Tampa Presentation
The OOram Object Analysis & Design Metamodel

Trygve Reenskaug

15 January 1997

TASKON
Work Environments

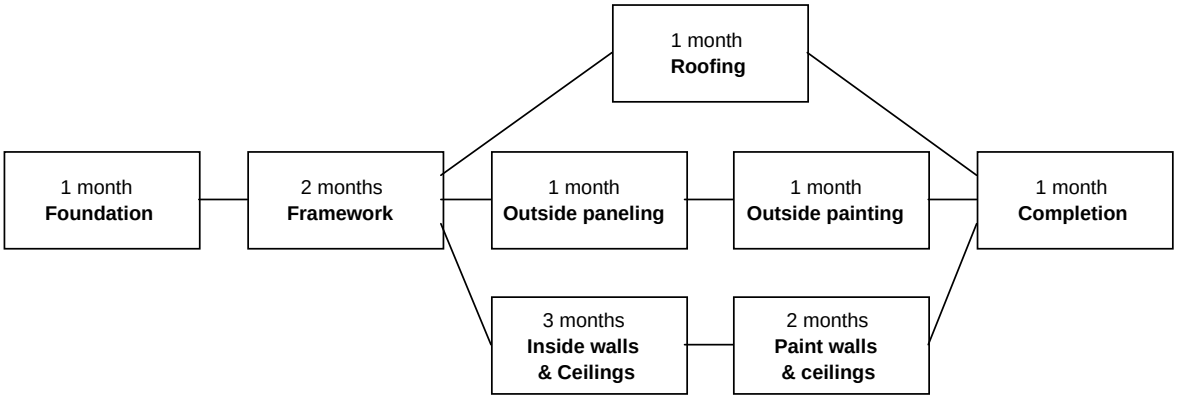
Taskon, Gaustadalléen 21, N-0371 Oslo 3 Norway. Tel. + (47) 22 95 86 31 Telefax: + (47) 22 60 44 27

1.
The OOram Meta-Model

combining
role models, interfaces, and classes
to support
system centric and program centric modeling

Submitted by
Taskon A/S
Reich Technologies
Humans and Technology

1.1.
Example Activity Network



1.2.
OOram Proof of Concept

- >20 years experience with OO system modeling
- >15 years evolution of OOram method and tools
- >5 years on the OO tool market
- Applied to
  - Design of programs and reusable frameworks
  - Specialized software factory for telecom. product
  - Domain models for intelligent networks
  - Domain models for ship design
  - Enterprise modeling in banks and other industries

1.3.
Highlights

1) The model as an object structure

2) System Centric and Program Centric modeling

3) Rationale for adding System Centric approach

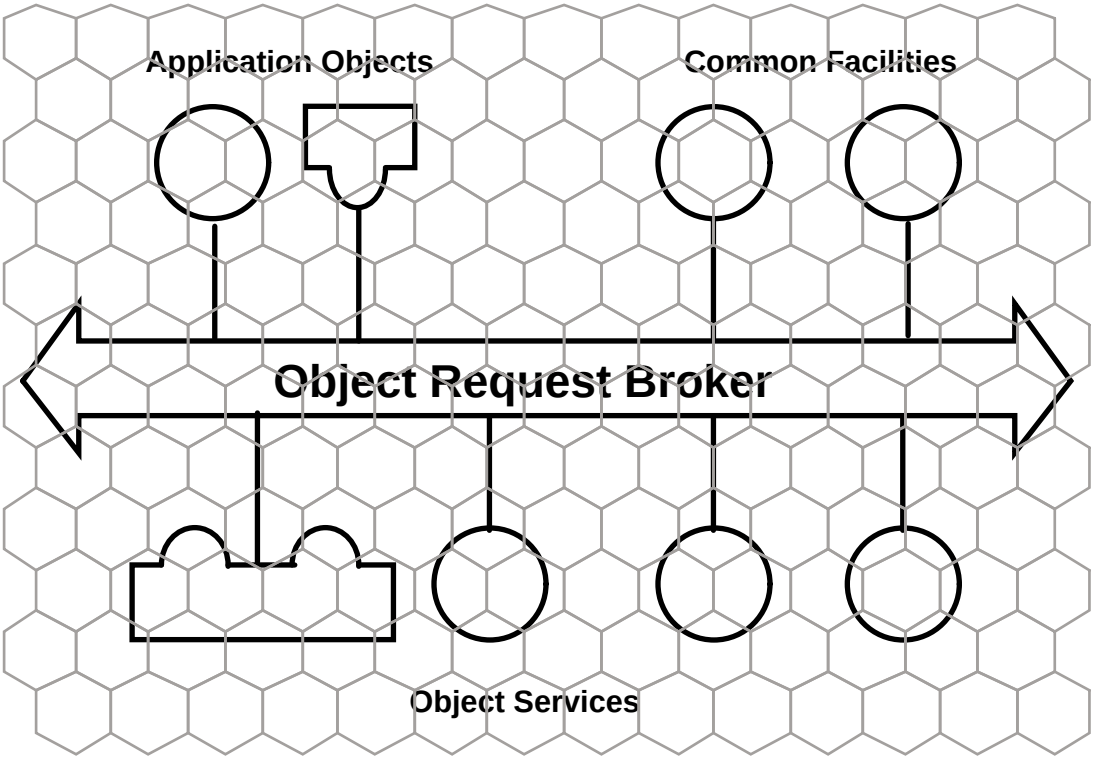
2.
Model as Object Structure

1) **The model as an object structure**
   - ¤ *Navigate model objects through CORBA*
   - ¤ *An INCLUSIVE model with common core*

2) System Centric and Program Centric modeling

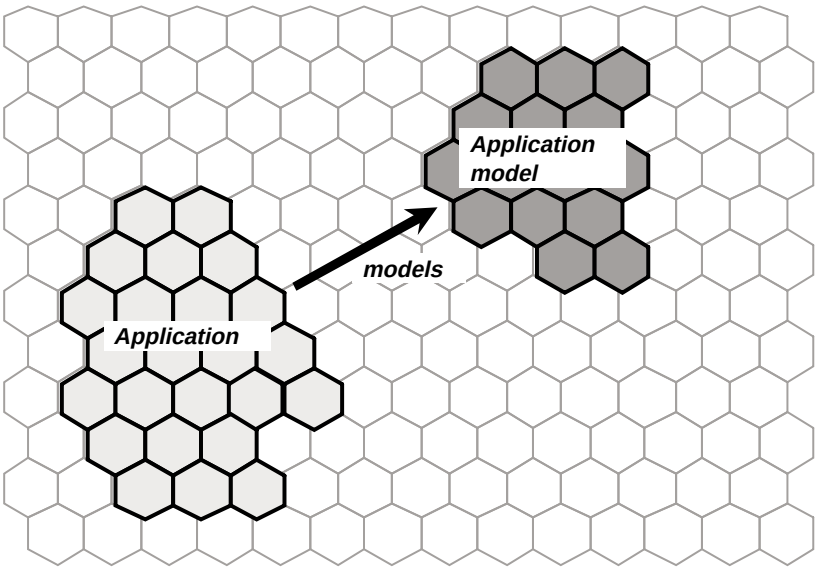
3) Rationale for adding System Centric approach

2.1.
OMG is all about collaborating objects



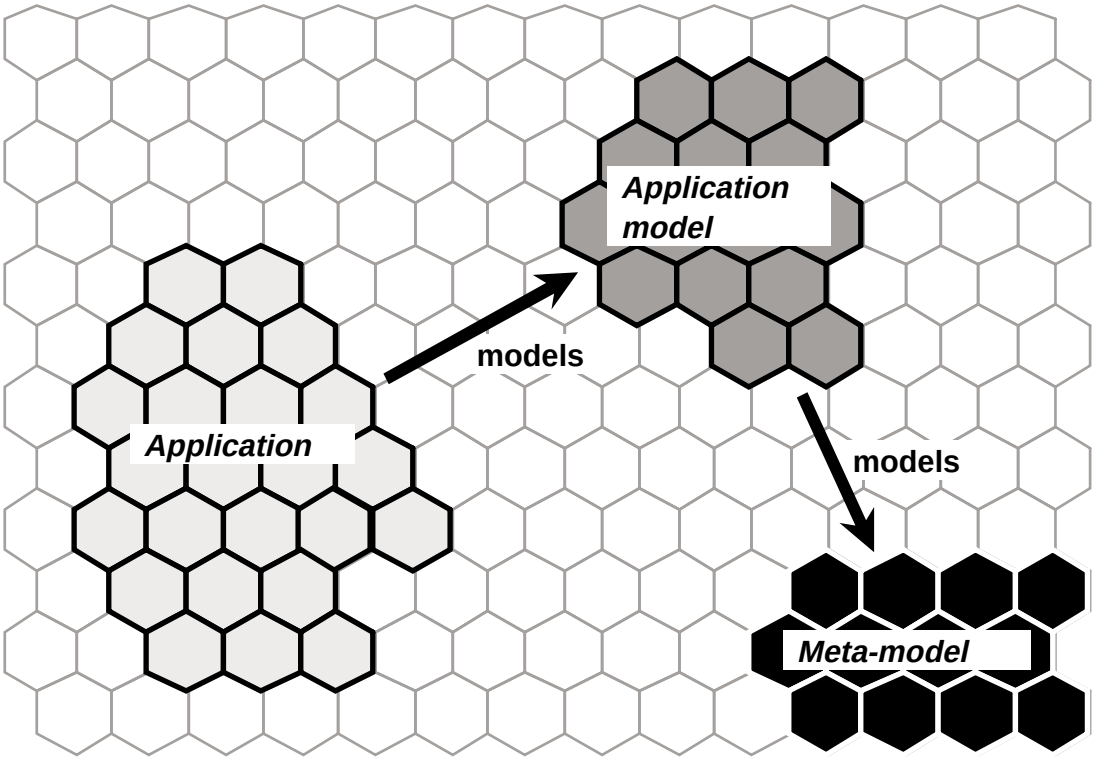
The world of OMG
consists of interacting objects

2.2.
Navigate model objects through CORBA

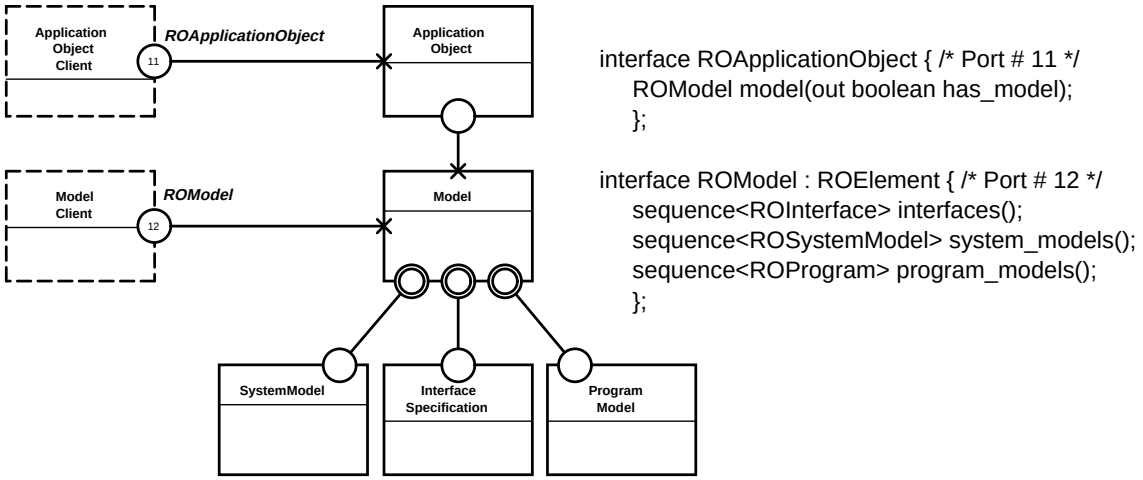


A system is a part of the real world which we choose to regard as a whole,
separated from the rest of the world during some period of consideration;
a whole that we choose to consider as containing a collection of objects,
each object characterized by a selected set of associated attributes and by acti
which may involve itself and other objects.

2.3.
Meta-Model models application model



2.4.
Model architecture for an INCLUSIVE standard




```
interface ROApplicationObject { /* Port # 11 */
    ROModel model(out boolean has_model);
    };

interface ROModel : ROElement { /* Port # 12 */
    sequence<ROInterface> interfaces();
    sequence<ROSystemModel> system_models();
    sequence<ROProgram> program_models();
    };
```

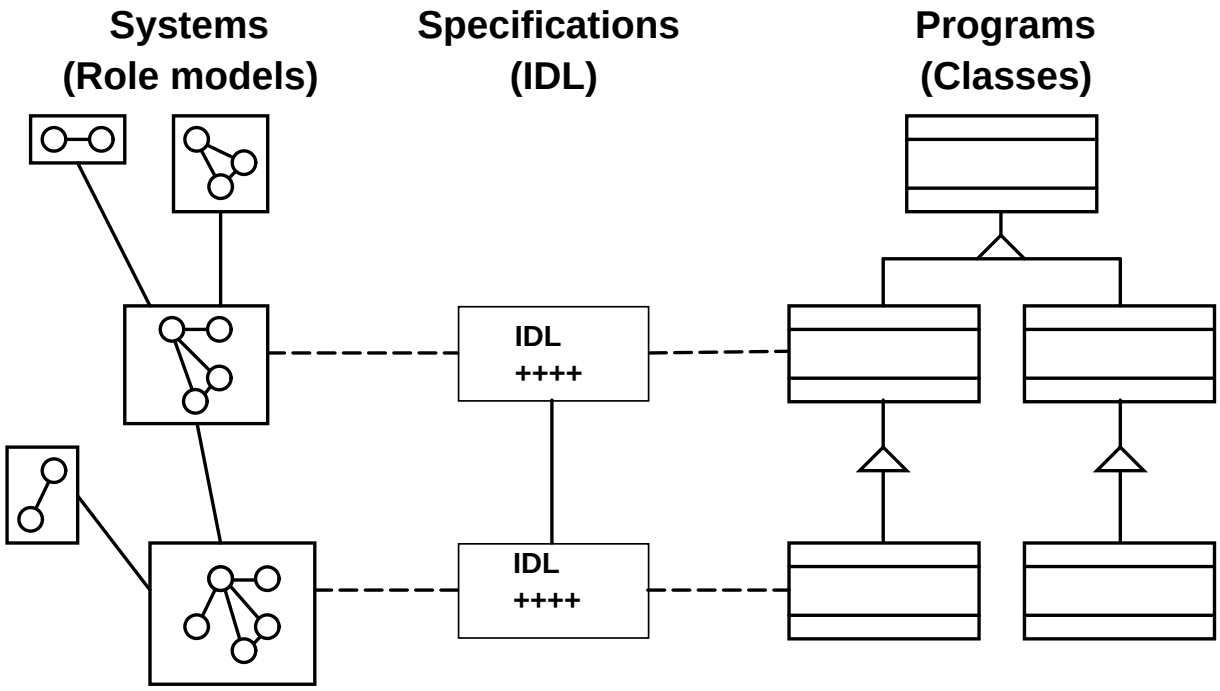
3.
System Centric and Program Centric modeling

1) The model as an object structure
   - ¤ Navigate model objects through CORBA
   - ¤ An INCLUSIVE model with common core

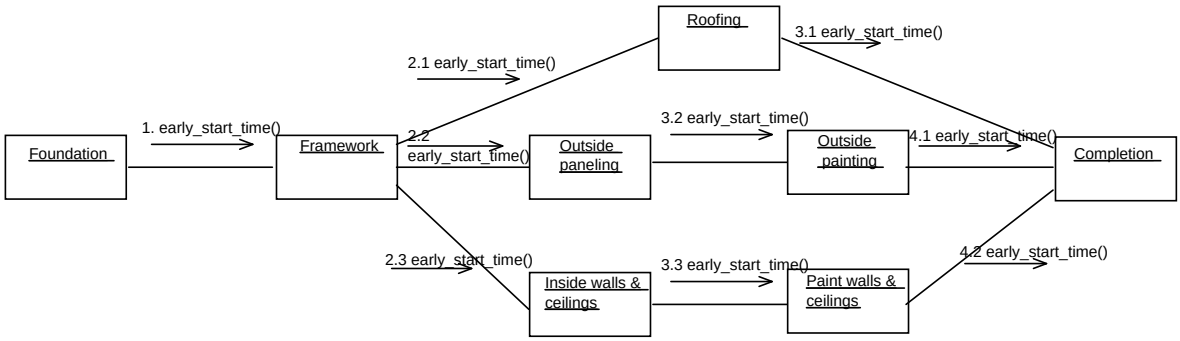
2) System Centric and Program Centric modeling
   - ¤ *Identity-preserving system models*
   - ¤ *Interfaces bound to collaboration paths*
   - ¤ *IDL interfaces first class citizens*
   - ¤ *System inheritance supplements class inheritance*
   - ¤ *Seamless transition System <-> Interface <-> Class*
   - ¤ *Clean extension of object models*

3) Rationale for adding System Centric approach

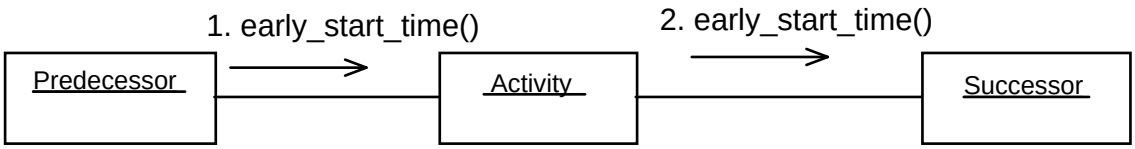
3.1.
System Centric and Program Centric Programming



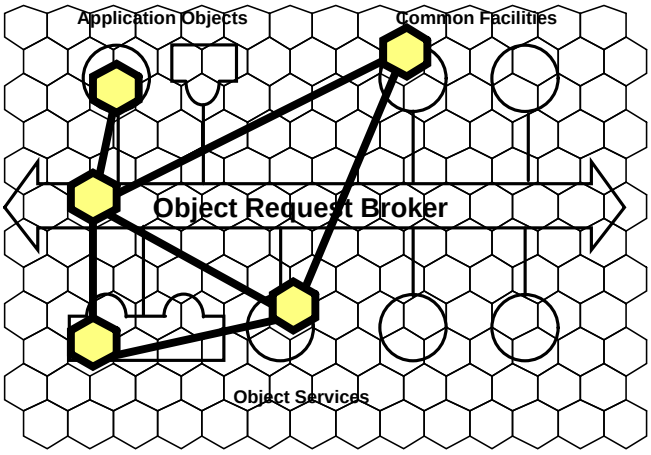
3.2.
Concrete object model of activity network



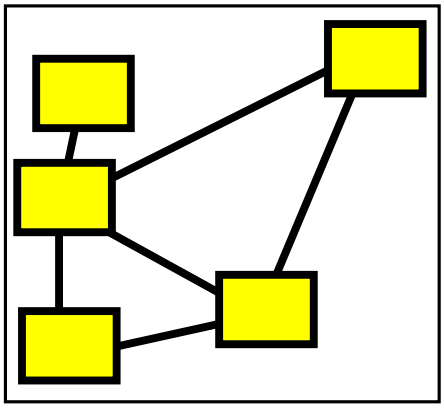
3.3.
Generic 'object' model of activity network



3.4.
The role model describes a system of interacting objects



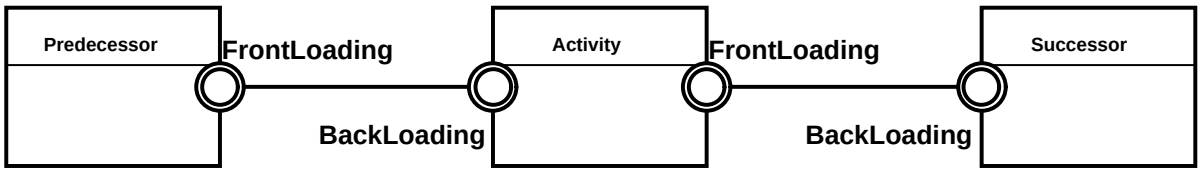
Object system



Role model

An activity is a task carried out by a set of collaborating objects.
In an instance of a role model, each role represents a single object involved in one or more activities.
The role only describes those object properties that are of relevance to these activities.

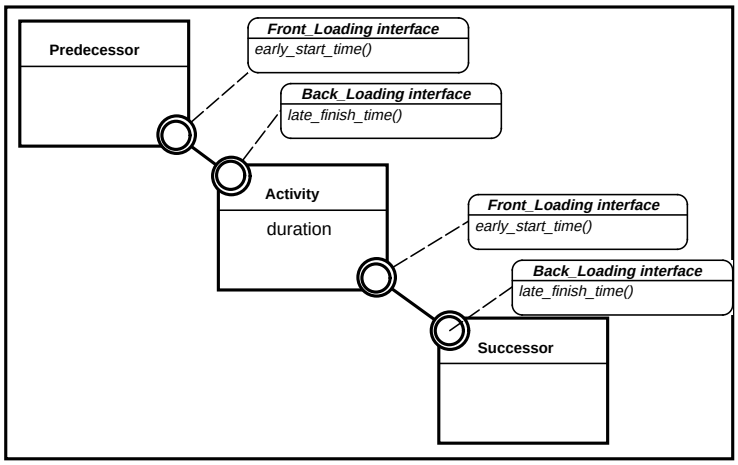
3.5.
Role model
of activity network

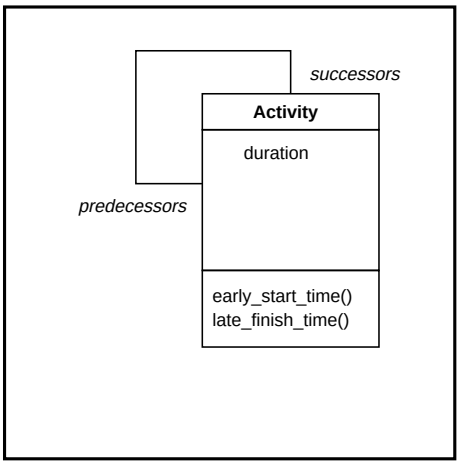


interface **FrontLoading** {
    void early_start_time(in integer receiver_earliest_start_time);
    };

interface **BackLoading** {
    void late_finish_time(in integer receiver_latest_finish_time);
    };

3.6.
Identity-preserving system model





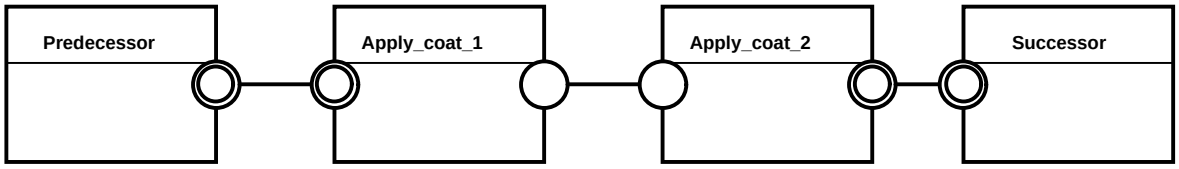
(SC) System-Centered Approach

Role --> object identity
Port --> object reference (variable)

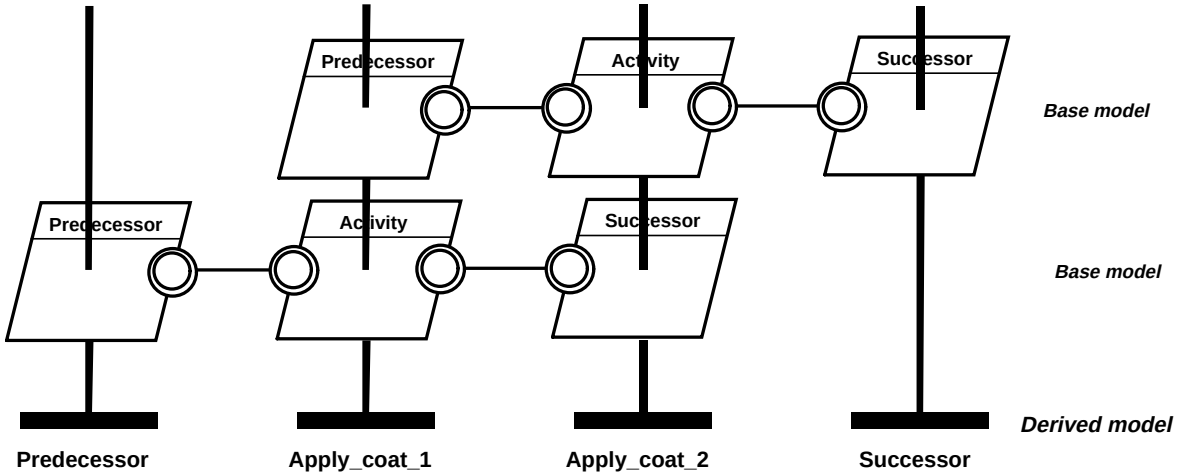
(PC) Program-Centered Approach

Class --> set of objects
Interface --> services offered

3.7.
Role model synthesis: Composite paint subsystem



3.8.
System inheritance
"Hat stand" view of synthesis



4.
Conclusion

1) The model as an object structure

- *Navigate model objects through CORBA*
- *An INCLUSIVE model with common core*

2) System Centric and Program Centric modeling

- *Identity-preserving system models*
- *Interfaces bound to collaboration paths*
- *IDL interfaces first class citizens*
- *System inheritance supplements class inheritance*
- *Seamless transition System <-> Interface <-> Class*
- *Clean extension of object models*

3) Rationale for adding System Centric approach

- *Separation of concern*
- *Capture synergy*
- *Reusable Architectures, Patterns, and Frameworks*
- *Model business objects and processes*

4.1.
The OOram proposal

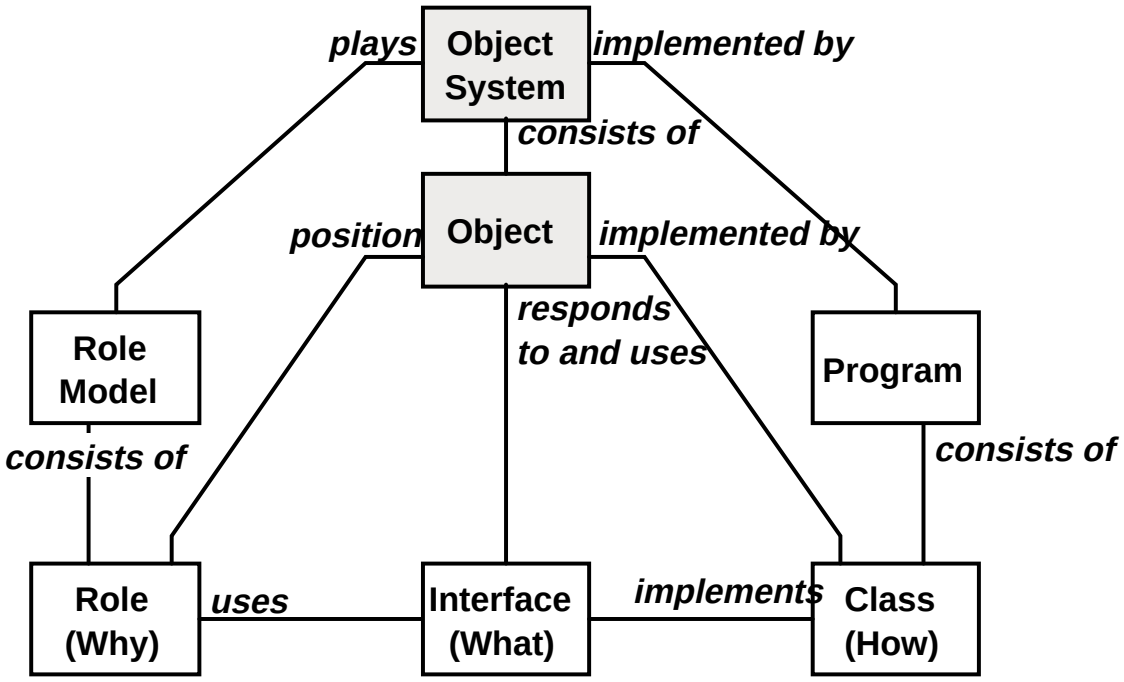


TABLE OF CONTENTS