

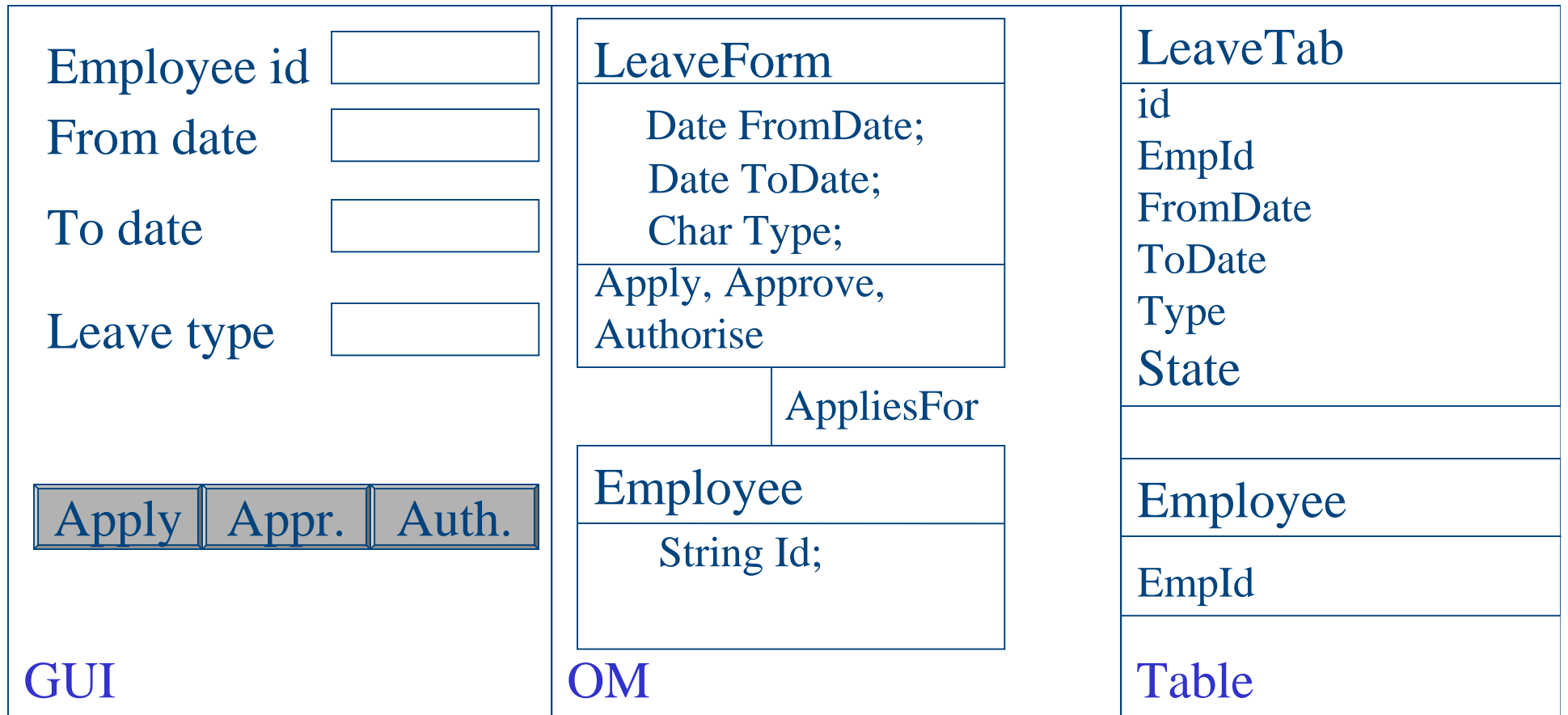
# Heavyweight Extension of UML for GUI Modeling

A Template Based Approach

# GUI Development of Large Systems

- ◆ A team of programmers develop screens as part of prototype
  - ◆ Based on standards
  - ◆ Based on an initial understanding of functionality
- ◆ This is later extended with code

# Independent Development



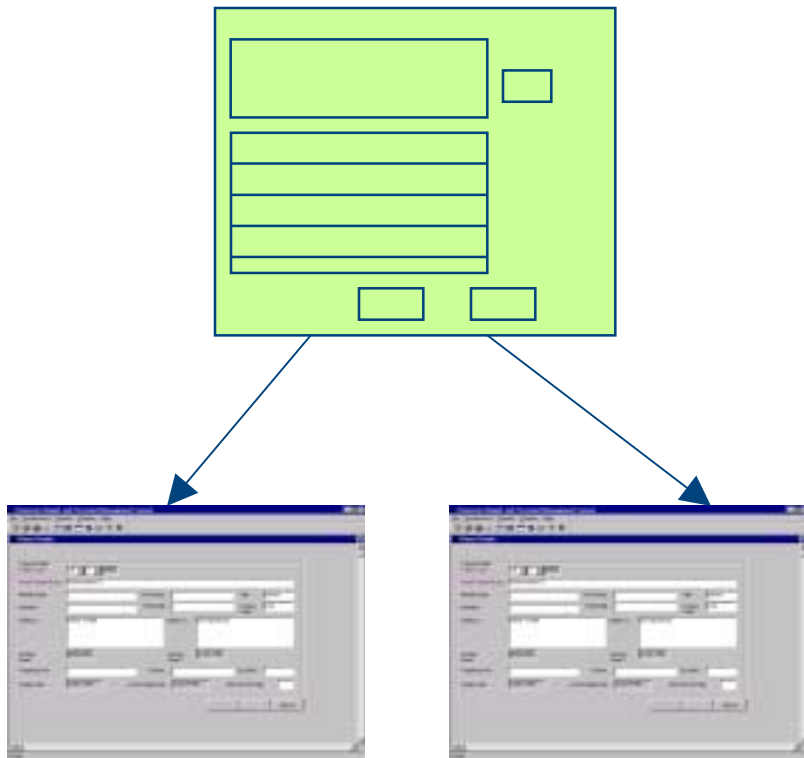
# Traditional Approach - Problems

- ◆ Unstructured
  - ◆ Difficult to ensure adherence to standards
  - ◆ Feasibility of implementing features
  - ◆ Little or no reuse
- ◆ No relationship between models
  - ◆ Difficult to guarantee consistency across layers

Solution - Modeling

---

# GUI Modeling as Templates



## Modeling

- ◆ Relationship between UI and Application
- ◆ UI standards

## Template

- ◆ Layout
- ◆ Behaviour

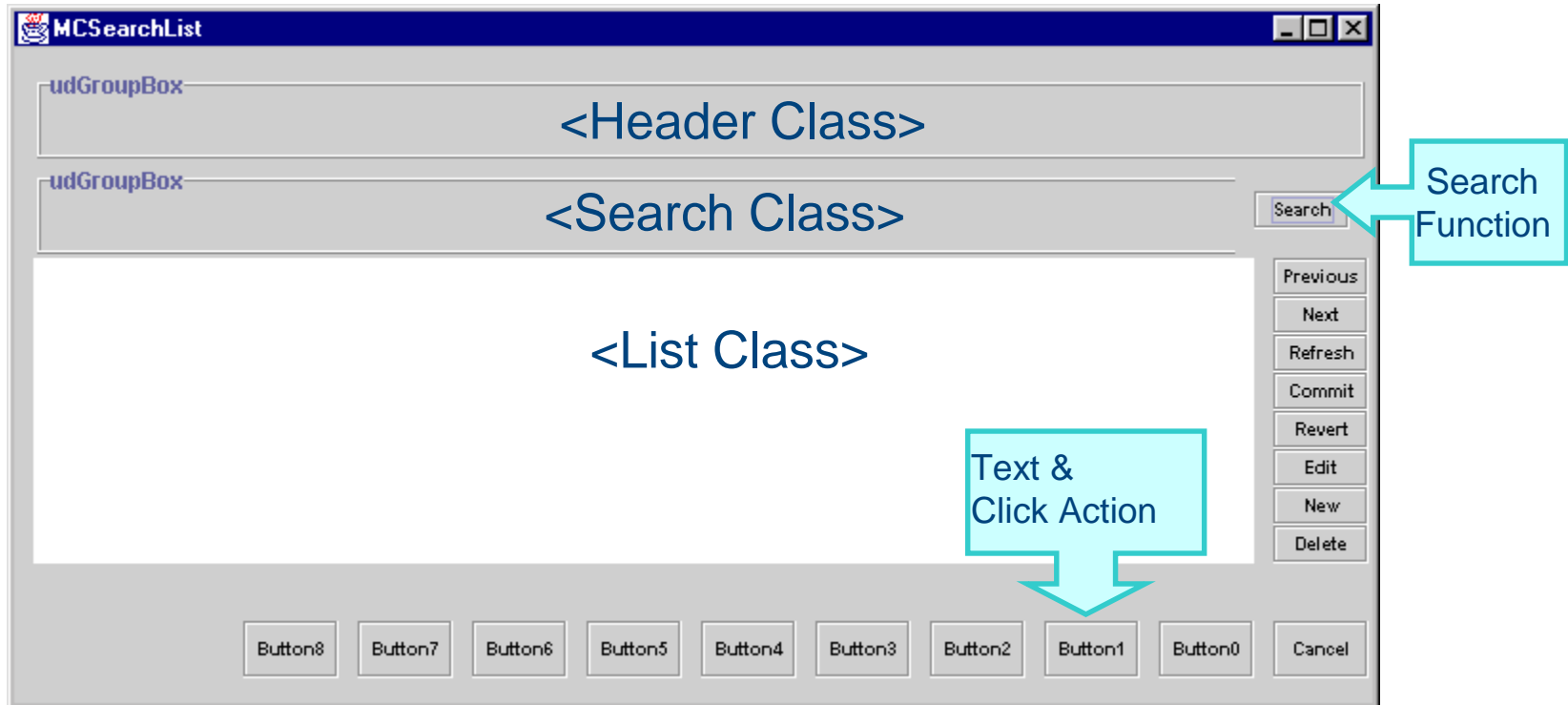
# GUI - Templates

- ◆ WindowTypes - parameterized templates
  - ◆ groups
  - ◆ functions
  - ◆ child windows
  - ◆ buttons

Includes interaction behavior as well as layout

- ◆ Each window is an instance of a WindowType

# List Type



# List Window

**Payee Class List**

Payee Class Details

**Payee Class Id**  **Description**

**Oblig Type Code**  **Area Code**

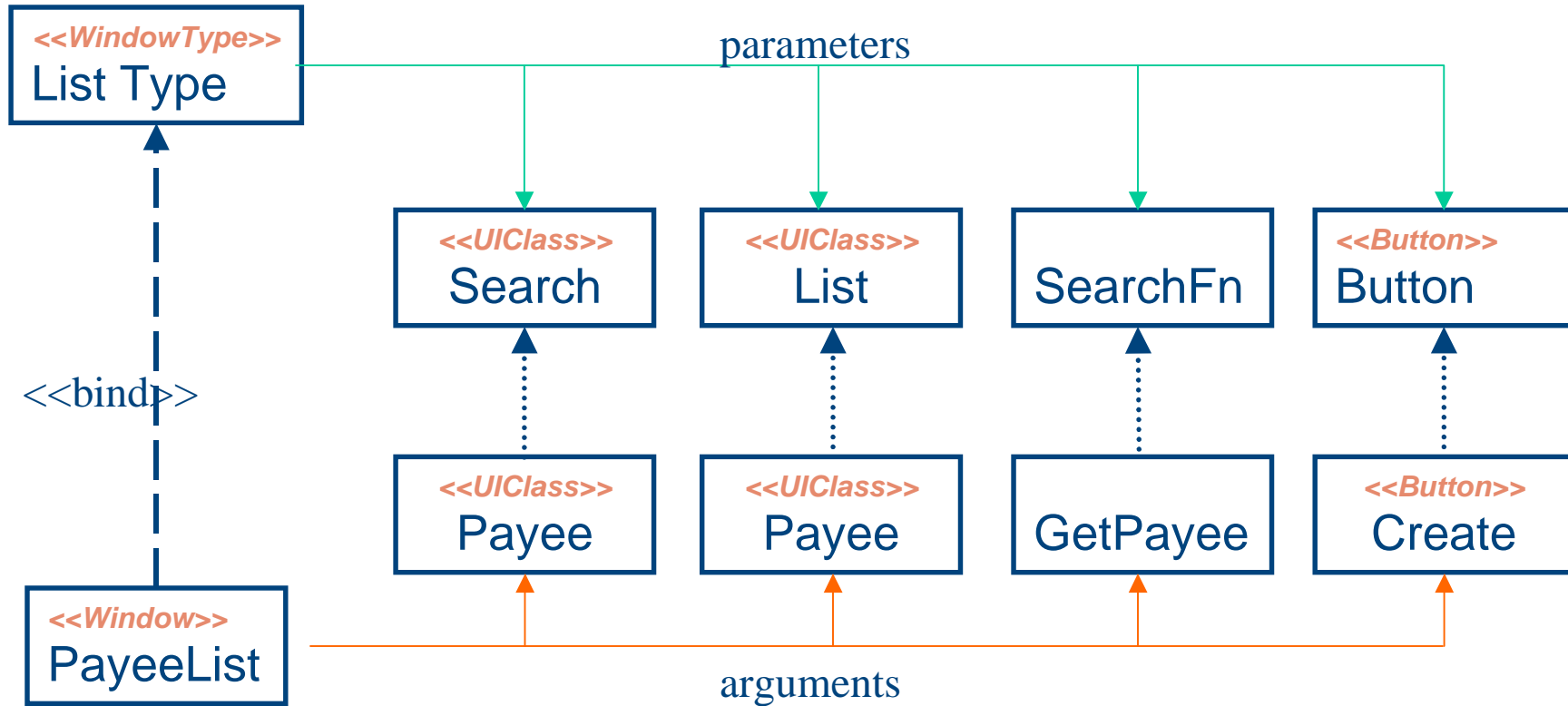
Search

| <b>Payee Class Id</b> | <b>Description</b> | <b>Oblig Type Code</b> | <b>Area Code</b> |
|-----------------------|--------------------|------------------------|------------------|
|                       |                    |                        |                  |

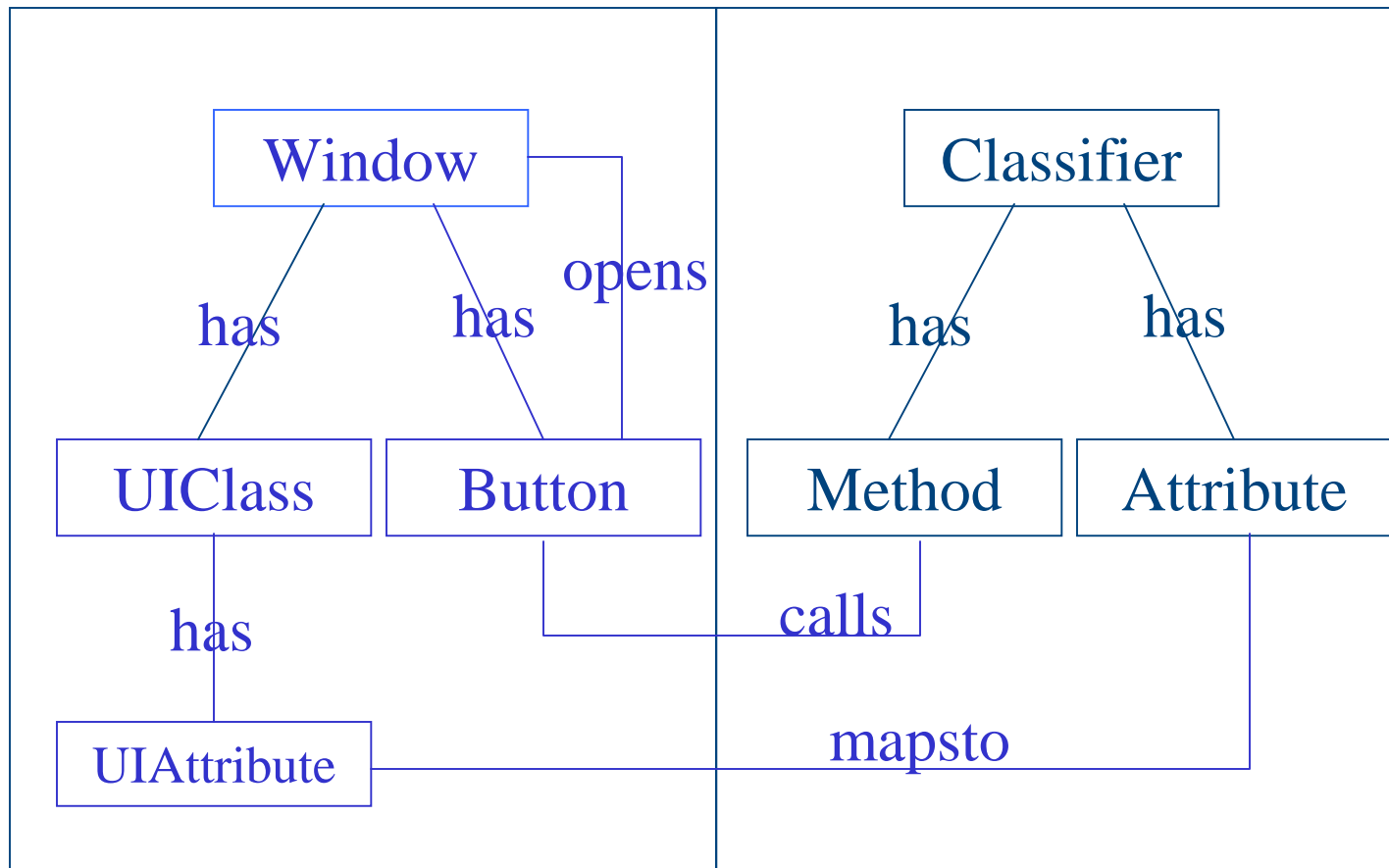
Previous  
Next  
Refresh  
Commit  
Revert  
Edit  
New  
Delete

Delete Modify Create Cancel

# WindowType Model



# Models - Relations



# Standards

- ◆ DataType - control mapping
  - ◆ date - calendar control
  - ◆ appropriate dropdowns
- ◆ DataType - label mapping
- ◆ Font, size, colour ....

# Stereotypes

| Stereotype      | Base Class | Description  |
|-----------------|------------|--|
| WindowType      | Class      | The GUI template which takes UIClasses and Button as parameter |
| Window          | Class      | Instance of WindowType, actual window of the application       |
| UIClass         | Class      | Contains the fields to be shown to the application user        |
| MapsToControl   | Dependency | The control to be used to display a particular datatype        |
| MapsToAttribute | Dependency | Sets up a mapping between UIClass and Class                    |
| Button          | Class      | Buttons on a screen that may open a window or invoke a method  |

# GUIMOD

UIModel  
Templates + Instances  
+  
Application Model  
+  
Template Code



Code  
in  
Java  
or  
JSP  
or  
Winform

# Summary

- ◆ UI modeling advantages
  - ◆ better integration - ui model and domain model
  - ◆ better standardisation
  - ◆ improved productivity
  
- ◆ Issues
  - ◆ stereotypes v/s meta-modeling

Thank You!