



# **Component Based Development**

**Leo Hermans (l.hermans@everest.nl)  
Principal Consultant and Knowledge Engineer at Everest**

---



# Architecture vs. City Planning

Architecture



City Planning



Through 2003, enterprises that apply architecture-level design patterns instead of city-planning design patterns to the challenge of application integration will be burdened with inflexible, costly and unmanageable IS portfolios (0.9 probability). *Gartner, IT Expo, 1999*





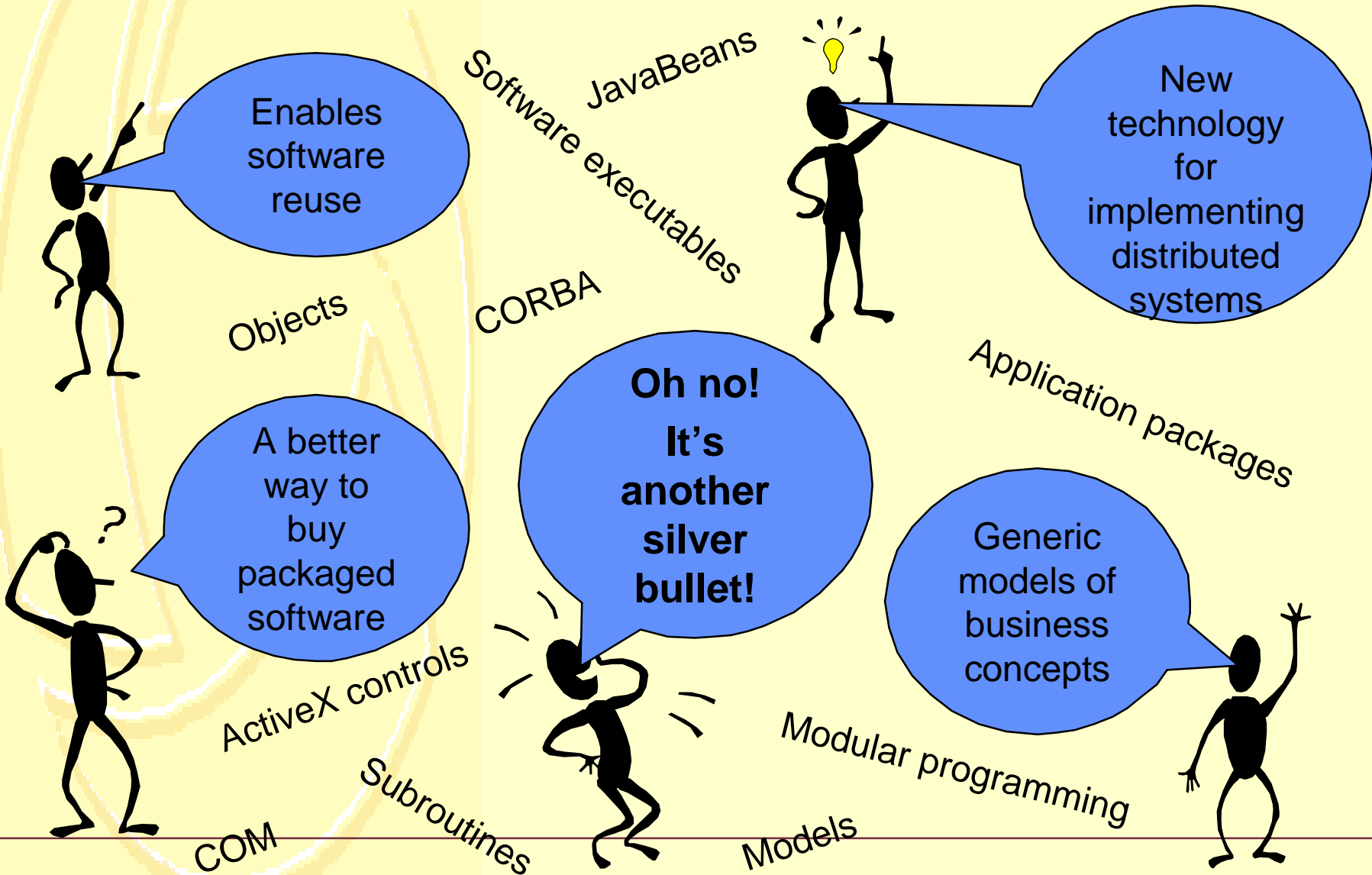
# Component Based Development

- ❖ what are Components
- ❖ Components and Services
- ❖ Separation of the What from the How





# But what are Components ?



# CBD can be compared to Sex

- ❖ **Everybody talks about it**
- ❖ **Everybody does it (and already for a long time)**
- ❖ **Paradise on earth**
- ❖ **Overestimation of the importance of tools**
- ❖ **Later-on a lot more appears to be important (organization)**
- ❖ **Once you made your choice, you will appear to be stuck for a long time**



# Components and Services

## Services

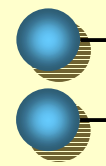
Let me check if we have it in stock

We can deliver it on Monday

Your order number is 2321344

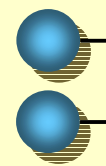


Product Availability



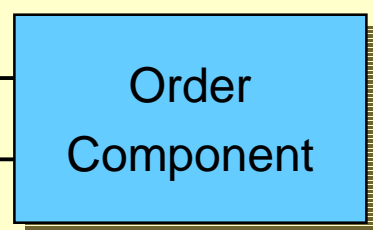
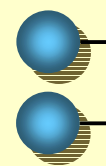
Product Information

Schedule Delivery



Delivery Status

Create Order

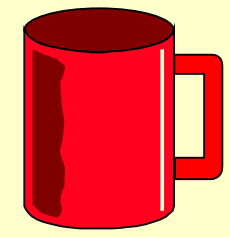


Amend Order

## Interfaces

## Components

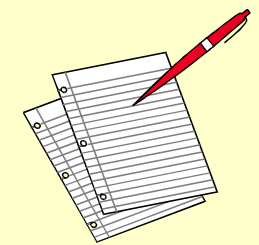
## Business Objects



Product



Delivery

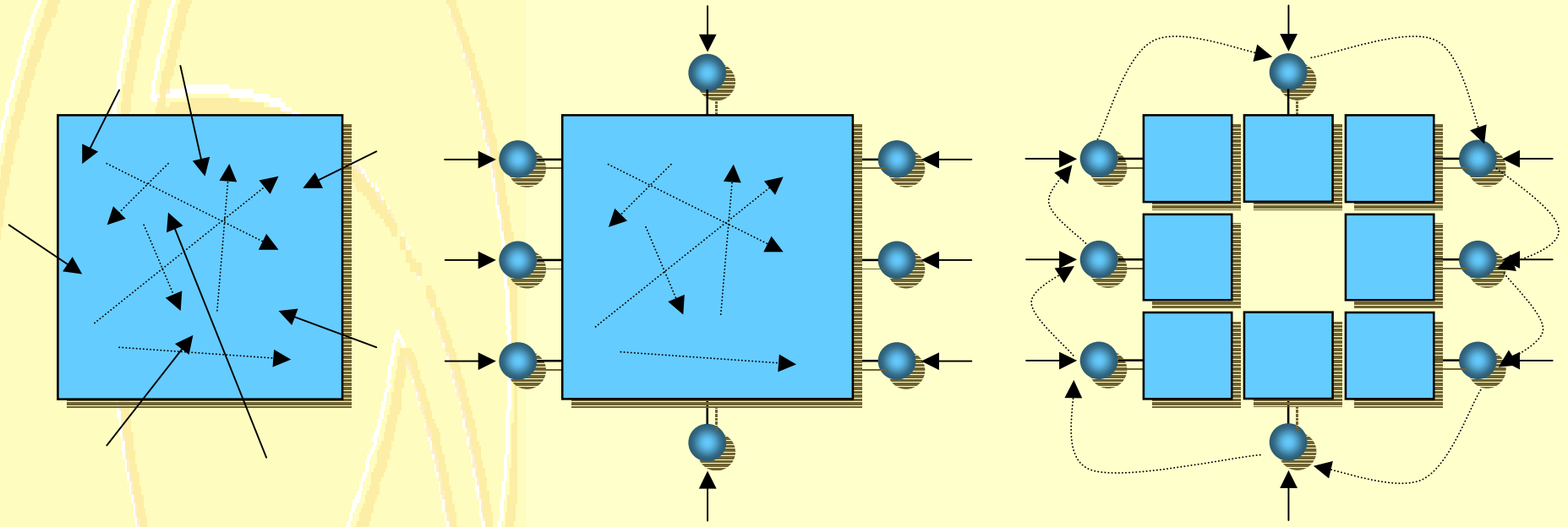


Order





# Component-Based Systems Vs Monolithic Systems need to enable transition



Monolithic Legacy System

Wrap Monolith with Interfaces

Replace Monolith with Components, retaining Interfaces

- ▶ *Internal Dependencies*
- ▶ *External Dependencies*



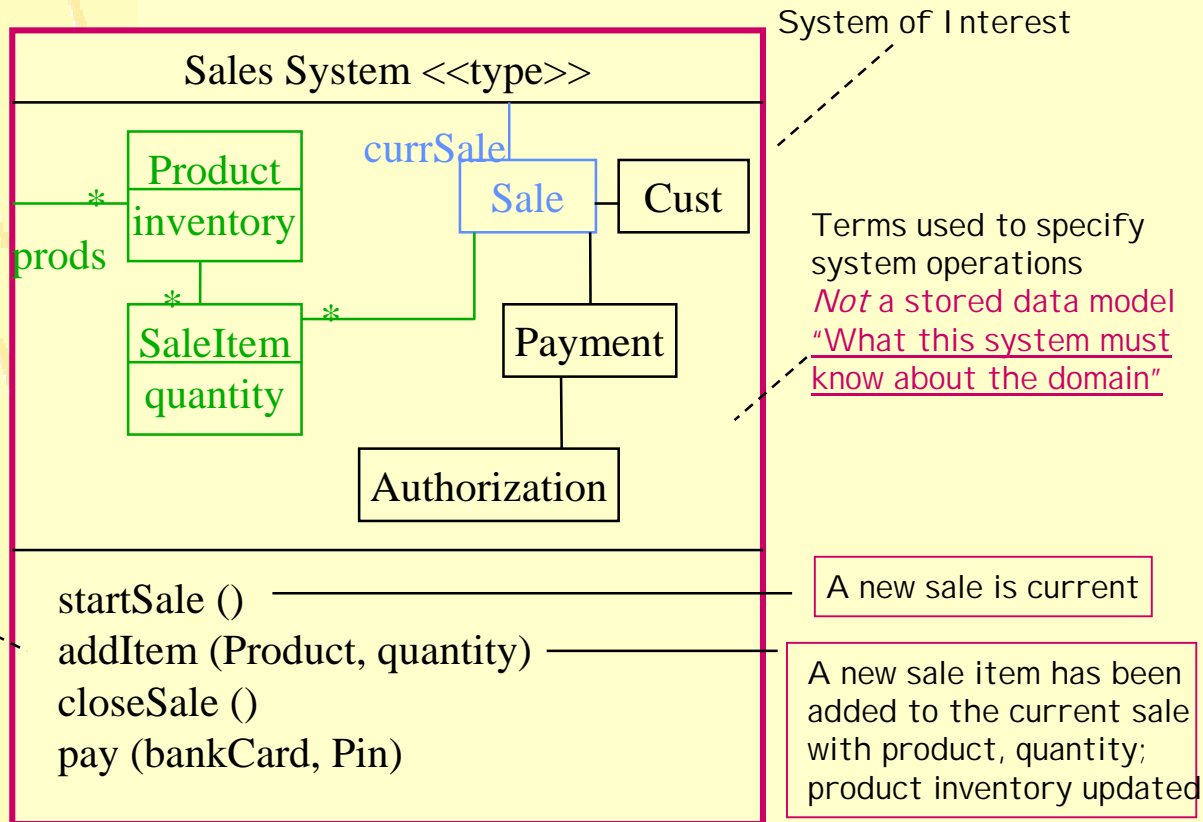
# Separation of the *what* from the *how* precision in service specification

Type name

Type model

Type interface

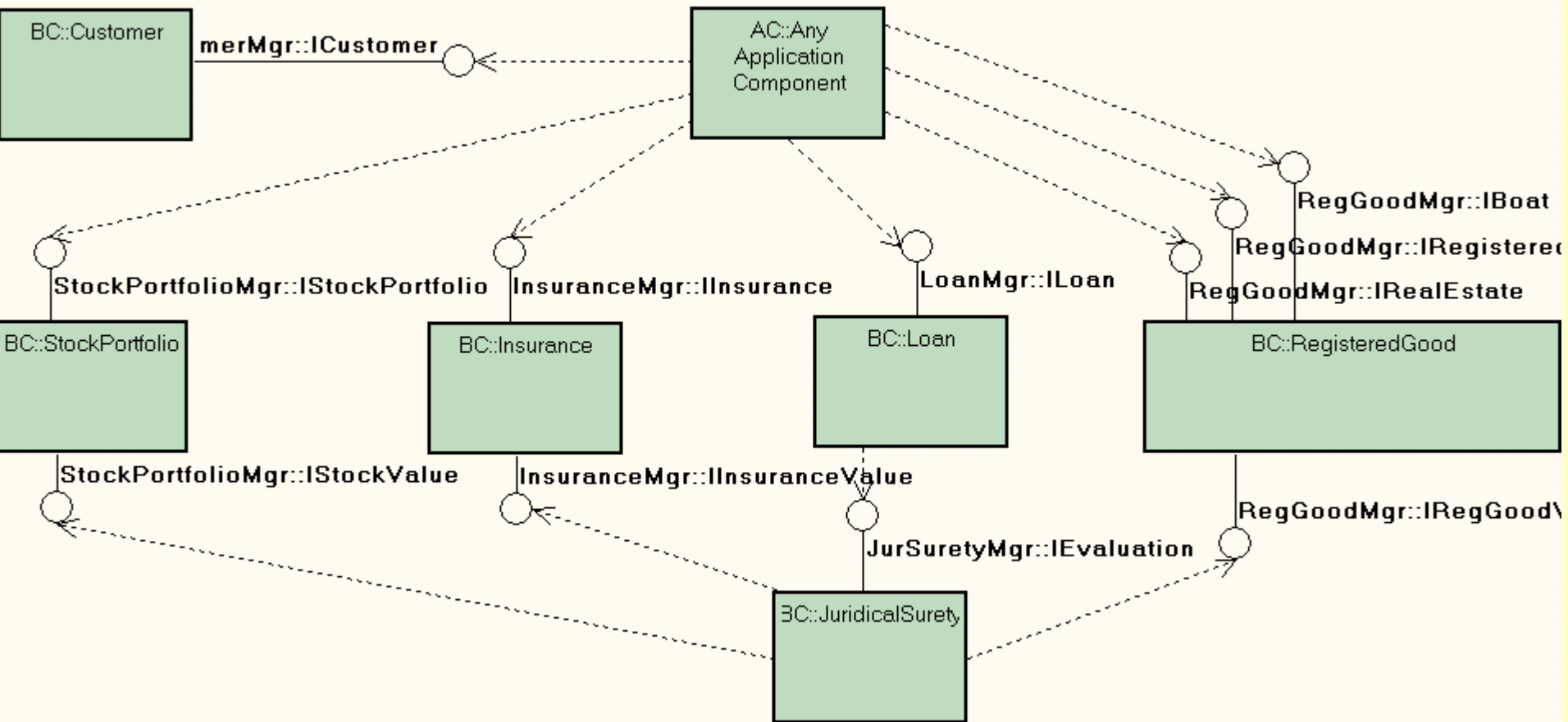
Interface Operations  
of System



Note: Behavior Specs can be made precise using UML/Object Constraint Language (OCL)



# Example of an Initial Application/Component Architecture



# Example of an Initial Application/Component Architecture Features

- ❖ **Application Decomposition**
- ❖ **Components, Interfaces and Operation**
  - ▲ Application
  - ▲ Business
  - ▲ Infrastructure
  - ▲ External
- ❖ **Operation Consumption**
  - ▲ by applications
  - ▲ by other component operations (component dependencies)
- ❖ **Cross Component Associations and RI Solutions**
- ❖ **Traceability links with Functional Requirements**

Tooling?

Demonstration





# Knowledge Engineering and OO/CBD

## ❖ CA Approach

- System Context Model
- Type Specification Model
- Refinement of Type Specification Model
- Refined Type Specification Model
- Model frameworks

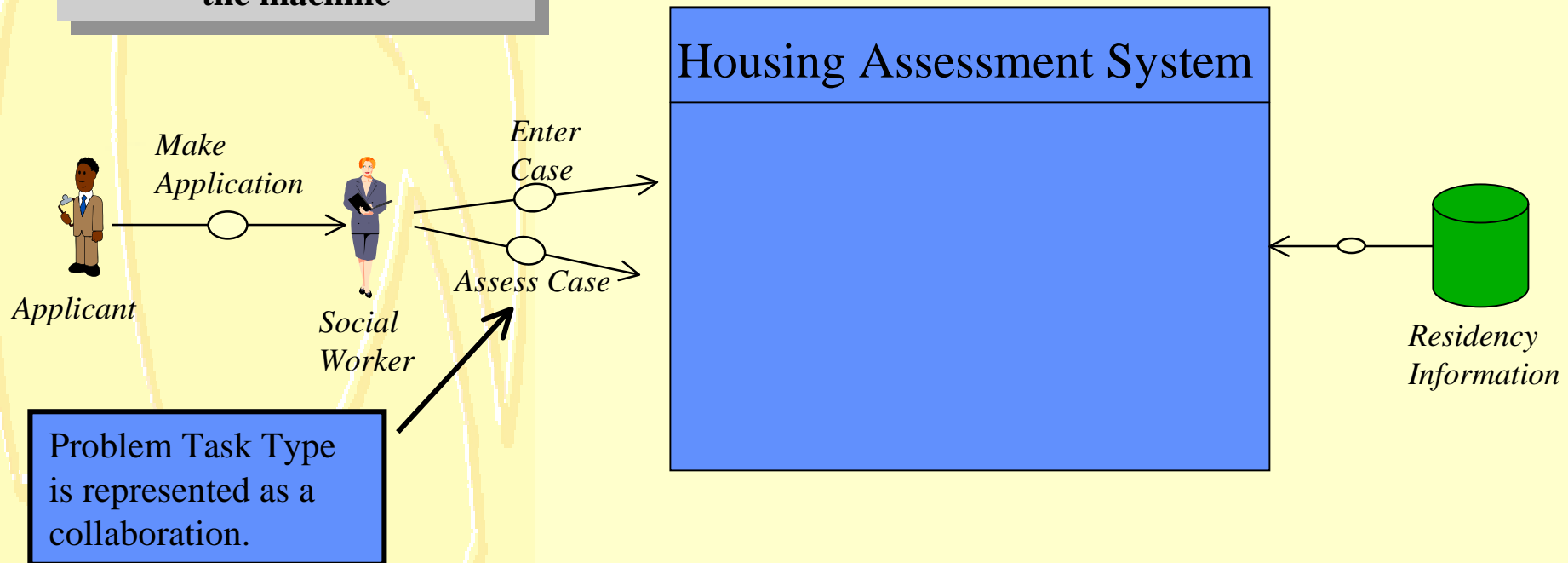
## ❖ SDF-II Approach

- System Context Model
- Type Specification Model
- Refinement of Type Specification Model
- Refined Type Specification Model
- Model frameworks



# System Context Model

The customer/user is usually interested in effects that are felt some distance from the machine



- Collaborations are precisely defined by pre and post conditions.

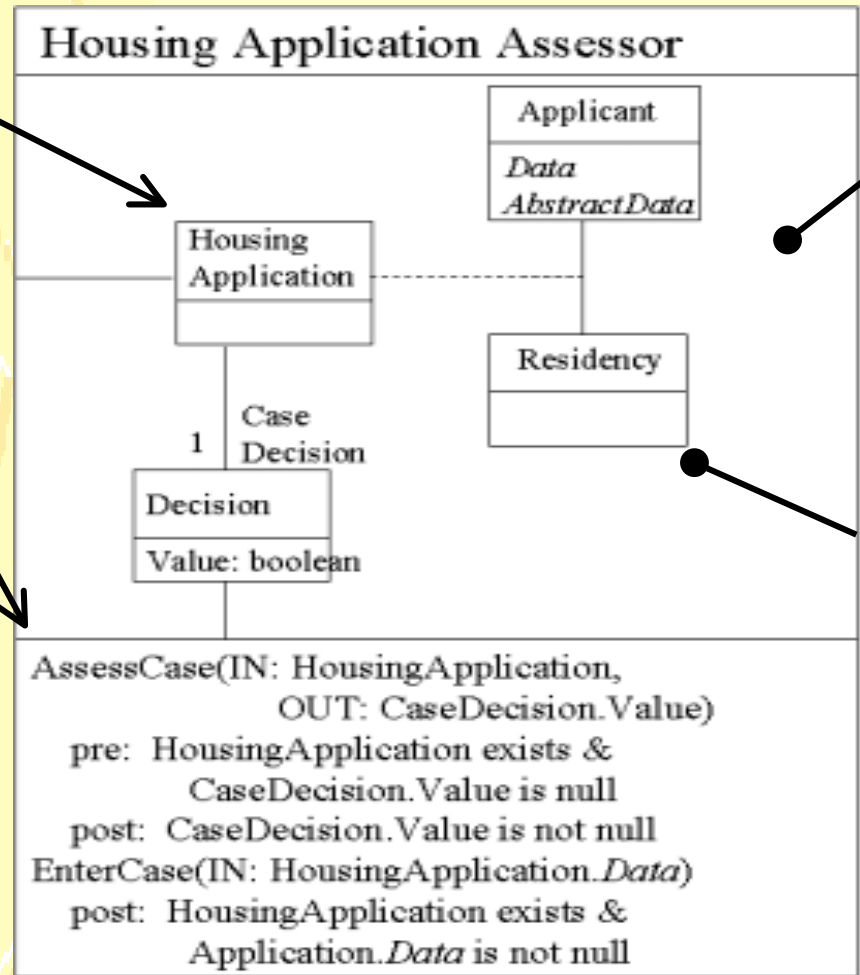




# Type Specification Model

The Housing Application is an Association Type

Problem task type is an operation.



Type Model represents semantics of pre- and postconditions

Note that the type model doesn't show Criteria. Criteria are not part of the external semantics.

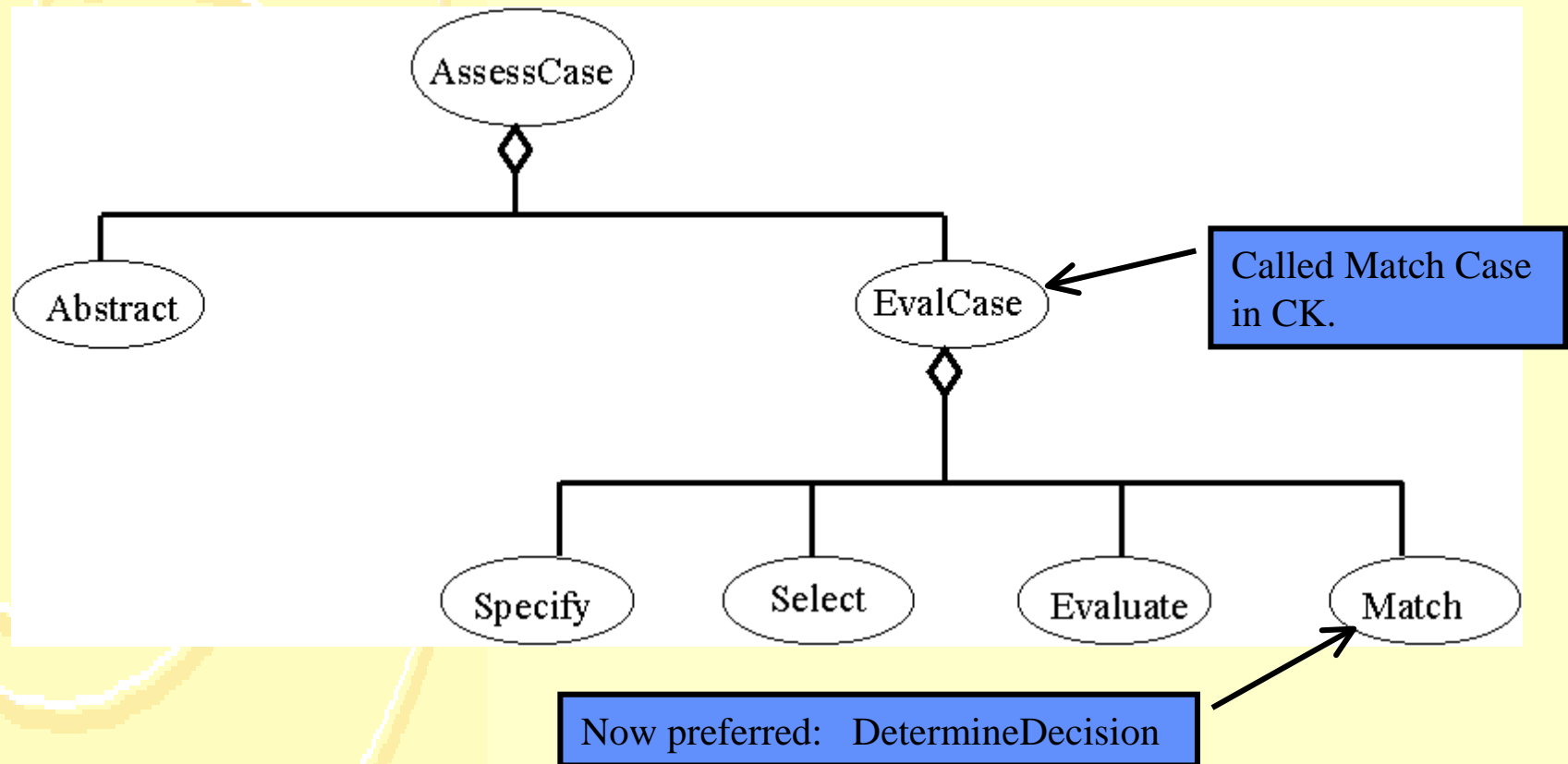


# Need to Refine the Type Specification

- ❖ **Knowledge Engineering: Problem task type decomposition = Constructing *internal* logical structure of the problem solving process.**
  - ▶ Knowledge engineering is more theoretical than decomposing external actions, e.g., buying coffee.
- ❖ **Refined Type Specification:**
  - ▶ Internal steps of AssessCase are shown as *private* operations.
  - ▶ Type Model reveals “internal types” used in reasoning, e.g., Norms.
- ❖ **Knowledge Level Analysis = Refinement Rules**



# Assessment Action Decomposition (CommonKADS)



# Action Specification for Assessment: Parameterized Attributes in Post Conditions

**Action** Abstract(happl:HousingApplication, Abstracthappl:AbstractHousAppl)  
**post:** abstracthappl.Applicant.AgeCategory = Abstraction(happl.Applicant.Age) and  
abstracthappl.Applicant.HouseholdType = Abstraction(happl.Applicant.HouseholdSize).

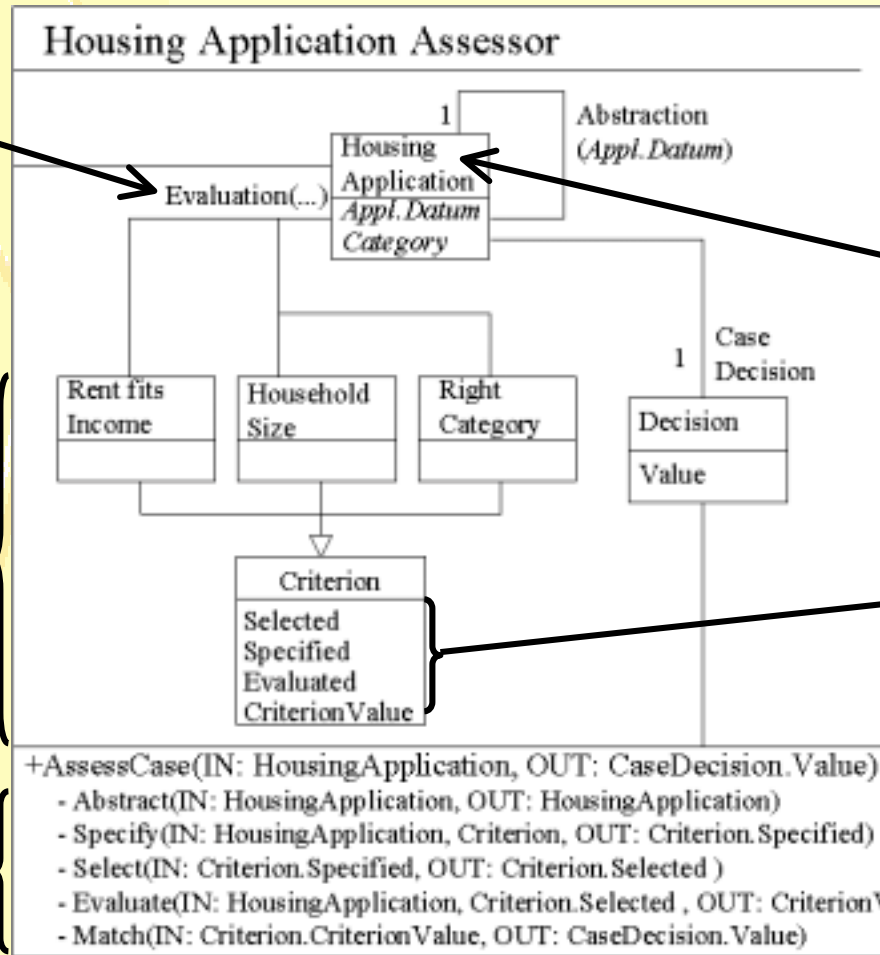
**Action** Evaluate(selectedCriterion:Criterion, abstracthappl:AbstractHousAppl,  
evaluatedCriterion:Criterion)  
**post 1:** RentFitsIncome.Value = Evaluation(abstracthappl.Applicant.Income,  
abstracthappl.Applicant.HouseholdType, abstracthappl.Applicant.AgeCategory,  
abstracthappl.Residency.Rent).  
HouseholdSize.Value = Evaluation(abstracthappl.Applicant.HouseholdType,  
abstracthappl.Applicant.AgeCategory, abstracthappl.Residency.Type).  
RightCategory.Value = Evaluation(...etc...)  
**post 2:** *After evaluation, the selected Criterion becomes an evaluated Criterion.*







# Refined Type Specification Model



Better if defined for each Criterion.

For simplicity, collapse representation of Housing Application

Internal Types

State Attributes

Internal Operations



# Model Frameworks

## ❖ What are Model Frameworks?

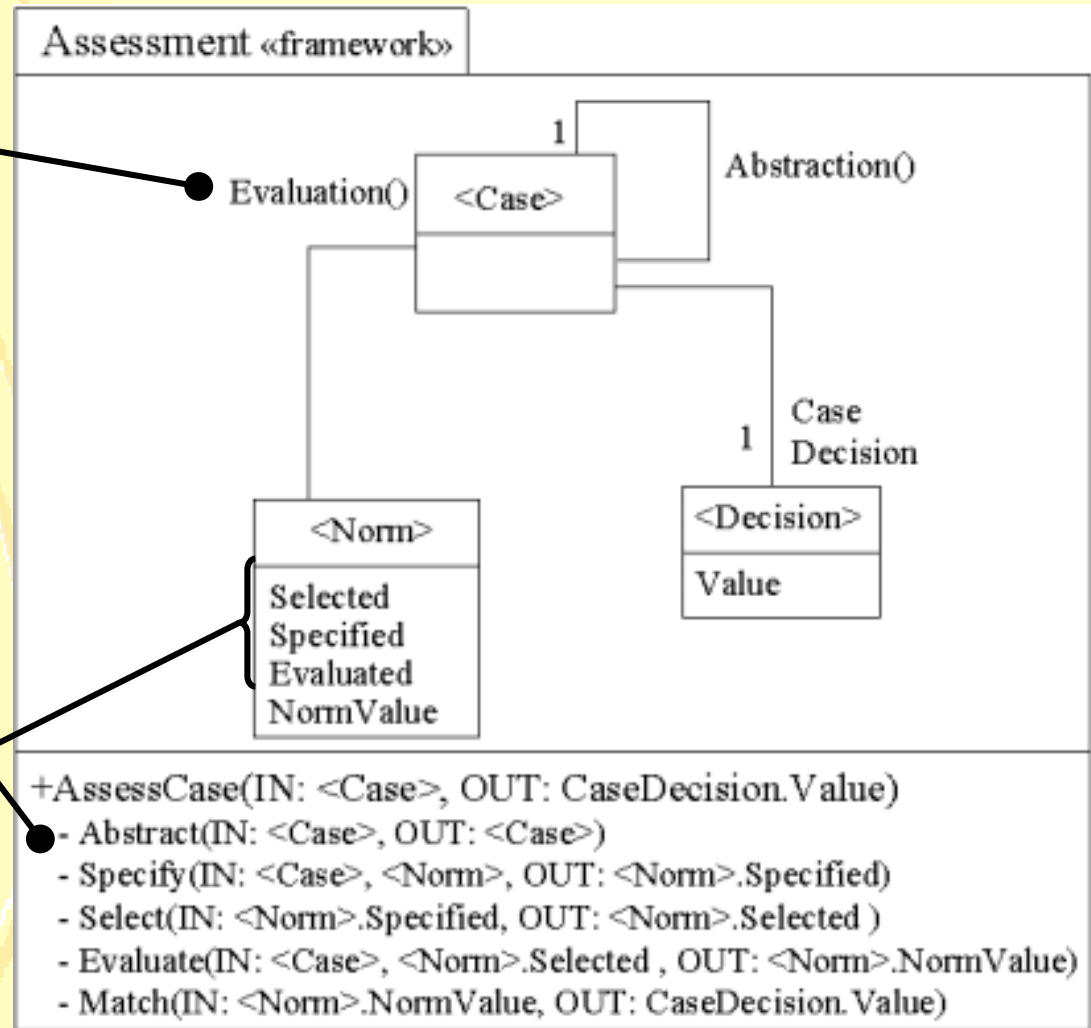
- ✦ A framework for building *models*, e.g., a (Refined) Type Specification Model.
- ✦ Model frameworks do not exist in code, but are packages that can be imported into models to provide basic vocabulary and structure for domain *types*.
- ✦ See Chapter 9 of the Catalysis book.

## ❖ Model frameworks contain placeholders, e.g. <Case>, which are instantiated through substitutions of domain types.

## ❖ Model frameworks are Type Specifications of the *generic* vocabulary of the problem task type!



# General Structure of a Model Framework (abstracted from the Housing application)



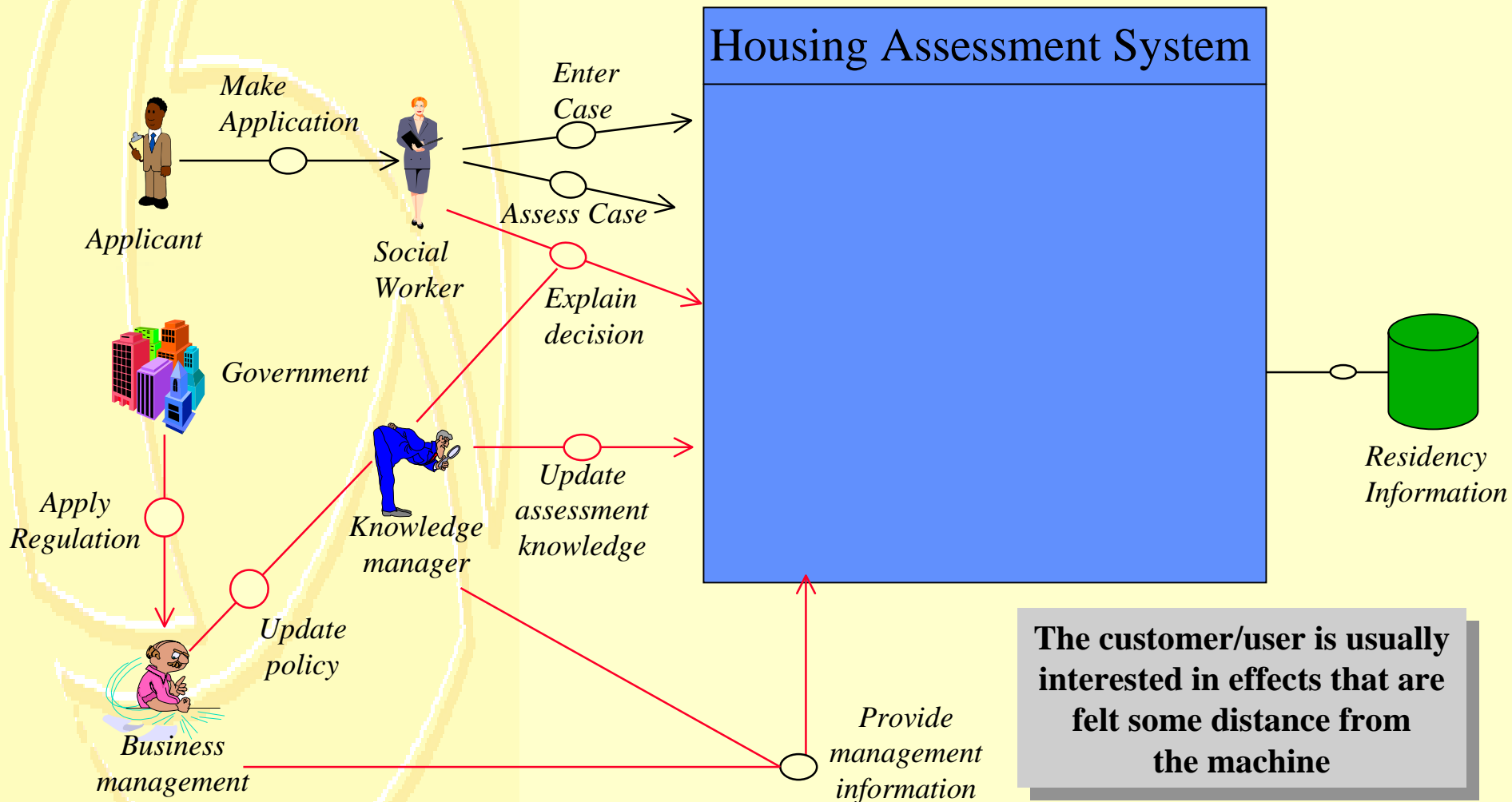
Parameterized attributes are insufficient to show reasoning structure of Assessment.

A Model Framework addresses the Refined Type Specification

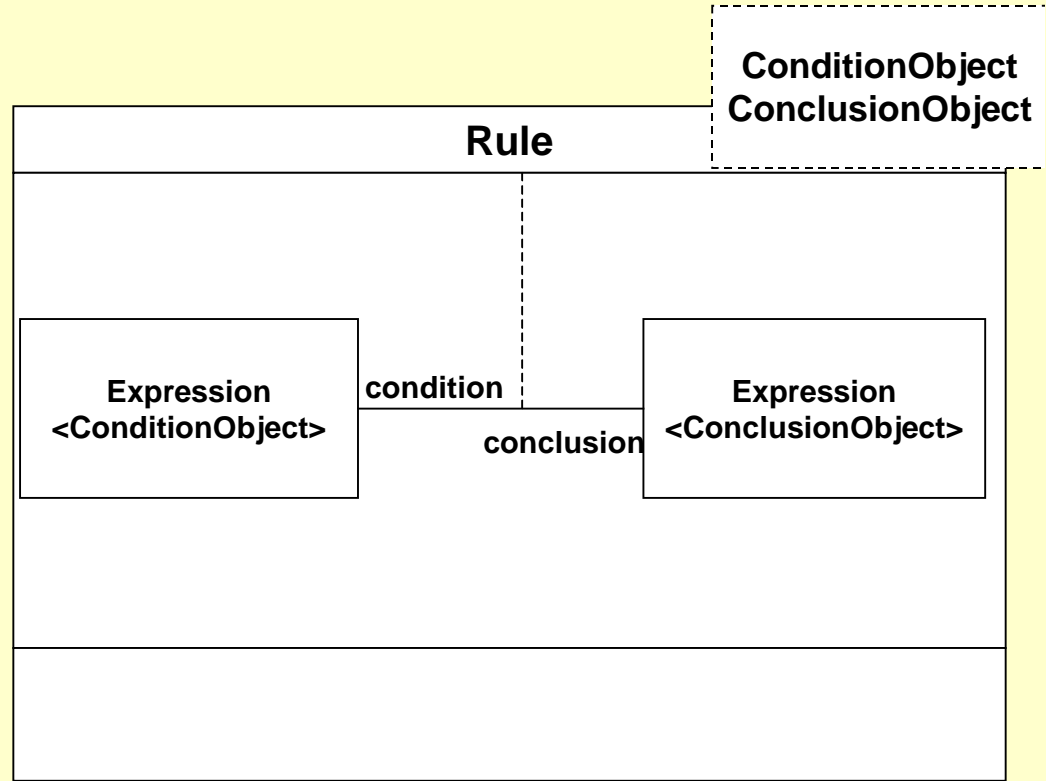
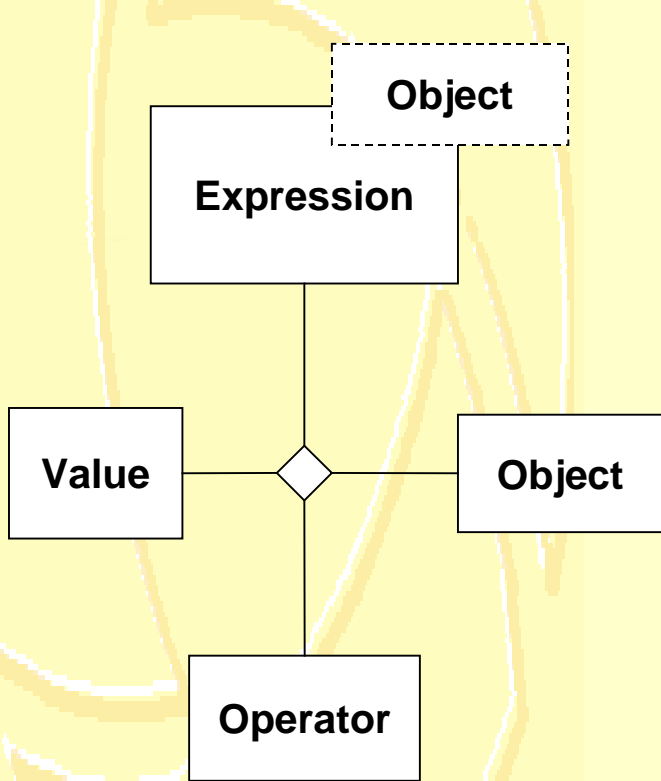




# System Context Model (II)



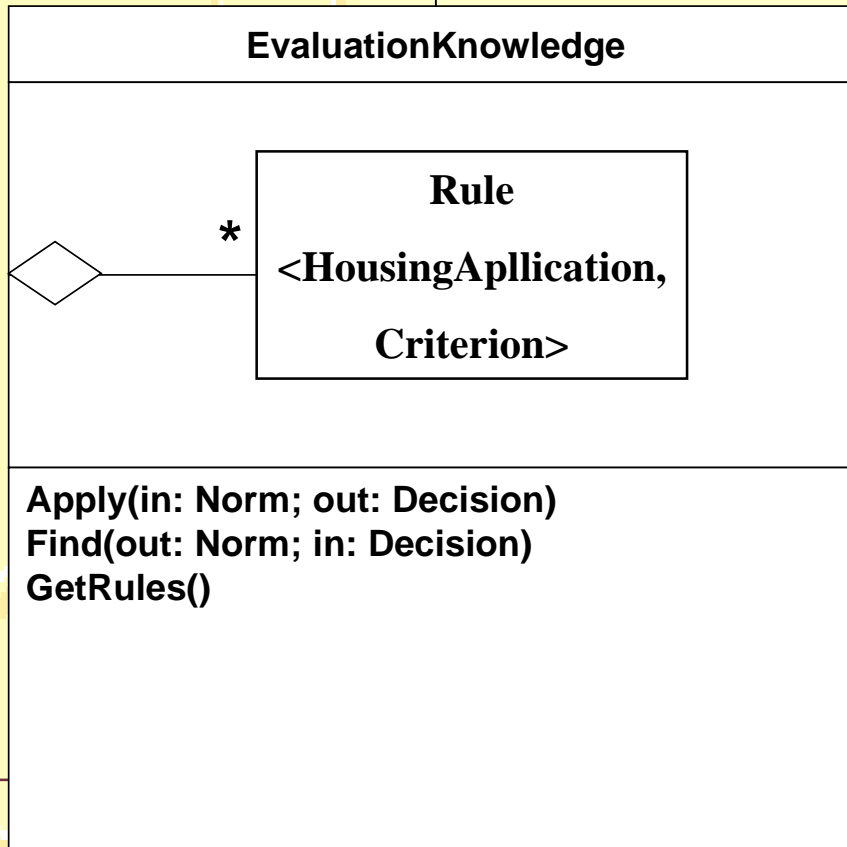
# Pattern for expressing rules



# Application of rule pattern

**Housing  
Application**

**Criterion**



**Clear separation between the form of knowledge, expressed in its knowledge type and its content, expressed in actual rules or tables.**

**Knowledge manager is responsible for updating the content, not the structure.**





# Application of assessment pattern

