

# Software product lines

Paulo Borba

Informatics Center

Federal University of Pernambuco

# Variability implementation mechanisms

Paulo Borba

Informatics Center

Federal University of Pernambuco

What would you do to  
implement slightly  
different services to  
different products  
from the same family?

# Conditional execution

```
canvas_midp2 = Boolean.valueOf(args[1]);  
canvas_siemens = Boolean.valueOf(args[2]);  
if (canvas_midp2 || canvas_siemens) {  
    this.mainCanvas.paint();  
    this.mainCanvas.flushGraphics();  
} else {  
    this.mainCanvas.repaint();  
    this.mainCanvas.serviceRepaints();  
}
```

# Conditional compilation

```
//#if canvas_midp2 || canvas_siemens
    this.mainCanvas.paint();
    this.mainCanvas.flushGraphics();
//#else
//#  this.mainCanvas.repaint();
//#  this.mainCanvas.serviceRepaints();
//#endif
```

From Meantime Mobile Creations

# Pre-processing followed by compilation

## Antenna file

```
//#if canvas_midp2 || canvas_siemens  
//#  this.mainCanvas.paint();  
//#  this.mainCanvas.flushGraphics();  
//#else  
//#  this.mainCanvas.repaint();  
//#  this.mainCanvas.serviceRepaints();  
//#endif
```

## Tags file

```
canvas_midp2,  
device_screen_128x128,  
...
```



## product

```
this.mainCanvas.paint();  
this.mainCanvas.flushGraphics();
```

# CC Core assets

## Antenna file

```
//#if canvas_midp2 || canvas_siemens
//#  this.mainCanvas.paint();
//#  this.mainCanvas.flushGraphics();
#else
//#  this.mainCanvas.repaint();
//#  this.mainCanvas.serviceRepaints();
#endif
```



Makefiles

## Tags files

```
canvas_midp2,
device_screen_128x128,
...
```

```
canvas_nokiaui,
device_screen_128x128,
...
```

# Inside commands...

```
if (display.getCurrent()==this.mainCanvas)
{
//#if canvas_midp2 || canvas_siemens
//#  this.mainCanvas.paint();
//#  this.mainCanvas.flushGraphics();
//#else
//#  this.mainCanvas.repaint();
//#  this.mainCanvas.serviceRepaints();
//#endif
}
```

From Meantime Mobile Creations



# and everywhere

```
///#if canvas_nokiaui
///# public class MainCanvas extends FullCanvas {
///#elif canvas_midp2 || canvas_siemens
///# public class MainCanvas extends GameCanvas {
///#else
///# public class MainCanvas extends Canvas {
///#endif
private static int keyStatesNotRepeatable;
///#if device_screen_128x128
///# public static final int SCREEN_WIDTH = 128;
///# public static final int SCREEN_HEIGHT = 128;
///#elif device_screen_128x117
///# public static final int SCREEN_WIDTH = 128;
...

```

# Analyzing conditional compilation

- Variation point (**site**)
  - any point in the program text
- Variation
  - any program element
- Binding time
  - precompilation time, based on makefile (build time)
- Mechanism
  - decision based on tags file, specified by makefile

# Parametrization via property files (PFP)

```
Properties p = new Properties();  
p.load(new  
    FileInputStream("device_screen.prop"));  
...  
    p.getProperty("SCREEN_WIDTH")  
...
```

device\_screen.prop

```
SCREEN_WIDTH=128  
SCREEN_HEIGHT=128
```

# Parametrization via property files, with reflection

```
Properties p = new Properties();  
p.load(new  
    FileInputStream("repositories.prop"));  
...  
c = Class.forName(p.getProperty("ACCREP"));  
accrep = (Interface) c.newInstance();  
...
```

```
ACCREP=RelationalRepository
```

# PFP core assets

## Java file

```
Properties p = new Properties();  
p.load(new  
    FileInputStream("device_screen.prop"));  
...  
    p.getProperty("SCREEN_WIDTH")  
...
```



Makefiles

## Property files

device\_screen.prop

```
SCREEN_WIDTH=128  
SCREEN_HEIGHT=128
```

device\_screen.prop

```
SCREEN_WIDTH=128  
SCREEN_HEIGHT=117
```

# Properties provide data and behavior too

```
Properties p = new Properties();
p.load(new
    FileInputStream("device_screen.prop"));
...
if (p.getProperty("SCREEN_WIDTH") == 128) {
    ...
} else {...}
...
```

# Analyzing PFP

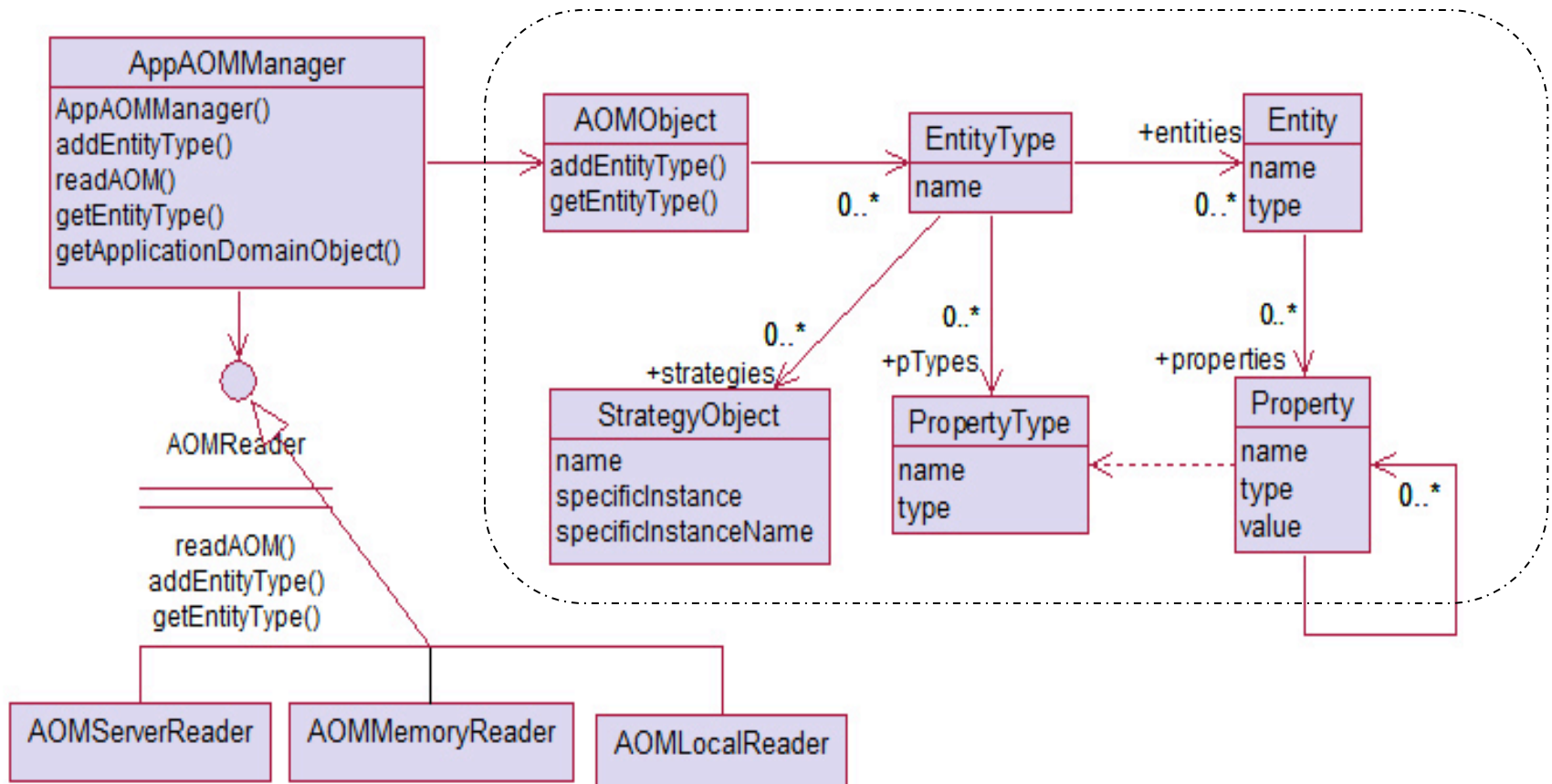
- Variation point
  - expressions in the program text, and commands too (with conditionals)
- Variation
  - constants and values, commands
- Binding time
  - execution time, based on makefile choice (build time)
- Mechanism
  - value instantiation, decision

# Adaptive object models

- Several OO patterns
- But not only subtype polymorphism
- Metadata plays an essential role
  - Supports flexible dynamic **adaptations**
  - Represents business rules and entities as data
  - POS performs a sequence of actions; the actions and sequence are represented as objects



# Adaptive object-model



# Parametrization via parametric polymorphism (PPP)

```
public class Record<ComplaintType> {  
    ...  
    public void insert(ComplaintType o) {  
        ...  
    }  
}
```

```
    ...  
    Record<HealthComplaint>  
    ...
```

# PPP core assets

## Java file, generic

```
public class Record<
    ComplaintType extends CodeIdentifiable> {
    ...
    public ComplaintType search(int code){
        ...
    }
    public void insert(ComplaintType o) {
        ...
    }
}
```



Makefiles

## Java files, specific

```
...
Record<HealthComplaint>
...
```

```
...
Record<SafetyComplaint>
...
```

# Parametrizing data + operations

```
public class Record<
    ComplaintType extends CodeIdentifiable> {
    ...
    public ComplaintType search(int code) {
        ...
    }
    public void insert(ComplaintType o) {
        ...
    }
}

interface CodeIdentifiable {
    int getCode();
}
```

# Analyzing PPP

- Variation point
  - class parameters (types and methods) in the program text
- Variation
  - types (including methods)
- Binding time
  - compilation time, based on makefile choice
- Mechanism
  - type instantiation, variation and “common” code modularization

# Mixing mechanisms: PPP, but instantiation with PFP and conditional

- Variation point
  - class parameters (types and methods) in the program text
- Variation
  - types (including methods)
- Binding time
  - **execution** time, based on makefile choice
- Mechanism
  - type and **value** instantiation, variation modularization, **decision (conditional)**

# Mixing mechanisms: PPP, with conditional and input parameter

- Variation point
  - class parameters (types and methods) in the program text
- Variation
  - types (including methods)
- Binding time
  - **execution** time, based on **user** choice
- Mechanism
  - type and value instantiation, variation modularization, **decision (conditional)**

Same modularization  
mechanism, 3 decision  
mechanisms, 2 instantiation  
mechanisms:

- Makefiles
- Makefiles + property\_files + conditionals
- Input + conditionals



# Variability implementation mechanisms

Paulo Borba

Informatics Center

Federal University of Pernambuco