DCC / ICEx / UFMG

Implementation Techniques for Software Product Lines

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Implementation Techniques

- Conditional compilation
  - Antenna
- Aspect-oriented programming
  - AspectJ
- Feature-oriented programming
  - AHEAD

Conditional Compilation

Conditional Compilation

- It consists of annotating code fragments related to a specific feature
  - Such annotations are interpreted by a pre-processor
  - The pre-processor decides about the inclusion (or not) of the annotated code into the final product
- Examples of annotations
  - *#ifdef*, *#else*, *#endif*

Example in MobileMedia

```
public class PhotoListScreen extends List {

  public static final Command viewCommand;
  public static final Command addCommand;
  public static final Command deleteCommand;
  public static final Command editLabelCommand;

  // #ifdef includeSorting
  public static final Command sortCommand;
  // #endif

  // #ifdef includeFavourites
  public static final Command favoriteCommand;
  public static final Command viewFavoritesCommand;
  // #endif
...
}
```

Code of optional features are annotated with conditional compilation (#ifdef)

Features: Sorting and Favourites

```
public class PhotoListScreen extends List {

  public static final Command viewCommand;
  public static final Command addCommand;
  public static final Command deleteCommand;
  public static final Command editLabelCommand;
  // #ifdef includeSorting
S public static final Command sortCommand;
  // #endif
  // #ifdef includeFavourites
F public static final Command favoriteCommand;
  public static final Command viewFavoritesCommand;
  // #endif
...
}
```

Code Annotated in a Method

```
public class PhotoListScreen extends List {
  ...
  public void initMenu() {
    this.addCommand(viewCommand);
    this.addCommand(addCommand);
    this.addCommand(deleteCommand);
    this.addCommand(editLabelCommand);

    // #ifdef includeSorting
    this.addCommand(sortCommand);
    // #endif

    // #ifdef includeFavourites
    this.addCommand(favoriteCommand);
    this.addCommand(viewFavoritesCommand);
    // #endif
  }
}
```

Annotated code can be inside a method or anywhere in a class.

Features: Sorting and Favourites

```
public class PhotoListScreen extends List {
  ...
  public void initMenu() {
    this.addCommand(viewCommand);
    this.addCommand(addCommand);
    this.addCommand(deleteCommand);
    this.addCommand(editLabelCommand);
```

S

```
    // #ifdef includeSorting
    this.addCommand(sortCommand);
    // #endif
```

F

```
    // #ifdef includeFavourites
    this.addCommand(favoriteCommand);
    this.addCommand(viewFavoritesCommand);
    // #endif
  }
}
```

Product Configuration

- Let’s suppose MobileMedia has only
  - One mandatory feature (Core)
  - Two optional features (Sorting and Favourites)

Configurations	**Core**	**Sorting**	**Favourites**
Product 1	Yes	Yes	Yes
Product 2	Yes	Yes	No
Product 3	Yes	No	Yes
Product 4	Yes	No	No

How to configure a product

- There are several ways
  - If you use Antenna, you have to tell the pre-processor the features to be included in a product

S F preprocessor.symbols = core, includeSorting, includeFavourites

S preprocessor.symbols = core, includeSorting

F preprocessor.symbols = core, includeFavourites

preprocessor.symbols = core

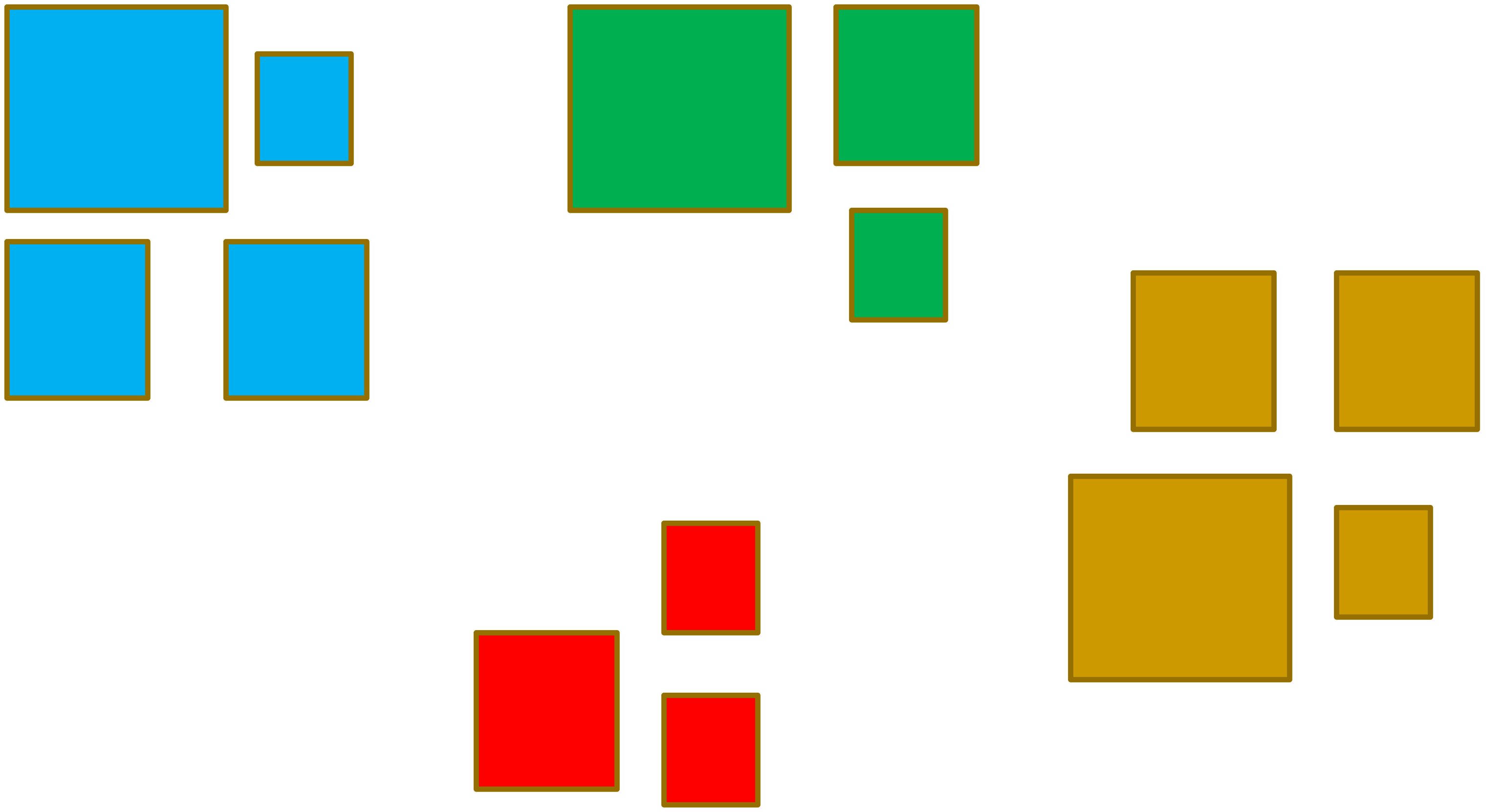
Aspect-Oriented Programming (AOP) for SPL

AspectJ

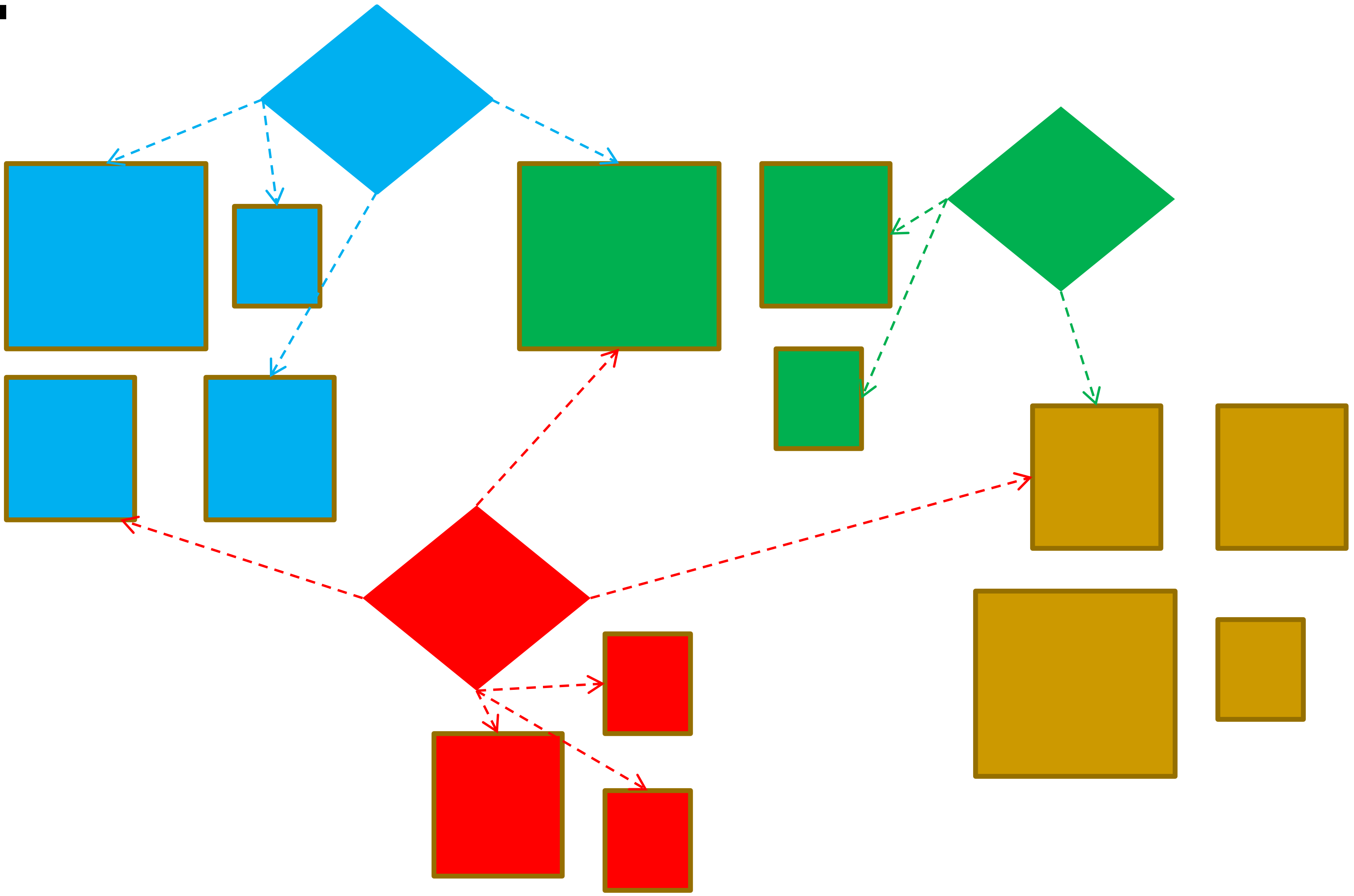
AOP for SPL

- Aspects can be used for
  - Modularizing crosscutting features
  - Composing features into a product
- Each optional feature can be modularly implemented as a set of classes and aspects
  - To extract the feature code to an aspect, we need to first identify such code, a process called feature location

Modularized Features

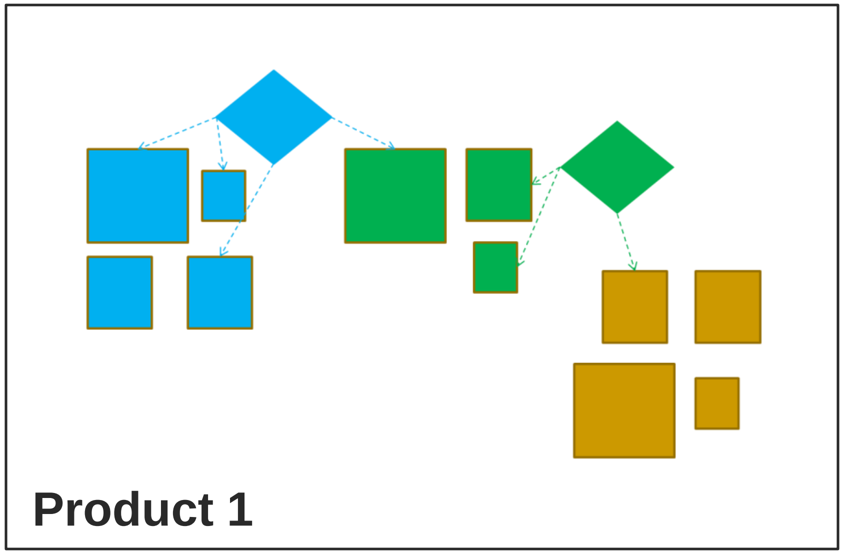


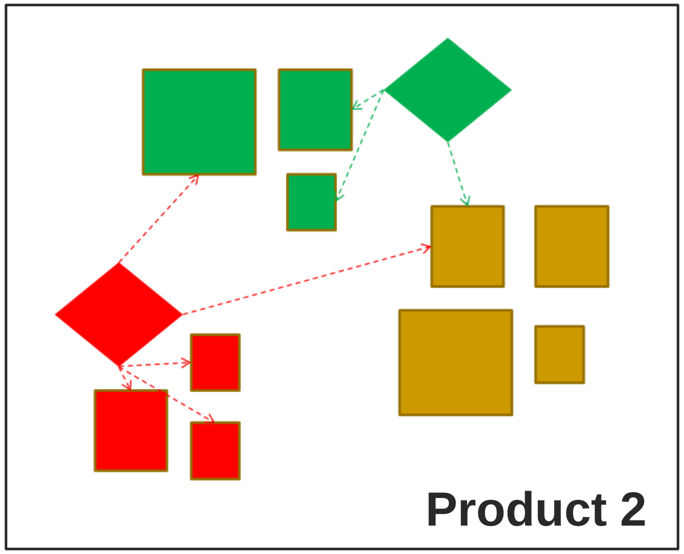
Aspects to Connect Features



Product Configuration

- To configure a product, you include classes and aspects which implement the aimed features





Example of Code in MobileMedia

```
public class PhotoListScreen extends List {
  public static final Command viewCommand;
  public static final Command addCommand;
  public static final Command deleteCommand;
  public static final Command editLabelCommand;
...
}
```

S

```
public aspect SortingAspect {
  public static final Command sortCommand;
  pointcut initMenu(PhotoListScreen screen):
    execution(public void PhotoListScreen.initMenu()) && this(screen);
  after(PhotoListScreen screen) : initMenu(screen) {
    screen.addCommand(sortCommand);
  }
...
}
```

Feature Oriented Programming (FOP)

AHEAD

Feature-Oriented Programming

- Feature Oriented Programming is a technique for developing software product lines
- A feature is a functional increment in software development

Successive Refinements

- The base code is successively refined aiming for a later composition
    - Each feature is a refinement of the base code
- FOP focuses on simplicity and understandability of each refinement (feature)

Example of Code in MobileMedia

```
public class PhotoListScreen extends List {
  public static final Command backCommand = new Command(…);
  …
  public void initMenu() {
    this.addCommand(backCommand);
  }
}
```

F

```
public refines class PhotoListScreen {
  public static final Command viewFavoritesCommand = new Command(…);
  public void initMenu() {
    Super().initMenu();
    this.addCommand(viewFavoritesCommand);
  }
}
```

S

```
public refines class PhotoListScreen {
  public static final Command sortCommand = new Command(…);
  public void initMenu() {
    Super().initMenu();
    this.addCommand(sortCommand);
  }
}
```

…

Bibliography

- E. Figueiredo, *et al*. **Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability**. International Conference on Software Engineering (ICSE), 2008. (*CC and AOP*)
- G. Ferreira, F. Gaia, E. Figueiredo e M. Maia. **On the Use of Feature-Oriented Programming for Evolving Software Product Lines - A Comparative Study**. Simpósio Brasileiro de Linguagens de Programação (SBLP), 2011. (*AOP and FOP*)