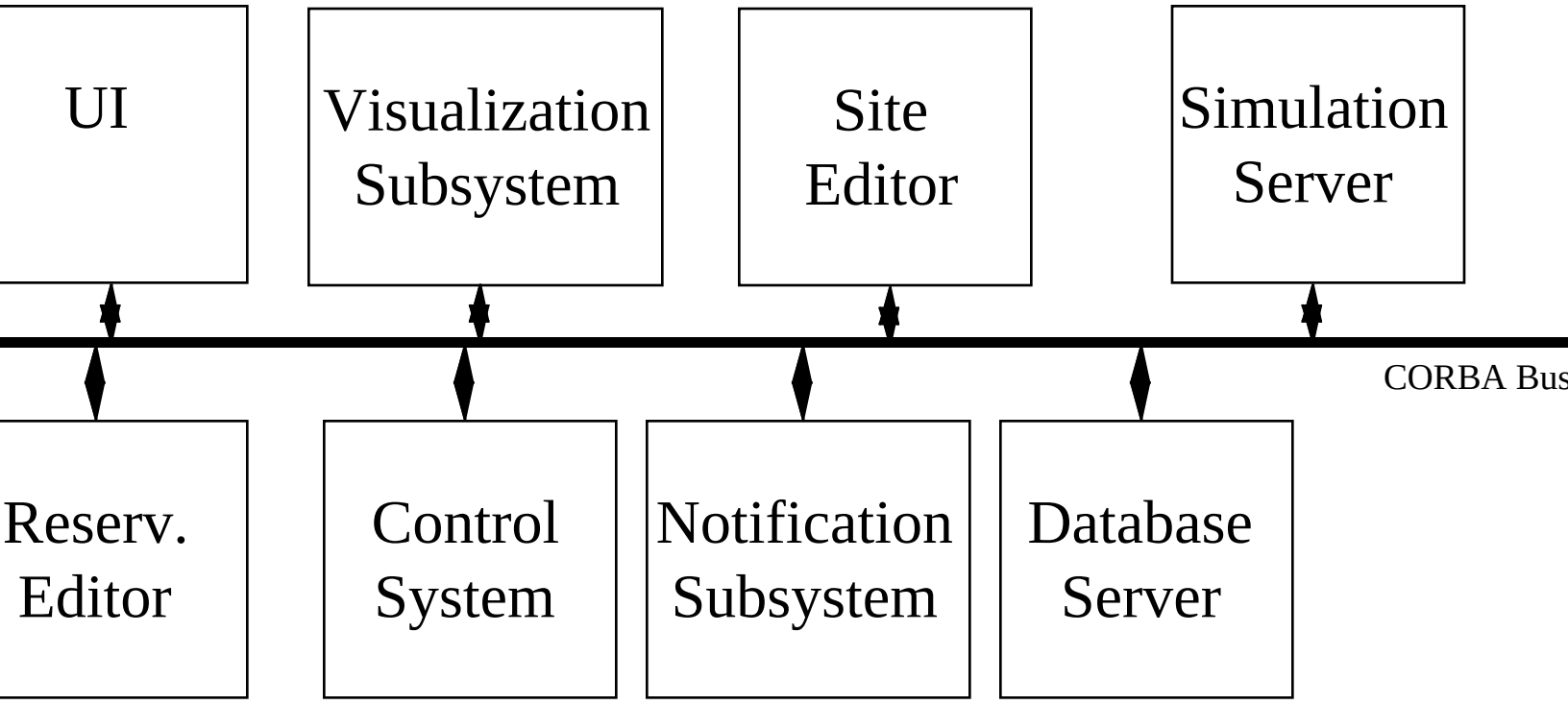
OWL Overview (as of Fall 1996)



Software Archaeology for the 1996 OWL Project

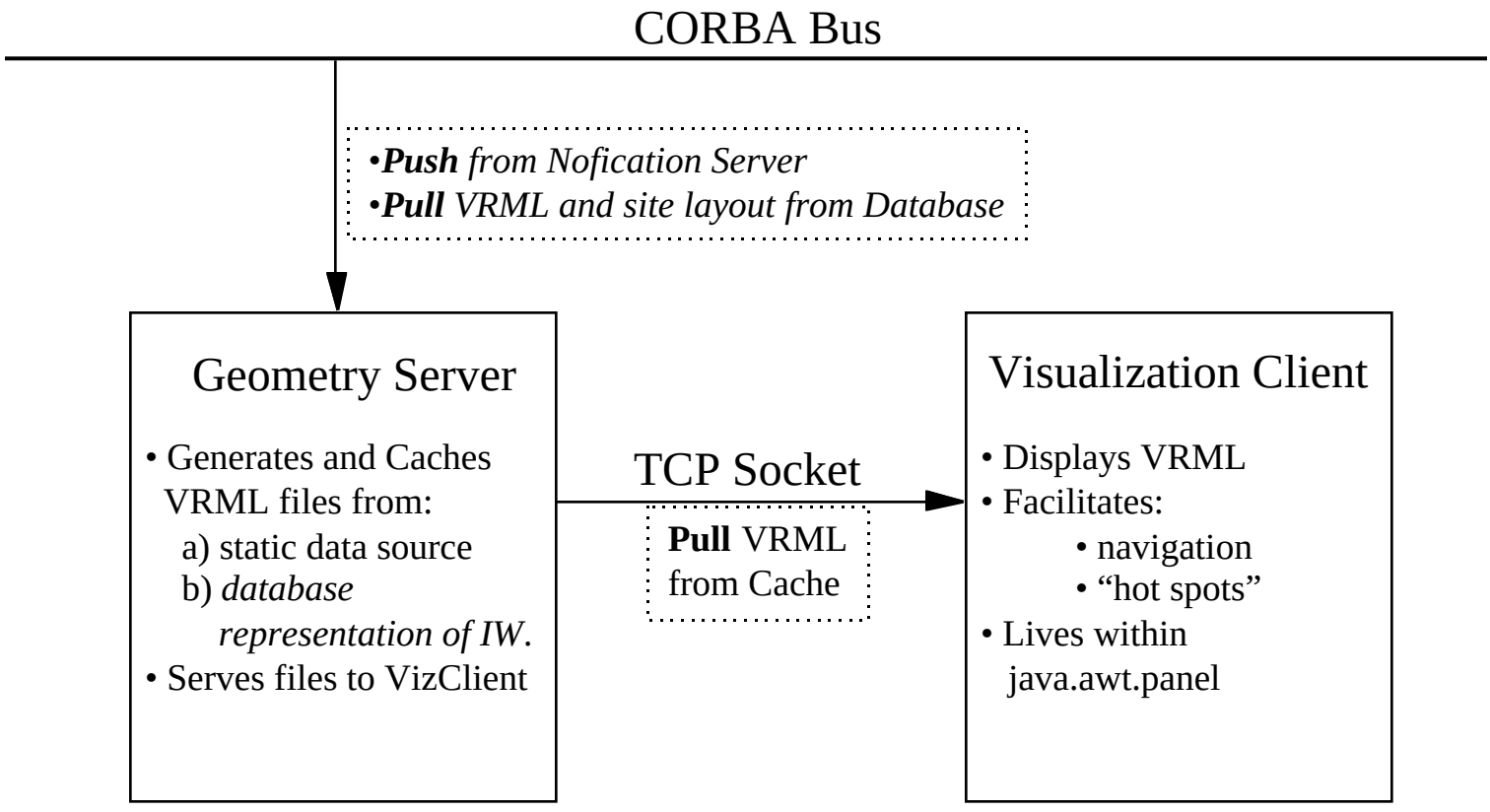
by

Truman Fenton, Ricardo Pravia,

and Aseem Sharma

Visualization Subsystem

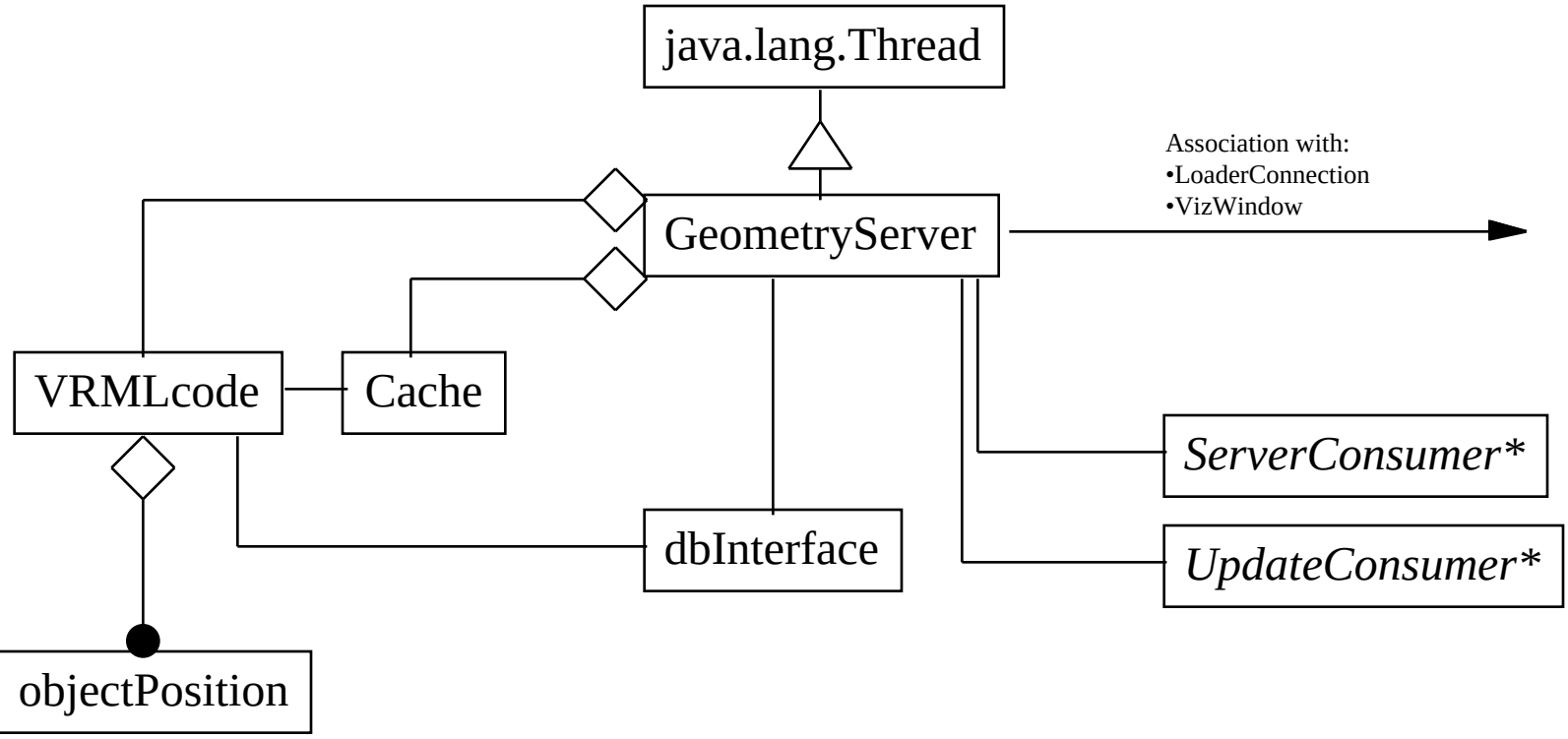
Package: edu.cmu.cs.se.owl1996.visualization



** unimplemented*

Geometry Server

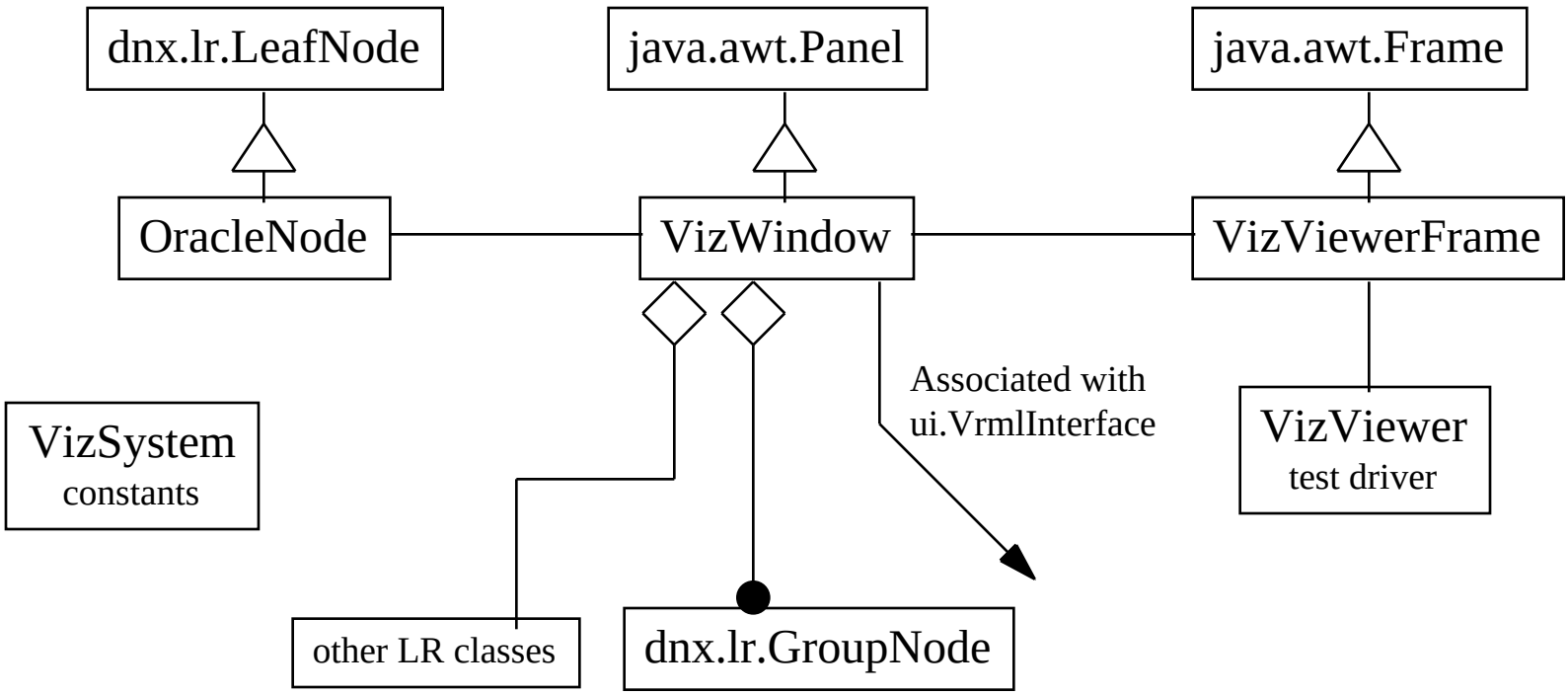
Package: edu.cmu.cs.se.owl1996.visualization.GeometryServer



** unimplemented*

Visualization Client

Package: edu.cmu.cs.se.owl1996.visualization.VizClient



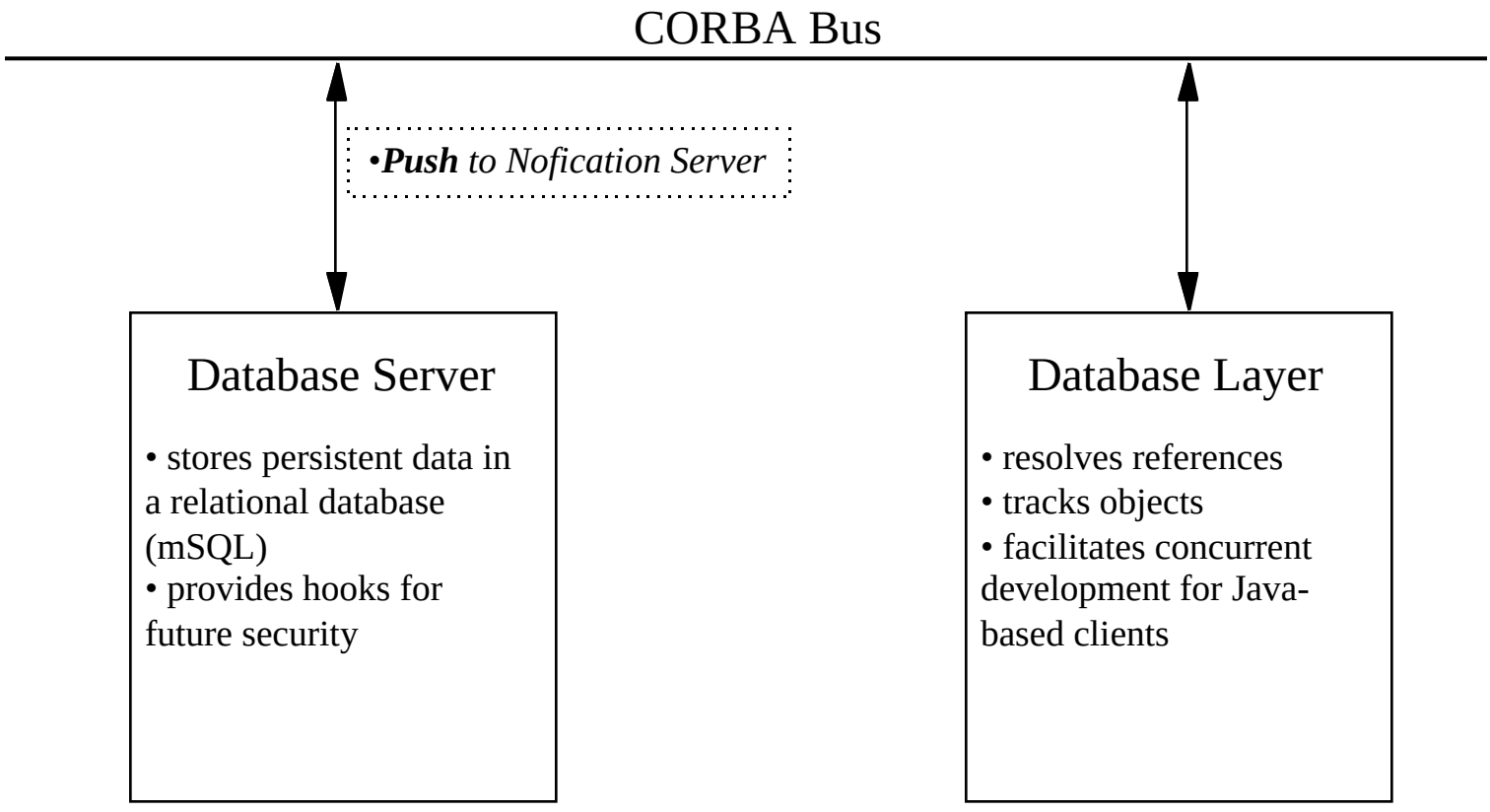
** unimplemented*

Visualization Status

- Tested with Database but not integrated.
- Tested and integrated with UI
- VizClient written to Liquid Reality Beta 8 and is incompatible with Beta 11 (current)
- Hot Spots are not fully implemented or demonstrated
- ODD is missing an object model

Database Subsystem

Package: edu.cmu.cs.se.owl1996.database



** unimplemented*

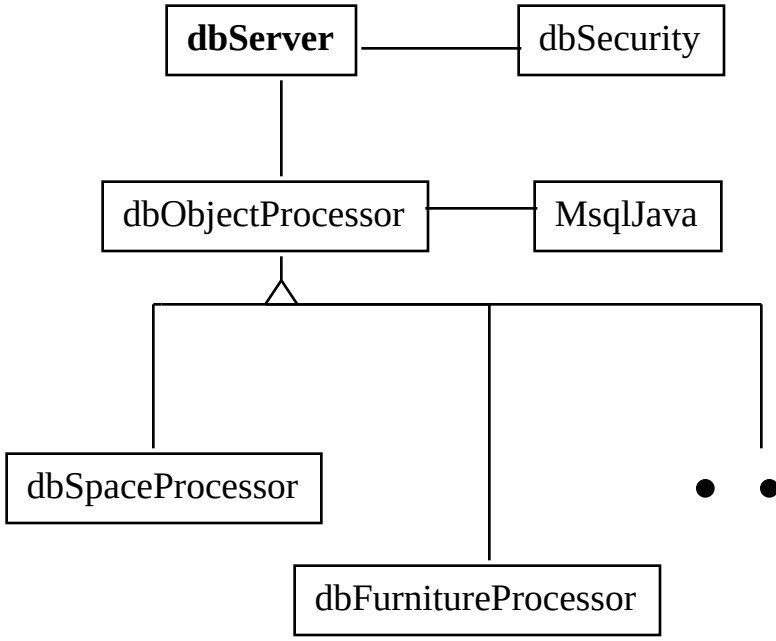
Database Server

Package: edu.cmu.cs.se.owl1996.database.dbServer

Server-side code which:

- enforces data consistency and locking
- converts from IDL generated objects to SQL

A client (or **dbLayer**) binds to this object over CORBA.

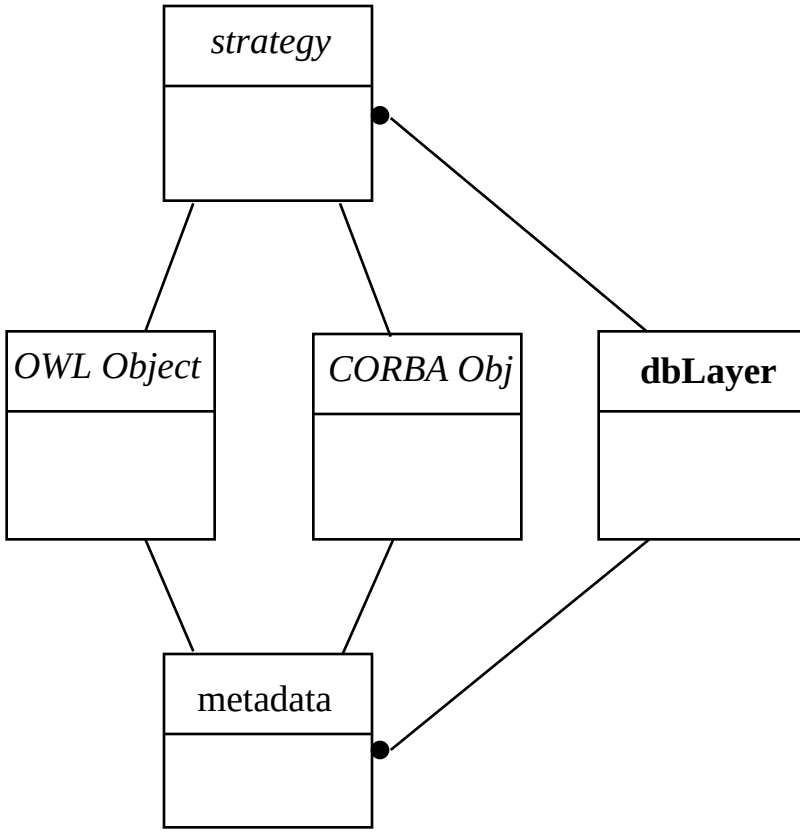


Database Layer

Package: edu.cmu.cs.se.owl1996.database.dbLayer

The client instantiates **dbLayer** and uses instances of *OWL Object* then **dbLayer**:

- tracks retrieved objects
- resolves references between objects
- sends changes to the database server



Database Status

- dbLayer not fully debugged: at least one known bug when saving a site to the database
- has too much “knowledge” of the usage of the data (e.g. site versions are cumbersom)
- is inefficient for making small modifications (e.g. adding a reservation)
- ODD is missing
- Tested with: Notification, Visualization, Site Editor, Reservation Editor

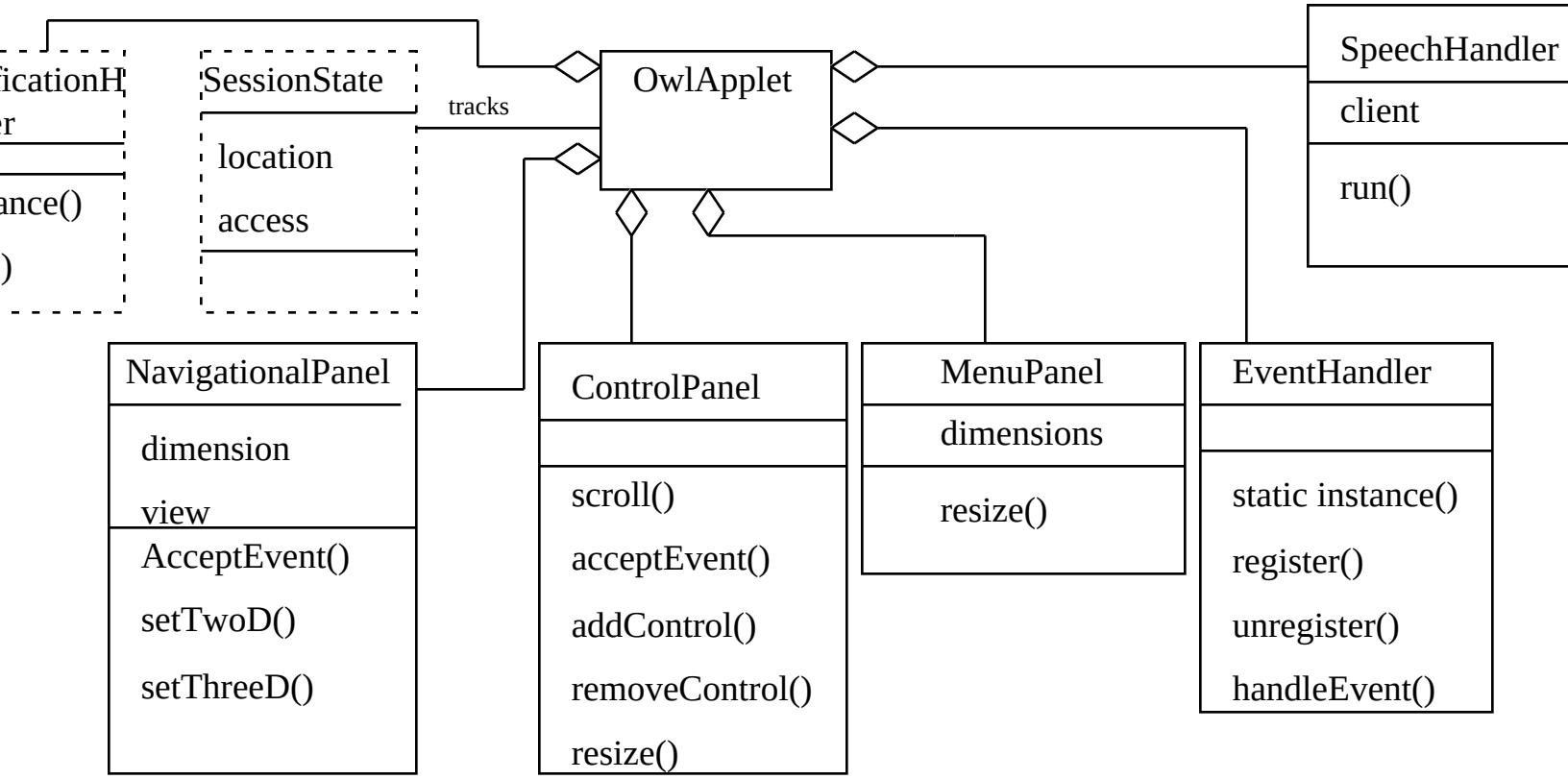
Functional Requirements of UI sub-system as of (Fall 96)

- The point - Provide Multi-Modal, platform independent interface to the OWL sub-system.
- The different types of Users will interface the system using the UI sub-system. The only two exceptions to this are the 3D model (Visualization) & the Site Editor (FMS).
- Allow users to view and control all aspects of their work place, like light levels, temprature, etc.
- Providing visualization of the current layout of the entire building.
- Give different types of users, access to do different things.

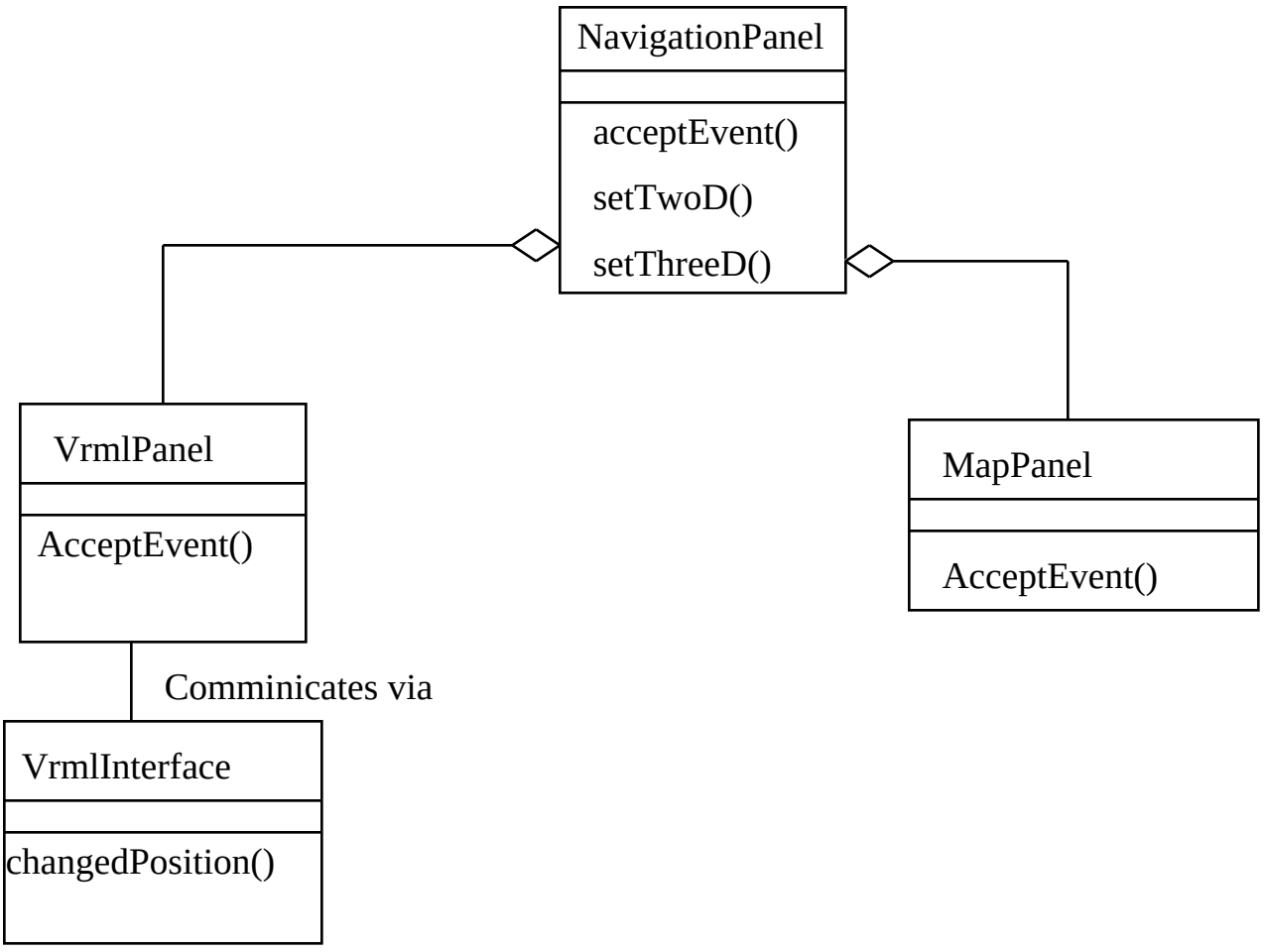
Sub-System Communication of the functional requirements

- Needs to talk to
  - Visualization Sub-System to display the VRML model in the navigational panel.
  - Control Sub-System, to display the controls for appropriate spaces.
  - Database Sub-System to get the layout, user-profiles,..
  - Notification to listen to the events happening
  - Speech Server to interact through speech

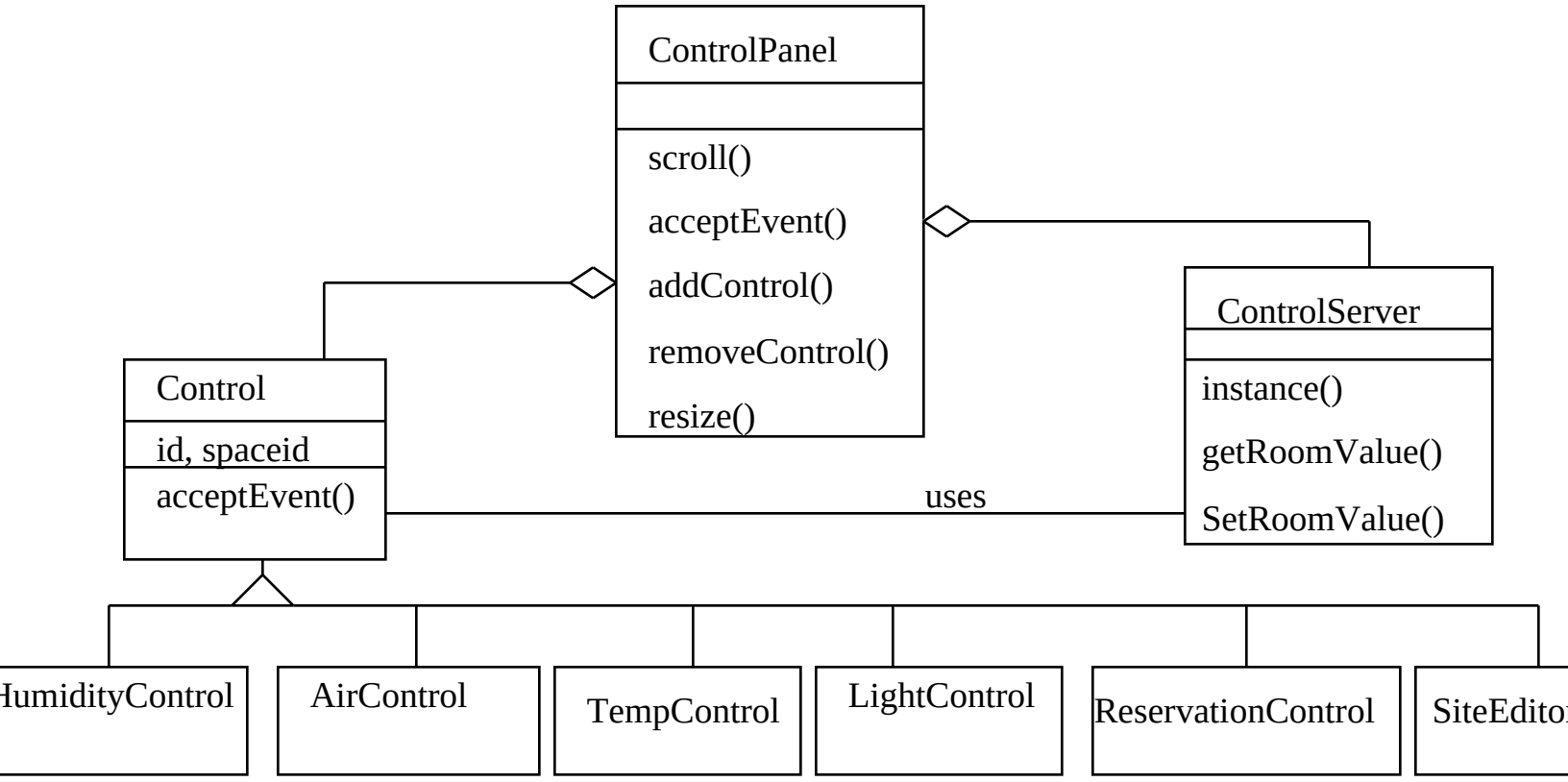
The Object Model



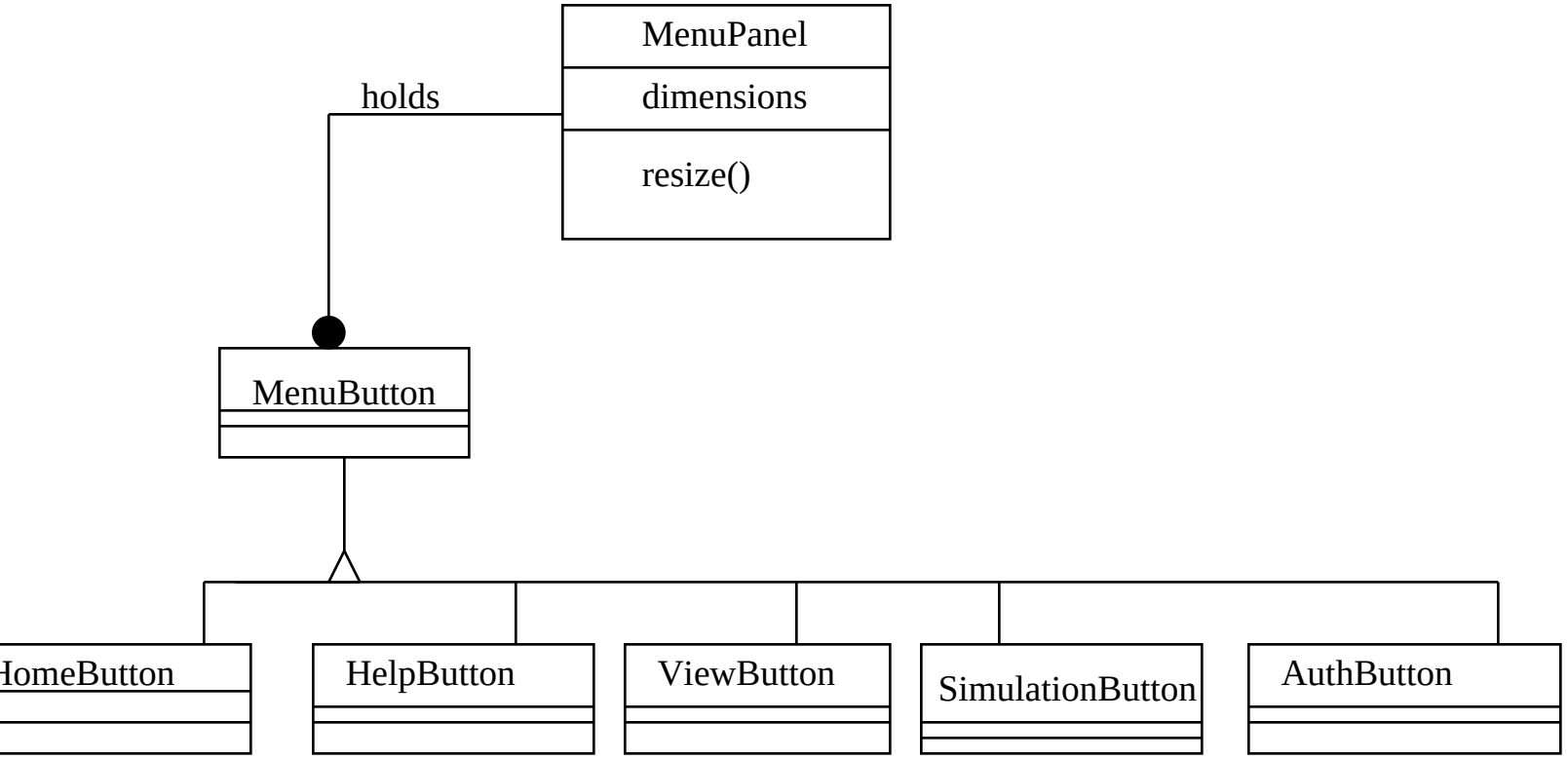
Object Model for NavigationPanel



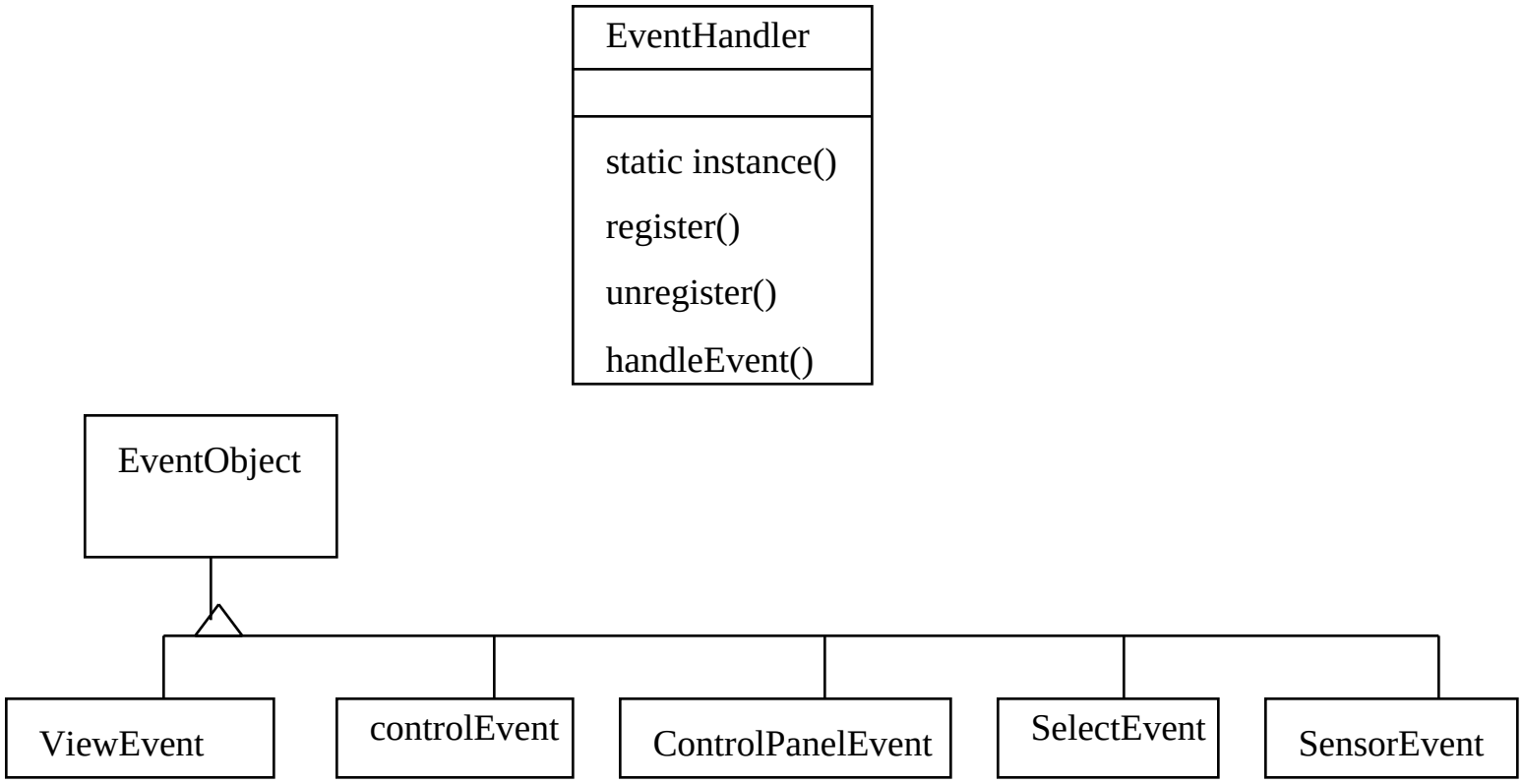
Object Model for ControlPanel



Object Model for Menu Panel



Object model for Event Handler



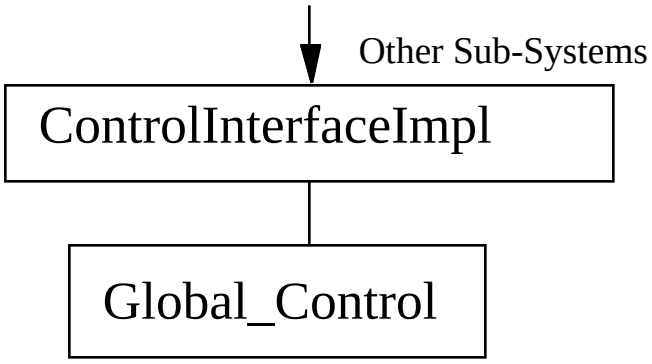
Integration Status

- Ability to switch between a 2D and 3D view
- Construction of a 2D map
- Ability to adjust provided controls.
- Unit testing done.

Control Overview

- Control the light levels in the Intelligent Workplace by manipulating the light louvers, lighting shades and incandescent light.
- The system will maintain user defined light levels in an energy efficient way.
- Also provide an API for an HVAC system.

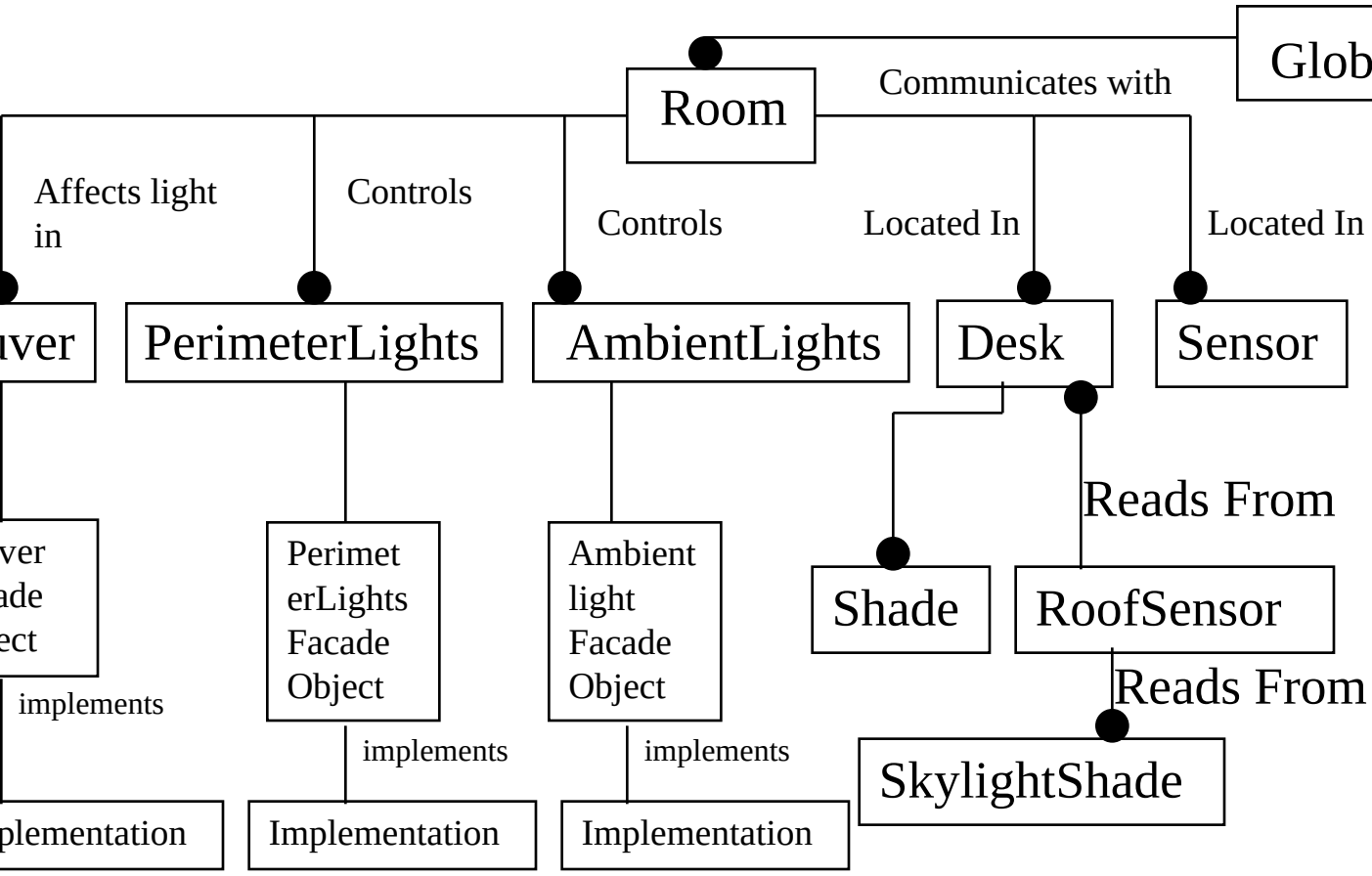
Object Model for Control



•ControlInterfaceImpl is the object that implements the interface to the control sub-system. It is the primary object of the sub-system that calls other objects.

•Global_Control object - maintains state of entire Sub-system. Keeps track of all the controls in all the relevant spaces.

Global_control Object Model



Status of the Sub-System

- Global_control Object is right now hard-coded to have some objects like 3 desks, etc.
  - Hence Integration with UI,Viz., not possible right now.
- A building right now is just a selection of rooms. The idea of floors is not present.
- Code does exist to integrate with Notification but is not used. So, now, cannot notify UI for changes in controls.
- Sensors & Actuators done through Interfaces. You implement these interfaces using objects that can change/ read values to/from hardware. Right now implementations do nothing.

Integration Status

- Unit Testing: The software subsystem was bottom-up tested from the hardware bridges up to the Room control module.
- The hardware problems did not allow the system to control with the hardware.

Reservation Editor Subsystem

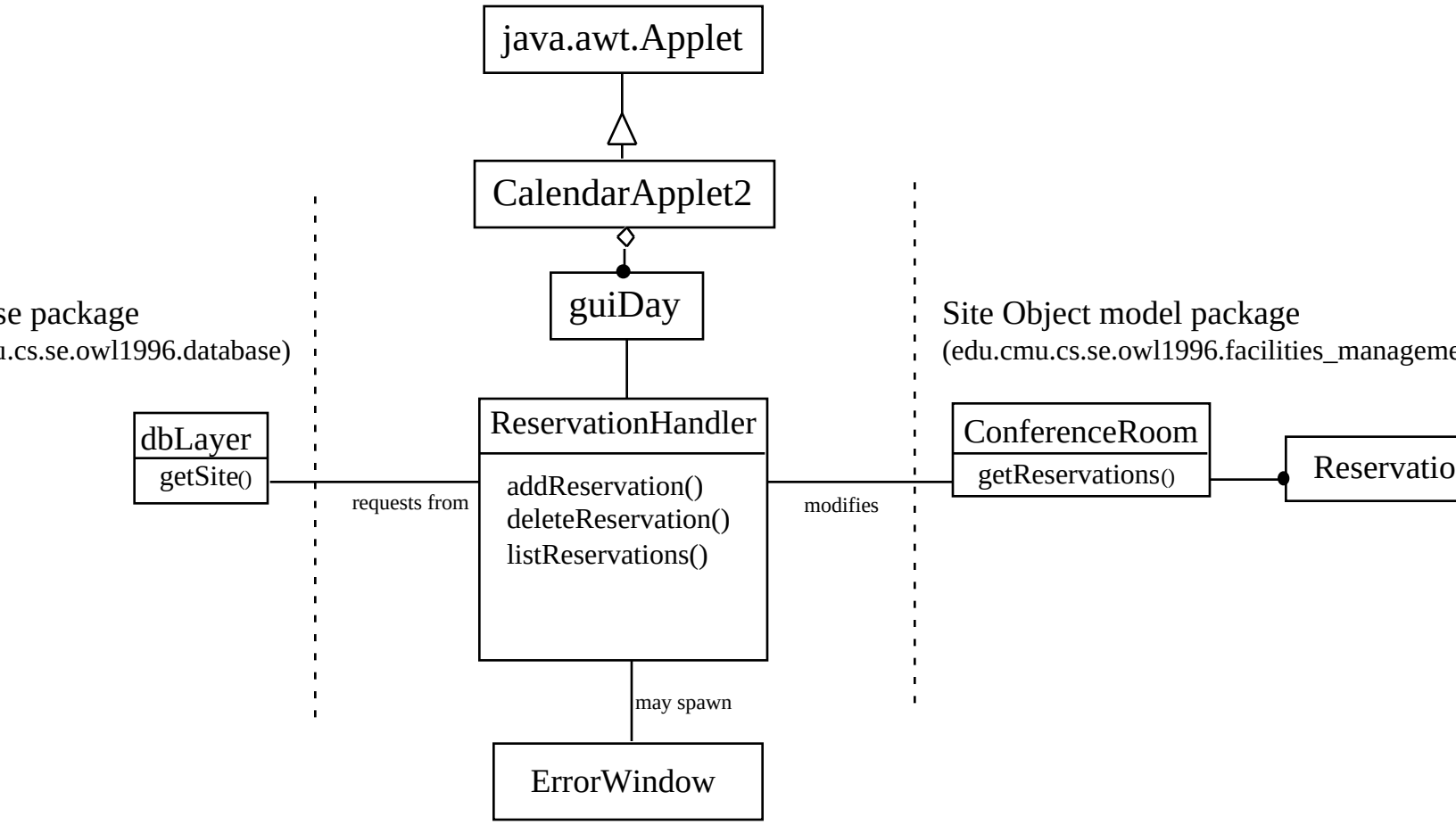
- Responsible for adding, deleting or changing reservations for Conference Rooms
- Provides a GUI interface
- Saves changes to the database*

**not implemented*

Reservation Editor Use Case

- User uses the UI interface, selects a room and hits reservation editor icon (single-click)
- UI Calls the Reservation editor with the Conference room id
- Reservation Editor then spawns a calendar
- User selects a day and the Editor spawns a reservation list window
- User can then add, delete or modify reservations

Reservation Editor Subsystem



Reservation Editor Status

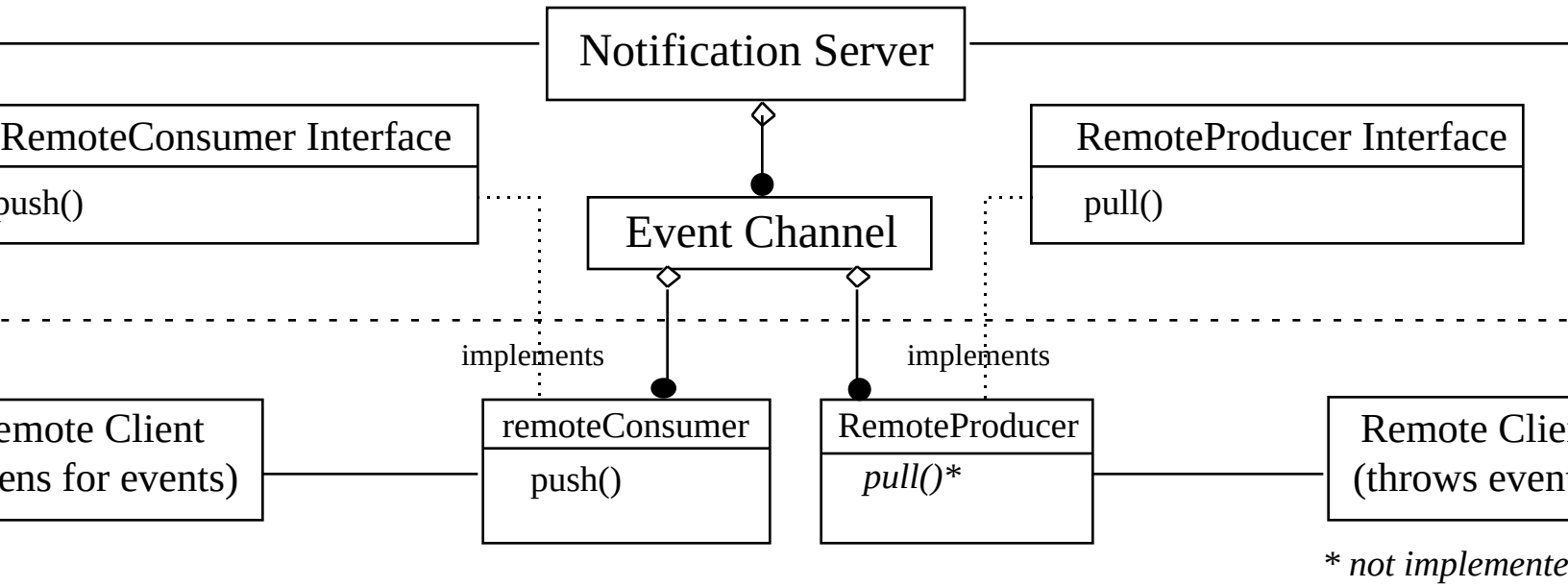
- Integrated with:
  - UI
- Tested with (only double tested)
  - UI, Database
- Conference Room id is hard coded
- Can only be run under JDK 1.02

Notification Subsystem

- Handles notification of events between producers and consumers
- Producers subscribe to a specific Event Channel as suppliers (of events)
- Suppliers subscribe to a specific Event Channel as listeners (of events)

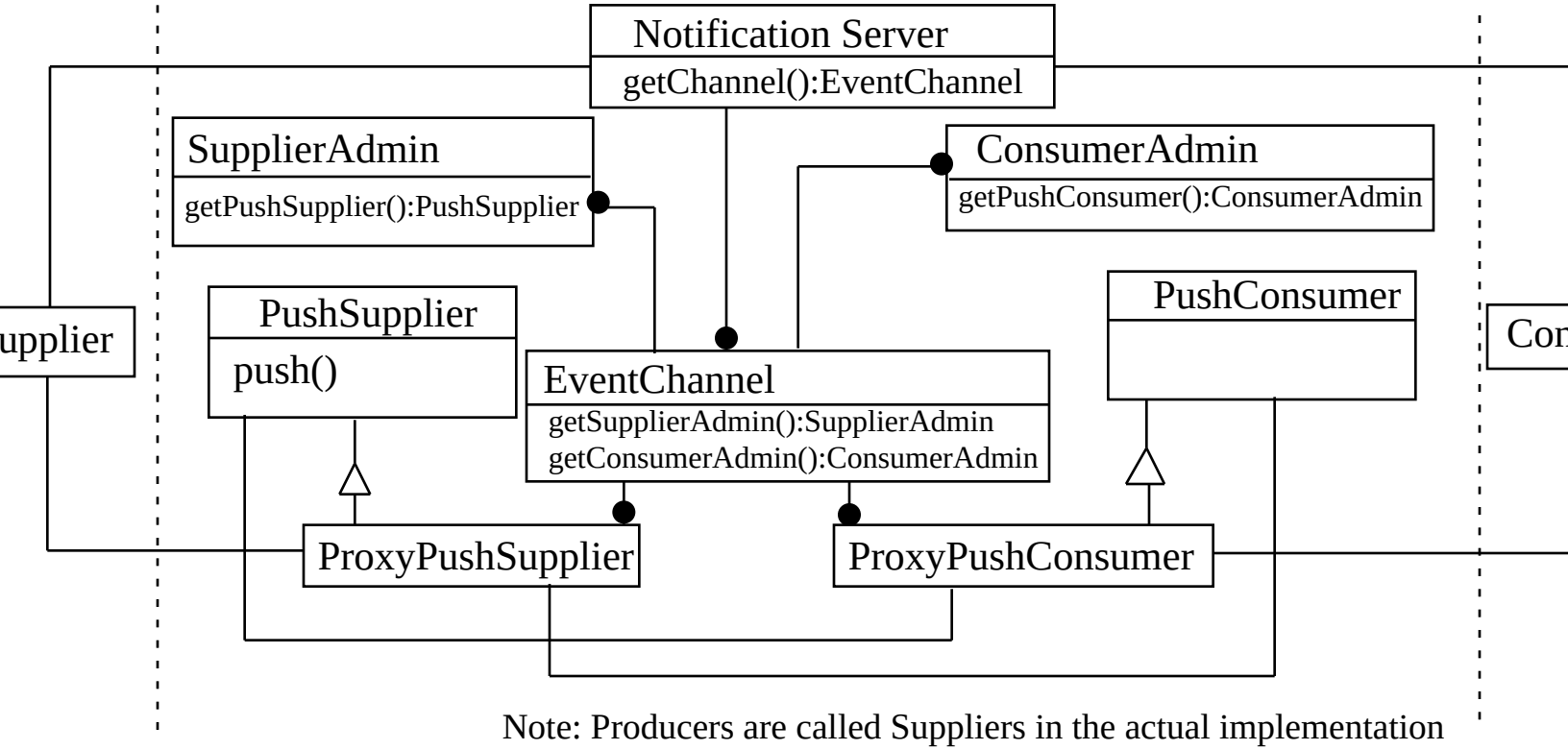
Note: Producers were called Suppliers in the Design

Notification System Design



onsumer Clients must create a remote interface and export it to t
erver (in order to get notified)
oducers must also create a remote interface and export it
the Server (in order to get *Pulled**)
he system design is different from the actual Object Model

Notification Object Model



Note: Producers are called Suppliers in the actual implementation

- Unnecesarily complex object model
- Good reusable parts, though
- Too complicated Use Case
- Events must be declared at compile time

Client Use Case

- For a Consumer to connect to an event channel, it must:

```
RemoteConsumerInterface my_interface;
EventChannel a = NotificationServer.getEventChannel();
ConsumerAdmin b = a.getConsumerAdmin();
ProxyPushConsumer c = b.getPushConsumer(my_interface);
```

  - Far too complicate for a Consumer.
  - Similar process for the Producer

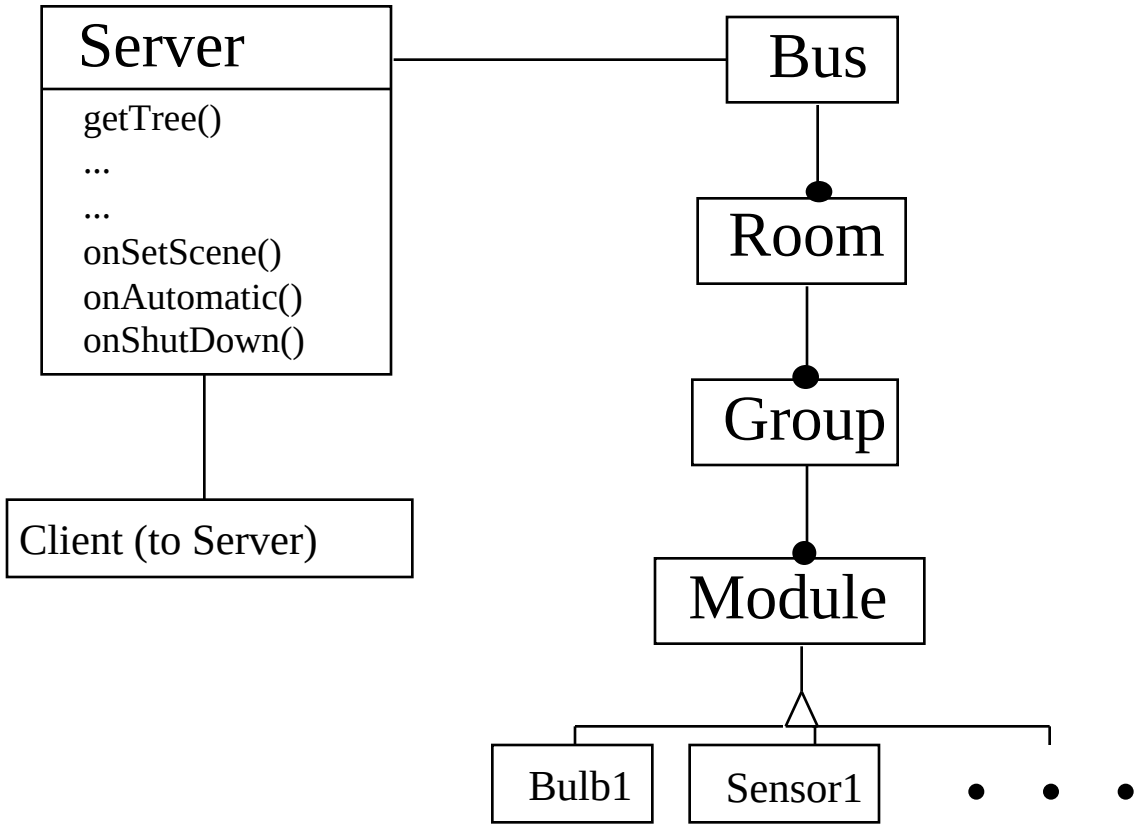
Notification Status

- Integrated only with:
  - Control
- Tested (only double tested with)
  - Database, Control, UI
- There is a bug in the actual implementation
- Only two events available:
  - FloorPlan changed
  - Sensor Event

LUXMATE

- An object oriented lighting control system
- Provides server software that:
  - Hides the connection details
  - Simulates an event driven system
  - Provides a software replica of the light system

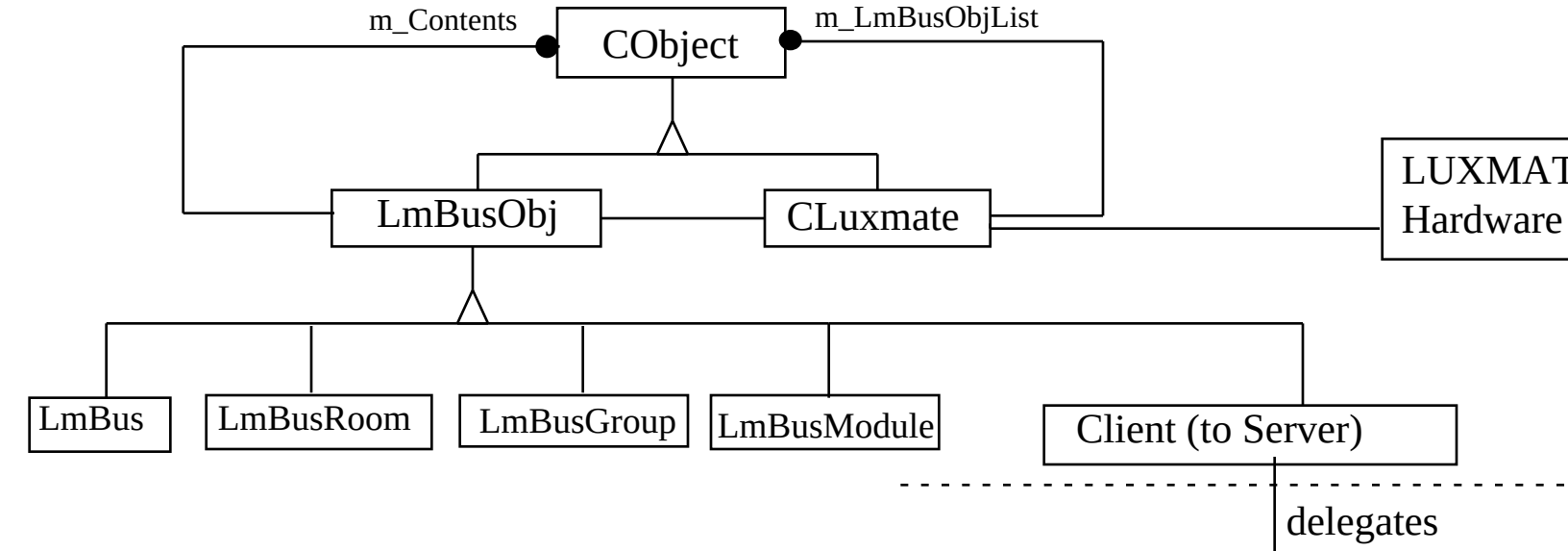
LUXMATE Architecture



nts methods are overridden by the Client

ctual implementation the Client gets a reference to Server by subclassing fro

LUXMATE Object Model



- Client must inherit from LmBusObj (can be Server)
- LmBusObj contains a lot of functionality, inherited classes rarely add much
- Events are declared in LmBusObj and the client must implement them
- CLuxmate communicates with hardware
- We will probably naval off after the Client Object

Status of LUXMATE

- Bound to Visual C++ 1.52 (VC++)
- Hard to test all functionality because of VC++’s steep learning curve
- Events must be tested in order to document the actual behaviour of them
- Need to better document the behaviour of the Server