HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

SOICT

School of Information and Communication Technology

ONE LOVE. ONE FUTURE.

TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

IT3180 – Introduction to Software Engineering

9 – Models for requirements

ONE LOVE. ONE FUTURE.

Models for requirements

Requirement analysis and specification includes selecting the appropriate tool for the particular task

- Models provide a bridge between the client's understanding and the developers'
- A variety of tools and techniques
- There is no correct technique that fits all situations

Models

A model is a simplification of reality

- We build models so that we can better understand the system we are developing

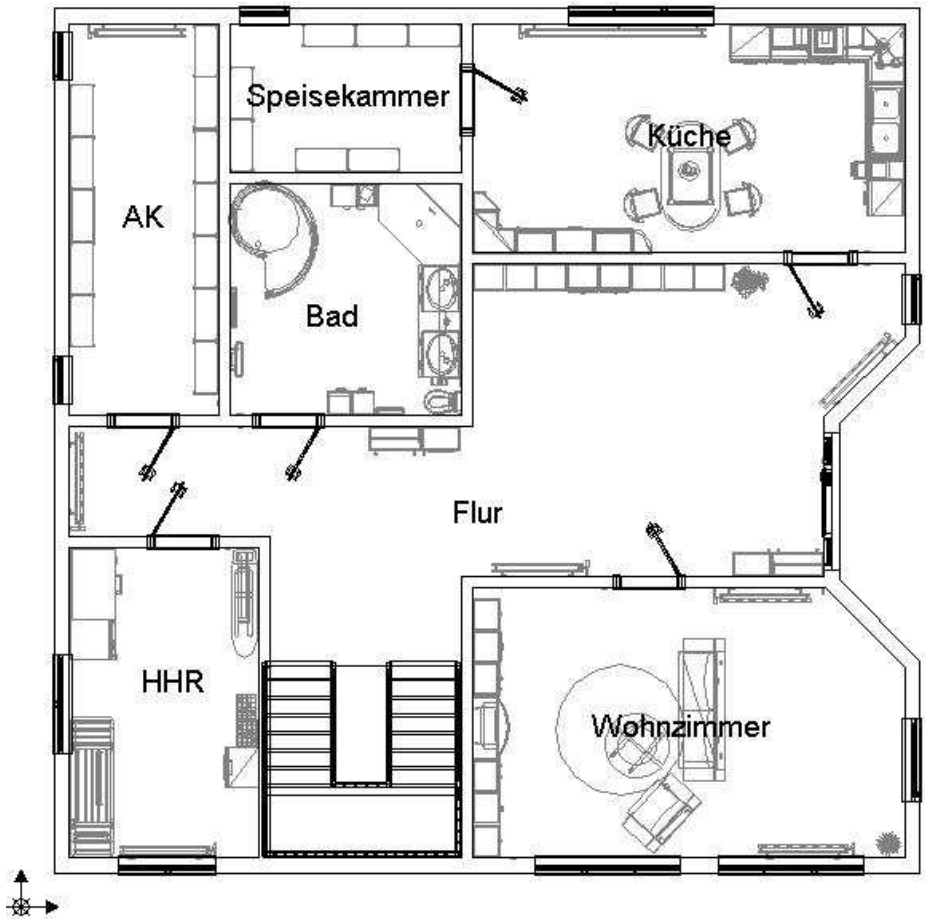
- We build models of complex system because we cannot comprehend such a system in its entirely

Example: Model as a Blueprint

1. Requirements

- Shall fit on given piece of land.
- Each room shall have a door.
- Furniture shall fit into living room.
- Bathroom shall have a window.
- Cost shall be in budget.

2. Design

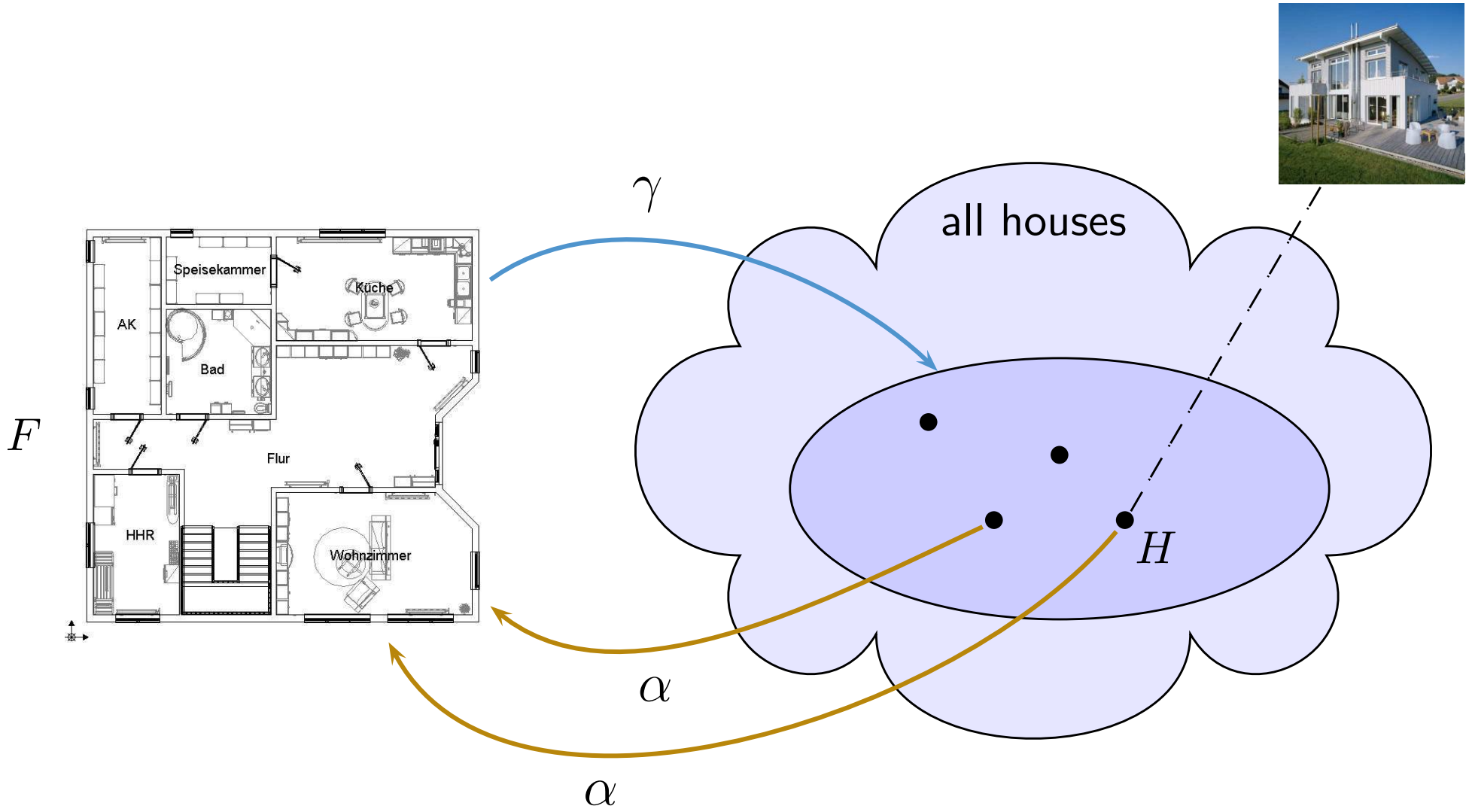


http://wikimedia.org (CC nc-sa 3.0, Ottoklages)

3. System

http://wikimedia.org (CC nc-sa 3.0, Bobthebuilder82)

Example (2): Model as a Blueprint



Principles of Modeling

- **The choice of what model to be created has a profound influence on how to resolve a problem**
- No single model is sufficient
- Every model can be expressed at **different levels of precision**
- Good models are connected to reality

- Every nontrivial system is best approached through a small set of nealy independent models

The Unified Modeling Language

- **UML is a standard language for modeling software systems**
  - Serves as a bridge between the requirements and the implementation
  - Provides a means to specify and document the design of a software system
  - It is intended to be processed and programming language independent, but is particularly suited to object-oriented program development

Data Flow Models (Data Flow Diagrams – DFDs)

- **Goal**
  - Represent the flow of information through the system and the activities that process this information
- DFDs provide a graphical representation of the system that aims to be accessible to computer specialist and non-specialist users
- The models enable software engineers, customers and users to work together effectively during the analysis and specification of requirements

DFD notations

- **Processes**
  - The activities carried out by the system which use and transform information
  - Process is denoted as a **rounded rectangle**
- **Data-flows**
  - The data inputs to and outputs from to these activities (**processes**)
  - Data flows are notated as **named arrows**
- **External entities**
  - The sources from which information flows into the system and the recipients of information leaving the system
  - External entities are notated as **ovals**
- **Data stores**
  - Where information is stored within the system
  - Notated as rectangles

DFD symbols

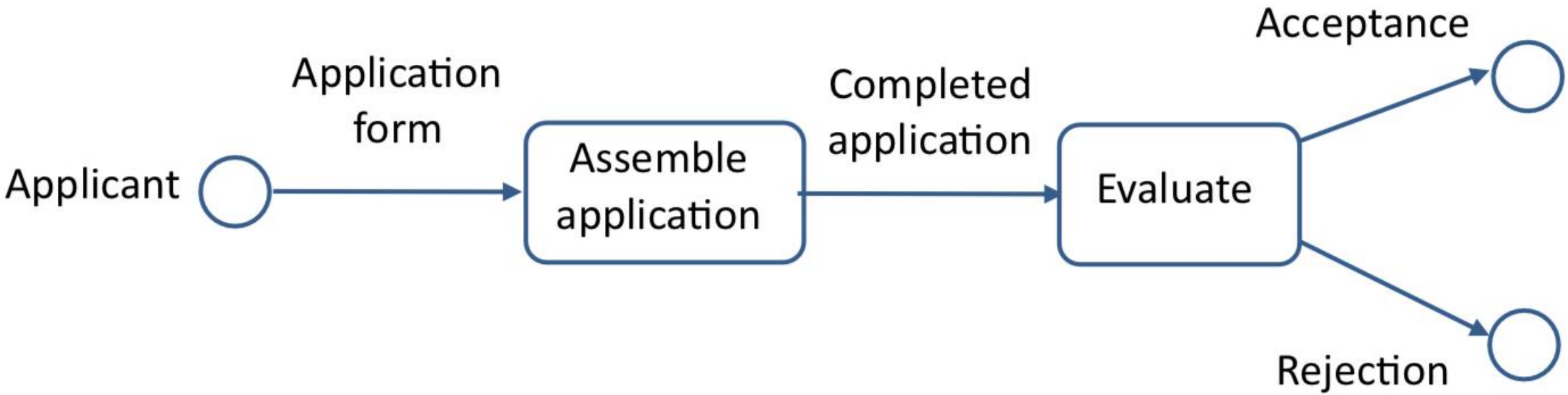
External entities

Processing steps

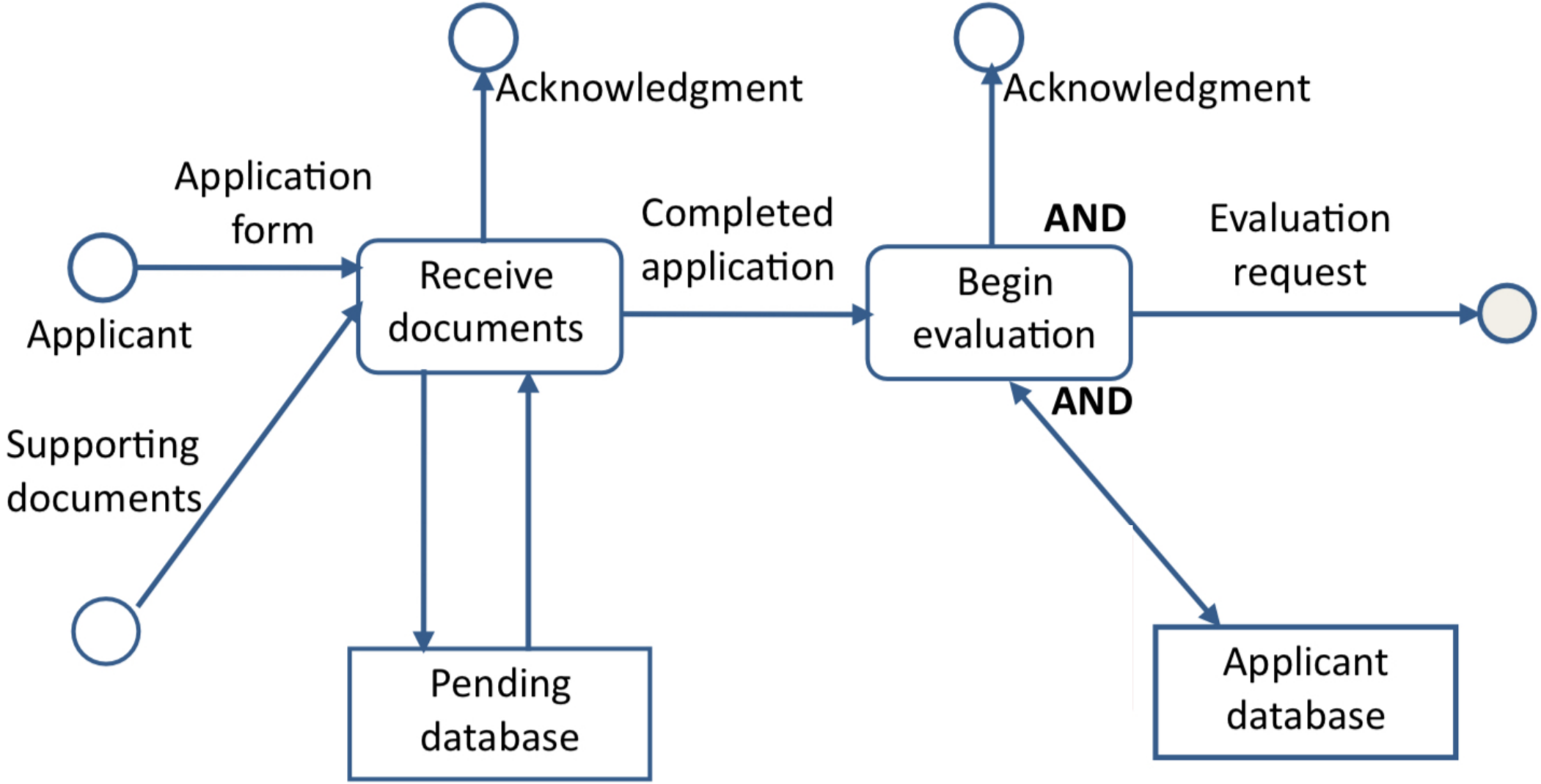
Data stores or sources

Data flows

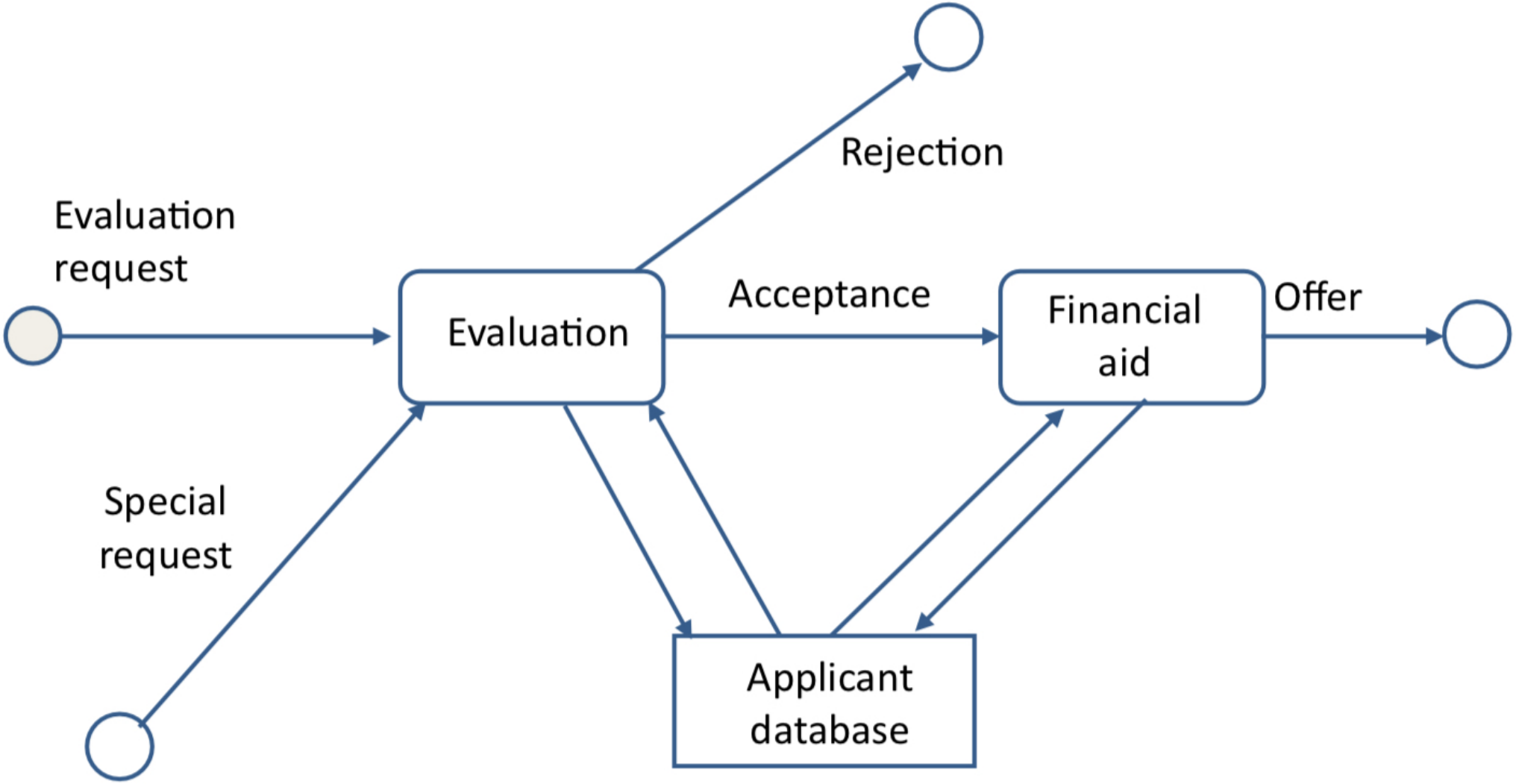
Example: University Admissions



Example (2): Assemble Application



Example (3): Process Completed Application



Decision Table Model

University Admission Decision

SAT > S1	T	F	F	F	F	F
GPA > G1	-	T	F	F	F	F
SAT between S1 and S2	-	-	T	T	F	F
GPA between G1 and G2	-	-	T	F	T	F
Accept	*X*	*X*	*X*			
Reject				*X*	*X*	*X*

Each column is a separate decision case. The columns are processed from left to right.

Note that the rules are specific and testable.

Flowchart Models

An informal modeling technique to show the **logic part** of a system and **paths** that data takes through a system

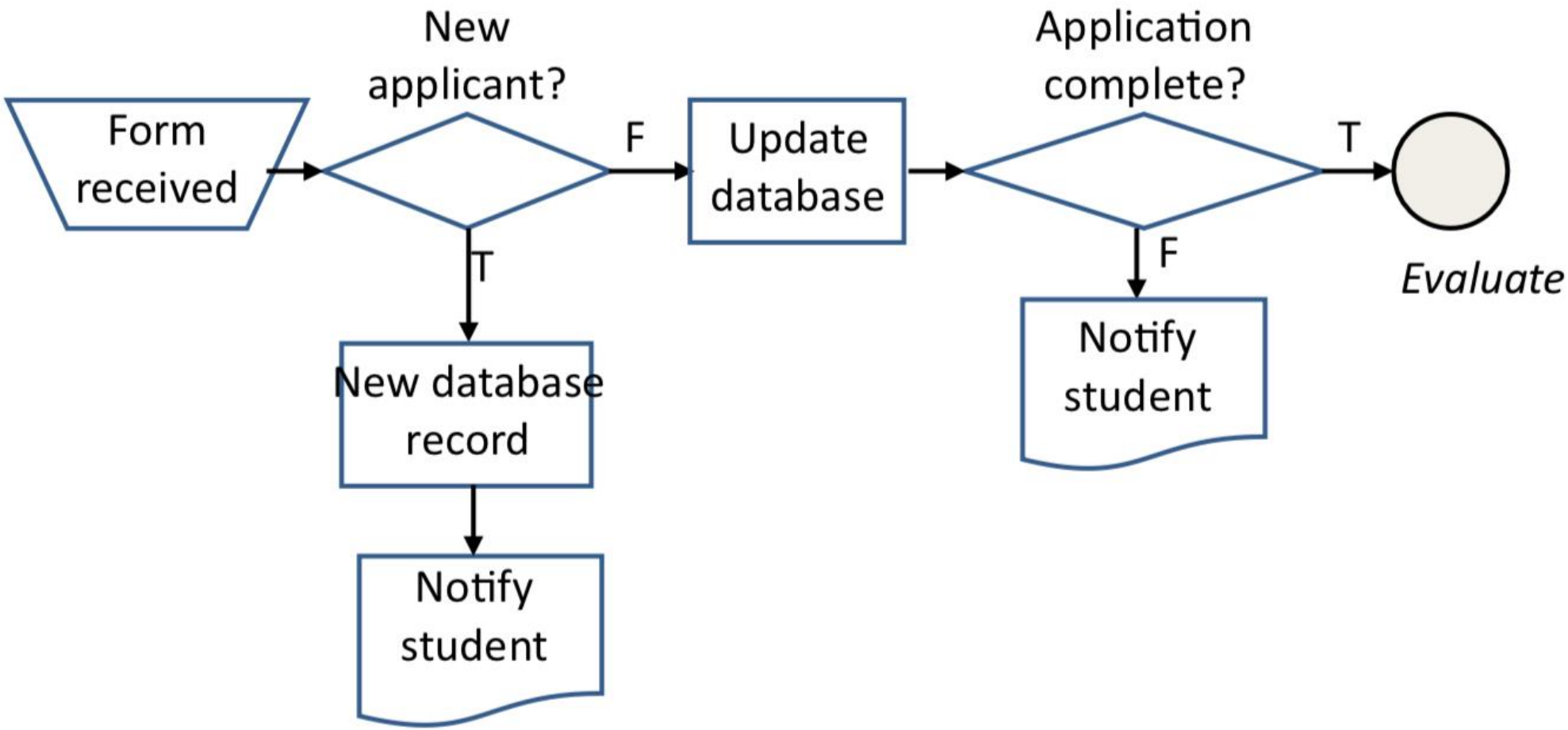
Operation

Decision

Manual operation

Report

Example: University Admission Assemble Application



Transition Diagrams

A system is modeled as a set of **states**, S_i

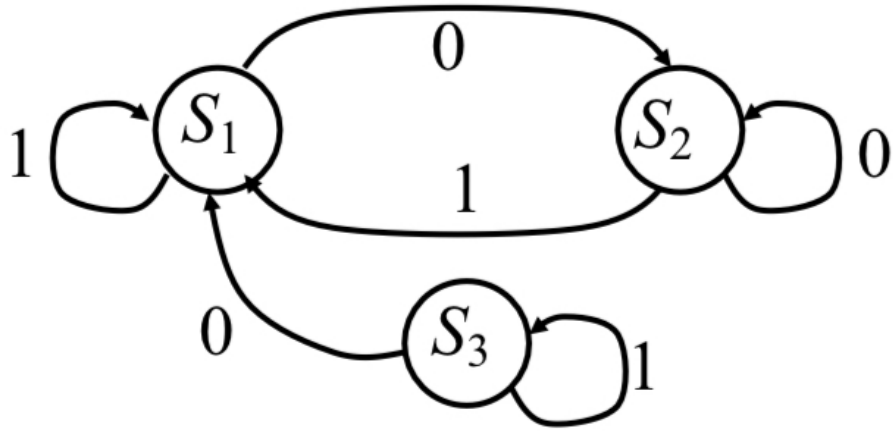
A **transition** is a change from one state to another.

The occurrence of a **condition**, C_i, causes the transition from one state to another

Transition function:

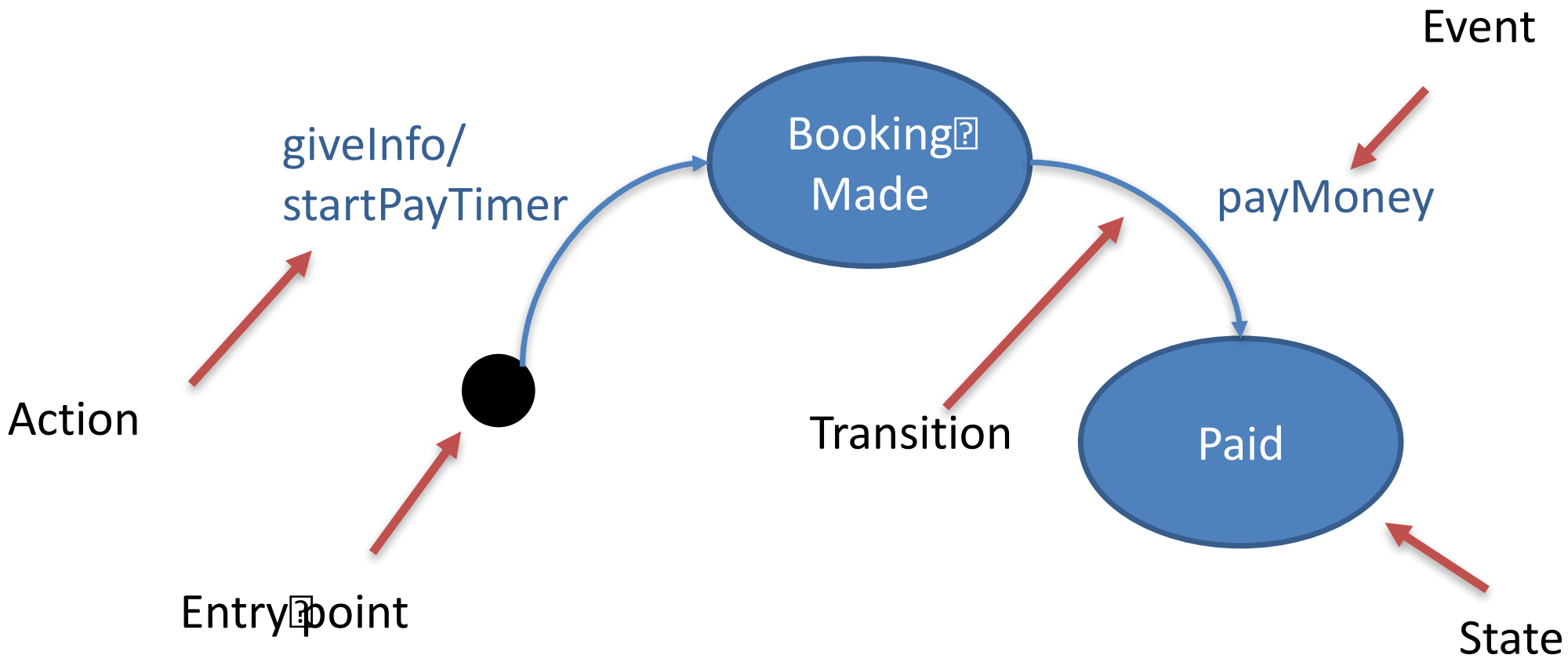
$$f(S_i, C_j) = S_k$$

Example

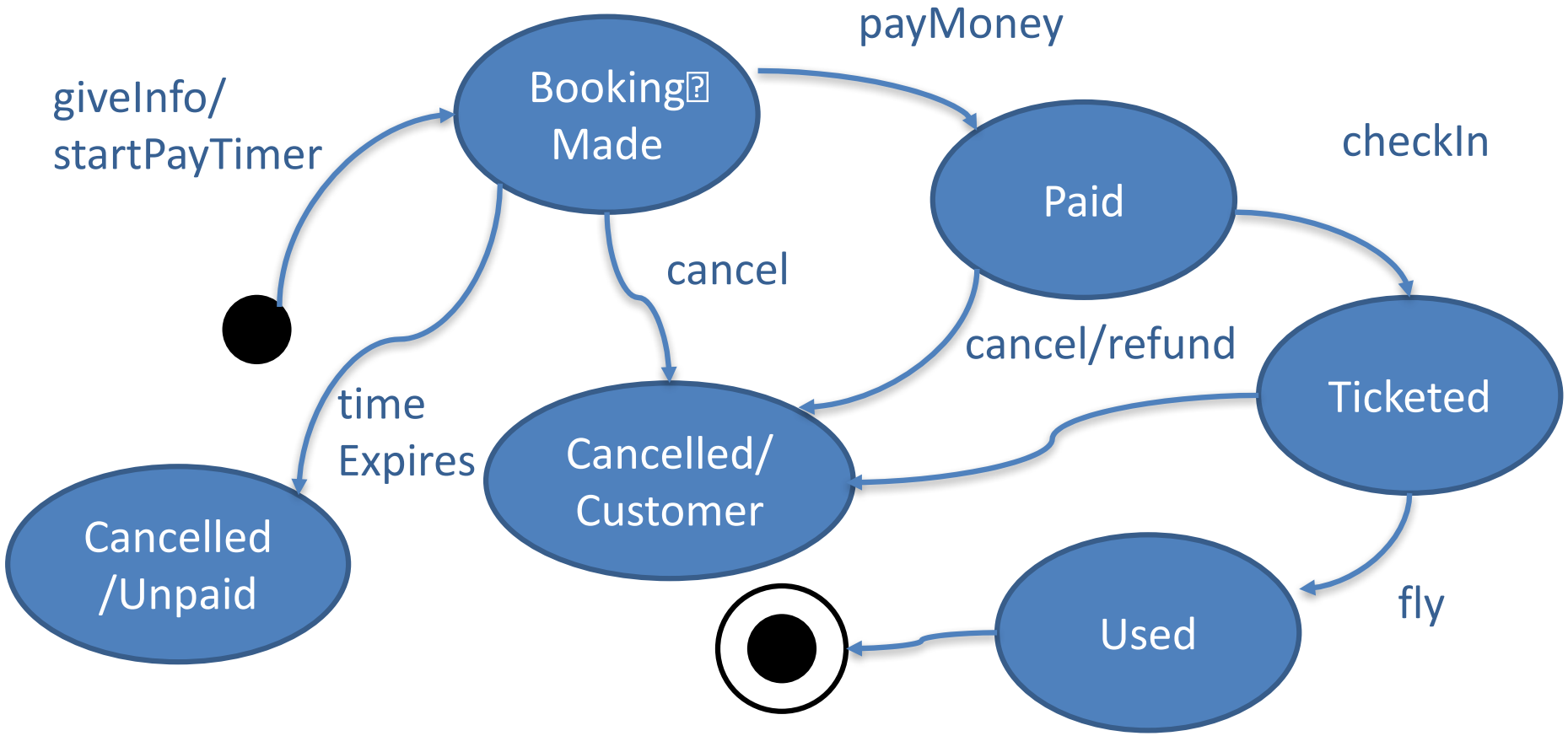


Example: Flight Booking System

Finite State Machine



Example (2) Finite State Machine for Flight Booking System



Example (3): Place Order

Description:

- To place an order, an user has to log in to the system. An order once submitted to the system is in the state "***new order***", the payment can be delayed for 2 hours after submission, after two hours, if the payment is not successful, the order will be automatically rejected. Once successfully paid, the order will be passed to be prepared. The store can also cancel order if there is one item not available. After being prepared, the order will be passed to the delivery. It can be delivering and delivered when the customers receive their order. Once received, the order is completed.
- The customer can also cancel order after submited or paid but he or she cannot cancel after delivery.

Entity Relation Model

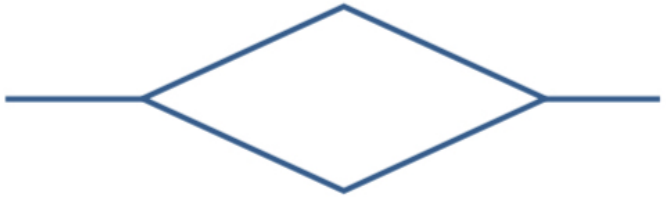
A **requirement** and **design methodology** for relational databases

- A database of entities and relations
- Tools for displaying and manipulating entity-relation diagrams
- Tools for manipulating the database

Entity Relationship Models can be used both for **requirement specification** and for the **design specification**

Entity Relationship Diagram

An entity (noun)

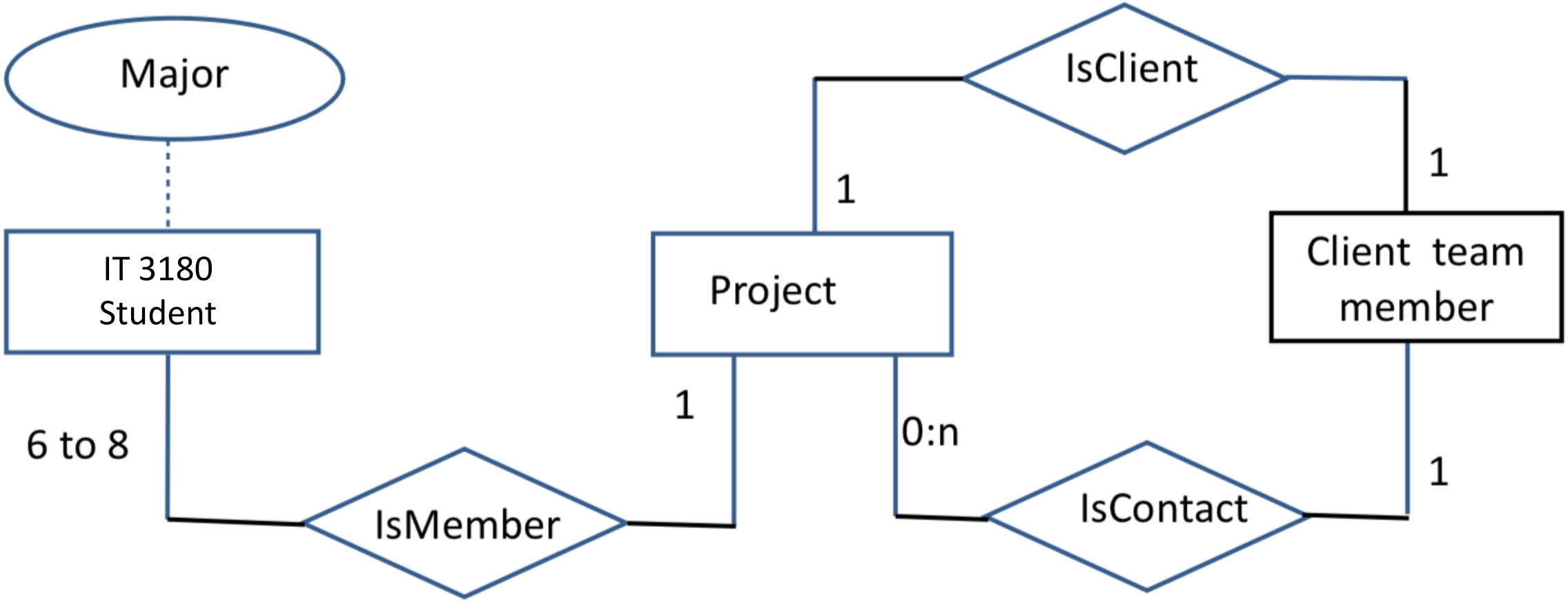


A relation between entities (verb)

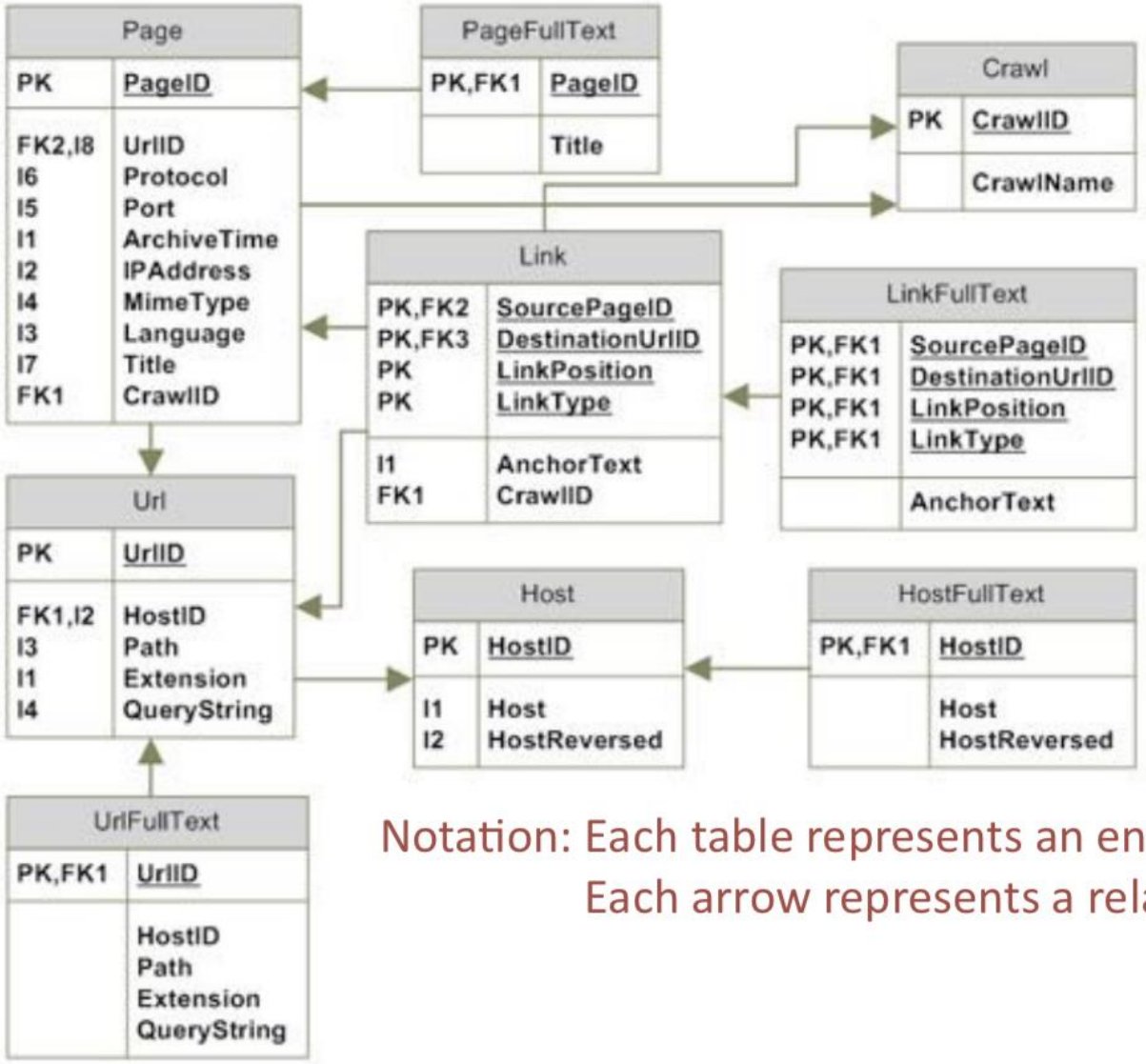
An entity or relation attribute

Note: There are various notations used for entity-relationship diagrams. This is the notation used by Chen (1976).

Example: IT3180 Project



Example: Database Schema for Web Data



Notation: Each table represents an entity
Each arrow represents a relation

Prototyping Requirements

Rapid prototyping is the most comprehensive of all modeling methods

- A method for **specifying requirements** by building a system that **demonstrates** the functionality of key parts of the required system
- Particularly valuable for user interfaces

Discussion

- **Class** and **object** models are used as a tool for **program design**, not for modeling requirements

- Some documents recommend class and object models for requirements definition, but it is **difficult** to use them without **constraining the system design**

- **Flow charts**, **finite state machines**, **entity relationship diagrams** are supported by UML as design models but are equally useful for **requirement modeling**.

9. Models for Requirements

(end of lecture)