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**

<table>
<thead>
<tr>
<th>Employee id</th>
<th>From date</th>
<th>To date</th>
<th>Leave type</th>
<th>Apply</th>
<th>Appr.</th>
<th>Auth.</th>
</tr>
</thead>
</table>

**LeaveForm**
- Date FromDate;
- Date ToDate;
- Char Type;
- Apply, Approve, Authorise

**ApplyFor**

<table>
<thead>
<tr>
<th>Employee</th>
<th>String Id;</th>
</tr>
</thead>
</table>

**LeaveTab**
- id
- EmpId
- FromDate
- ToDate
- Type
- State

**Employee**
- EmpId

**Table**
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

<Header Class>

<Search Class>

<List Class>

Search Function

Text & Click Action
List Window

Payee Class List

Payee Class Details
- Payee Class Id: 12
- Oblg Type Code
- Description
- Area Code

Payee Class List
<table>
<thead>
<tr>
<th>Payee Class Id</th>
<th>Description</th>
<th>Oblg Type Code</th>
<th>Area Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Buttons: Previous, Next, Refresh, Commit, Revert, Edit, New, Delete, Delete, Modify, Create, Cancel
Models - Relations

Window

has

hasOneClass

Button

has

UIButtonAttribute

opens

Classifier

has

Method

has

Attribute

calls

mapsto

TATA RESEARCH DEVELOPMENT AND DESIGN CENTRE
Standards

♦ **DataType - control mapping**
  - date - calendar control
  - appropriate dropdowns

♦ **DataType - label mapping**

♦ Font, size, colour ....
<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Base Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WindowType</td>
<td>Class</td>
<td>The GUI template which takes UIClasses and Button as parameter</td>
</tr>
<tr>
<td>Window</td>
<td>Class</td>
<td>Instance of WindowType, actual window of the application</td>
</tr>
<tr>
<td>UIClass</td>
<td>Class</td>
<td>Contains the fields to be shown to the application user</td>
</tr>
<tr>
<td>MapsToControl</td>
<td>Dependency</td>
<td>The control to be used to display a particular datatype</td>
</tr>
<tr>
<td>MapsToAttribute</td>
<td>Dependency</td>
<td>Sets up a mapping between UIClass and Class</td>
</tr>
<tr>
<td>Button</td>
<td>Class</td>
<td>Buttons on a screen that may open a window or invoke a method</td>
</tr>
</tbody>
</table>
GUIMOD

UIModel
Templates + Instances
+ Application Model
+ Template Code

Model Based GUI Generator

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!