

# Object oriented modeling

ADORA

Sebastian Sanchez, Mario Viso

---

# What are the key features of the ADORA modeling language?

Main features that distinguish ADORA from other approaches:

- **Abstract objects** instead of classes
- Structuring with **hierarchical decomposition** of the modelled system
- Using an **integrated model** instead of collections of models
- Allowing users to express different parts of a specification with an **adaptable degree** of formality
- **Visualizing models** in context with **generated views** by presenting details of a model always with an abstraction of its surrounding context

# Abstract objects

Problem:

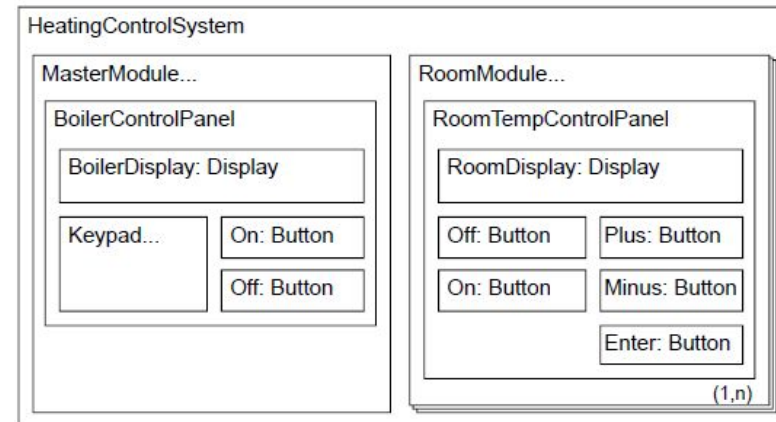
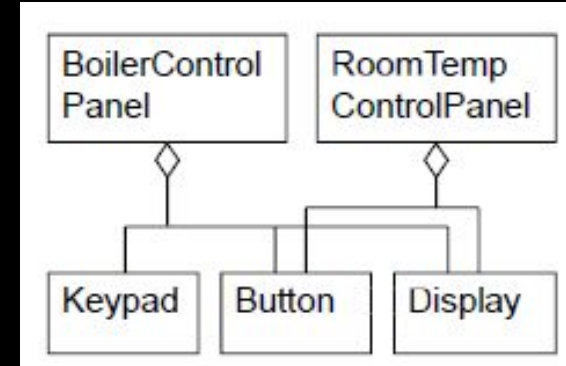
Class models don't work when different objects of the same class need to be distinguished and can't be decomposed properly.

Solution:

**Abstract objects:** represents a single instance

**Abstract object sets:** represent a set of instances

**Types:** an equivalent to classes

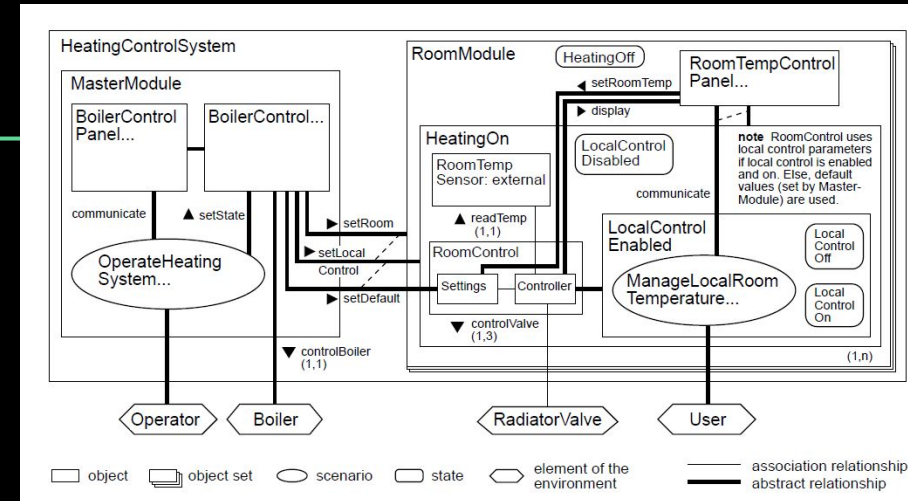
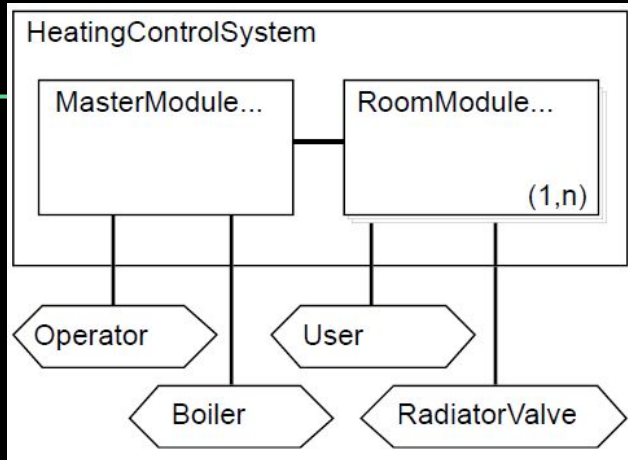


# Hierarchical decomposition

Problem:

Good decomposition is not possible by modelling systems as collections of models or by using container constructs

Solution:



# Integrated model

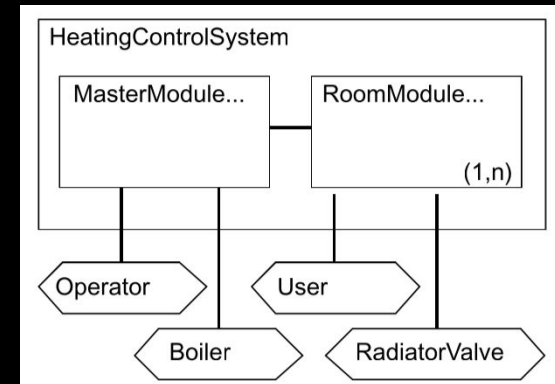
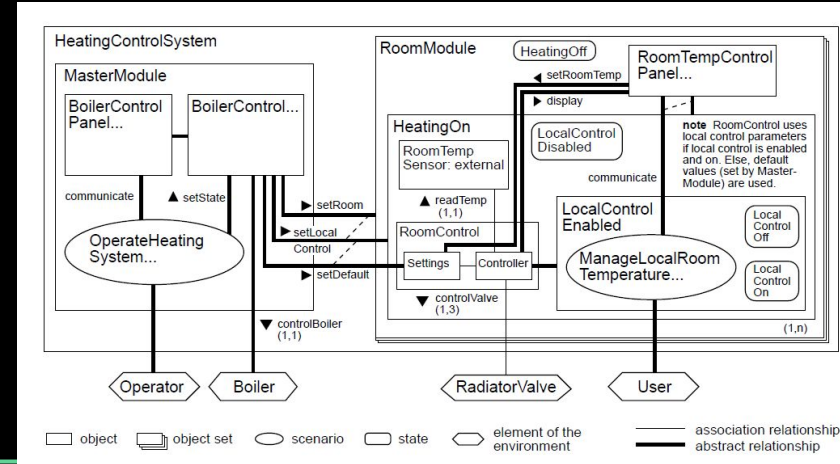
Problem:

Loosely coupled collections of models make the equally important issues of integration and abstraction of concerns quite difficult

Solution:

ADORA integrates all modeling aspects:

- structure, data, behavior, user interaction
- in one coherent model



# Varying degrees of formality

A modeling language should allow its users to adapt the degree of formalism in a specification to the difficulty and the risk of the problem at hand

Therefore ADORA gives the modeler the choice between:

- Informal specifications
- Formal specifications
- Textual specifications
- Mixture of both

```
object Settings
part of RoomControl
provides ActualTemp;
    "Operations to inspect / manipulate control intervals (consisting
    of start time and desired temperature), both default and user-
    defined"

requires //nothing

type
    TempIntervals is list of TempInterval;
    TempInterval is structure of (start: Time, temp: Temperature)

attribute //public attributes
    ActualTemp: Temperature

var //private attributes
    DefaultIntervals: TempIntervals; //default temperature settings
    UserSetIntervals: TempIntervals //user-defined temperature settings

syncoperation CurrentTemp (t: Time): Temperature
    pre 1 ≤ t ≤ 24*60 //minutes
    post with usi = UserSetIntervals;
        //returns desired temperature at current time t
        CurrentTemp = usi[i].temp and usi[i].start ≤ t and
        not exists j • (usi[i].start < usi[j].start ≤ t)
    end CurrentTemp

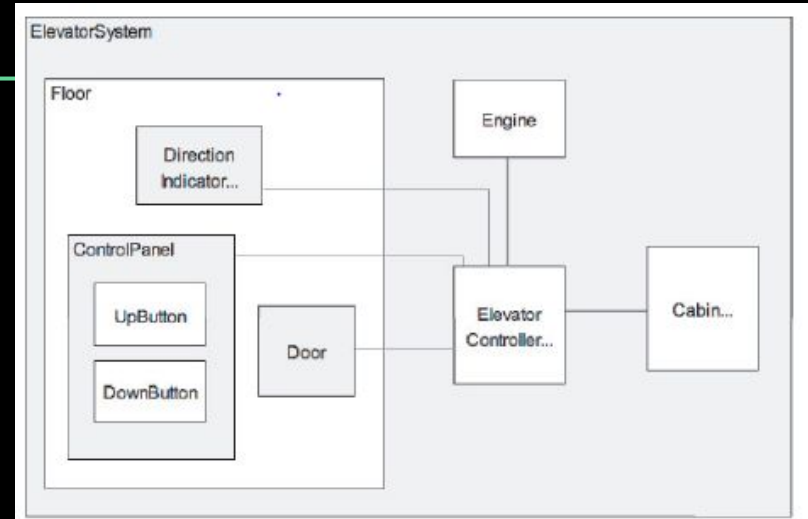
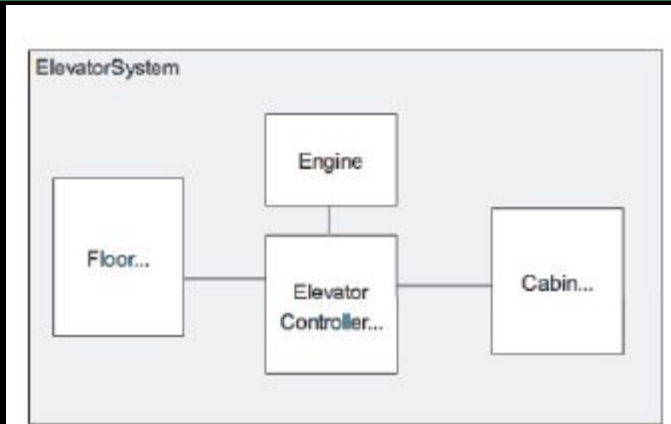
syncoperation DefaultTemp (t: Time): Temperature
    "Same as CurrentTemp, but returns current default value"
    end DefaultTemp

operation RevertToDefault
    post UserSetIntervals' = DefaultIntervals
    end RevertToDefault

"Settings must also provide operations for setting and deleting intervals and
for browsing the currently defined intervals."
end Settings.
```

# Contextual Visualisation

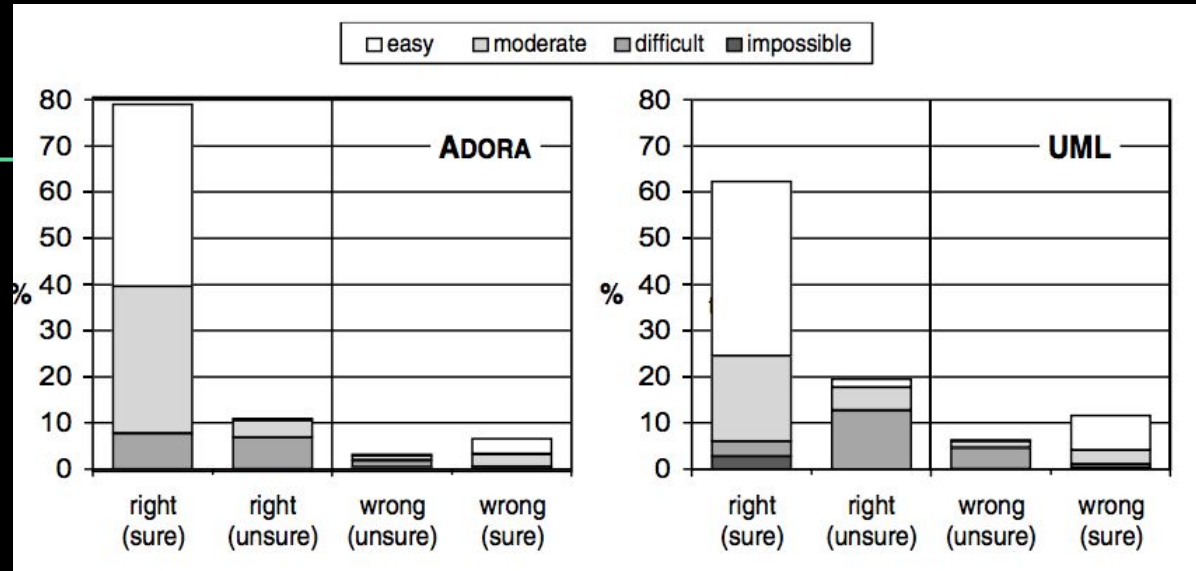
- Fisheye view concept (zoom in and zoom out)
- Layout adjustments (hierarchy, smart editing, node filtering)
- Context diagram depending on the selected degree of abstraction
- External objects (e.g. COTS components) can be also represented
- One coherent model, where size and complexity of diagrams are kept within reasonable limits



# What are the main differences between ADORA and UML?

- In UML There is still no true integration of the aspects of data, functionality, behavior and user interaction.

- There is no systematic hierarchical decomposition of models.



## UML

In UML, the modeling approach is decomposition of the problem by providing a container construct in the language that allows the modeler to partition a model into chunks of related information.

The behavior model and the class or activity model are modeled and kept separately

Can't express constraints on functional and nonfunctional requirements.

## ADORA

In Adora, modeling is done recursively to decompose objects into objects. So we have the full power of object modeling on all levels of the hierarchy and only vary the degree of abstractness.

“Combining objects and states. Adora combines the object hierarchy with a statechart-based state machine hierarchy in a single hierarchical structure”

The requirements themselves are expressed as Adora standardized properties. Every kind of non-functional requirement can be expressed by its own property kind.

Table 2  
Comparison of basic concepts of ADORA vs. UML

| ADORA                                                                                                           | UML                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Specification is based on a model of abstract objects, types are supplementary.                                 | Specification is based on a class model, object models are partial and supplementary.                                                               |
| Specifies all aspects in one integrated model; separation of concerns achieved by decomposition and views.      | Uses different models for each aspect. Separates concerns by having a loosely coupled collection of models.                                         |
| Hierarchical decomposition of objects is the principal means for structuring and comprehending a specification. | Class and object models are flat. Only packages can be decomposed hierarchically.                                                                   |
| Scenarios are tightly integrated into the specification; they can be structured and decomposed systematically.  | Use cases (= type scenarios) are loosely integrated with class and object models. Structuring capabilities are weak, decomposition is not possible. |
| Precise rules for consistency between aspect views.                                                             | Nearly no consistency rules between aspect models.                                                                                                  |
| Conceptual visualization eases orientation and navigation in the specification and improves comprehensibility.  | UML tools provide traditional scrolling and explosive zooming only.                                                                                 |

# How is navigation done in ADORA models?

We have two types of navigation: Physical and Logical.

---

On the physical we zoom to reach details that are not shown and in the logical we go through all the hierarchical structure zooming in and out.

# Questions

- What are the challenges related to capturing model semantics using UML or ADORA?
  - Do you think ADORA can be a substitute for UML 2.0?
-