refactoring efforts assumed