Cornell University
Computing and Information Science

CS 5150 Software Engineering

Models for Requirement Analysis and Specification

William Y. Arms

Models for Requirements Analysis and Specification

As you build understanding of the requirements through viewpoint analysis, scenarios, use cases, etc., use **models** to analyze and specify requirements. The models provide a bridge between the client's understanding and the developers'.

The craft of requirements analysis and specification includes selecting the appropriate tool for the particular task.

- A variety of tools and techniques.
- Many familiar from other courses.
- No correct technique that fits all situations.

Models: Useful Texts

Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language.* Addison-Wesley 1999.

The next few slides are based on the approach taken in this book (BRJ).

Grady Booch, et al., *Object-Oriented Analysis and Design with Applications*, third edition. Benjamin/Cummings 2007.

Rob Pooley, Perdita Stevens, *Using UML Software Engineering with Objects and Components*, second edition. Addison-Wesley 2005.

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 entirety.

Models can be informal or formal. The more complex the project the more valuable a formal model becomes.

BRJ

Principles of Modeling

- The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
- **No single model is sufficient.** Every nontrivial system is best approached through a small set of nearly independent models.
- Every model can be expressed at different levels of precision.
- Good models are connected to reality.

BRJ

The Unified Modeling Language

UML is a standard language for modeling software systems

- Serves as a bridge between the requirements specification and the implementation.
- Provides a means to specify and document the design of a software system.
- Is process and programming language independent.
- Is particularly suited to object-oriented program development.

Rational Rose

Rational Rose is an IBM-owned system for creating and managing UML models (diagrams and specifications).

It is available on computers in the Computer Science MEng Lab.

Models: Diagrams and Specification in UML

In UML, a **model** consists of a **diagram** and a **specification**.

A **diagram** is the graphical representation of a set of elements, usually rendered as a connected graph of vertices (things) and arcs (relationships).

Each diagram is supported by technical **documentation** that **specifies** in more detail the model represented by the diagram.

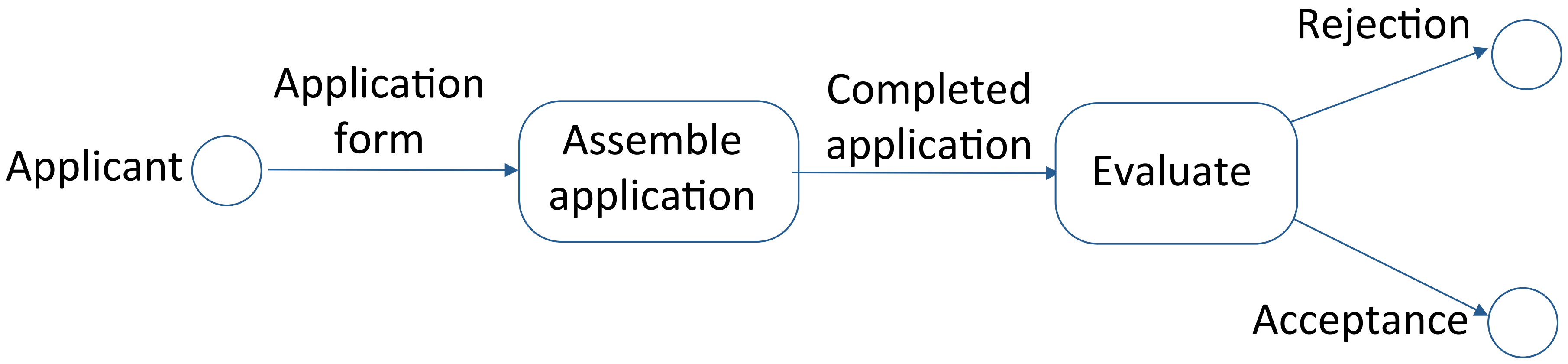
A diagram without a specification is of little value.

Data-Flow Models

An informal modeling technique to show the flow of data through a system.

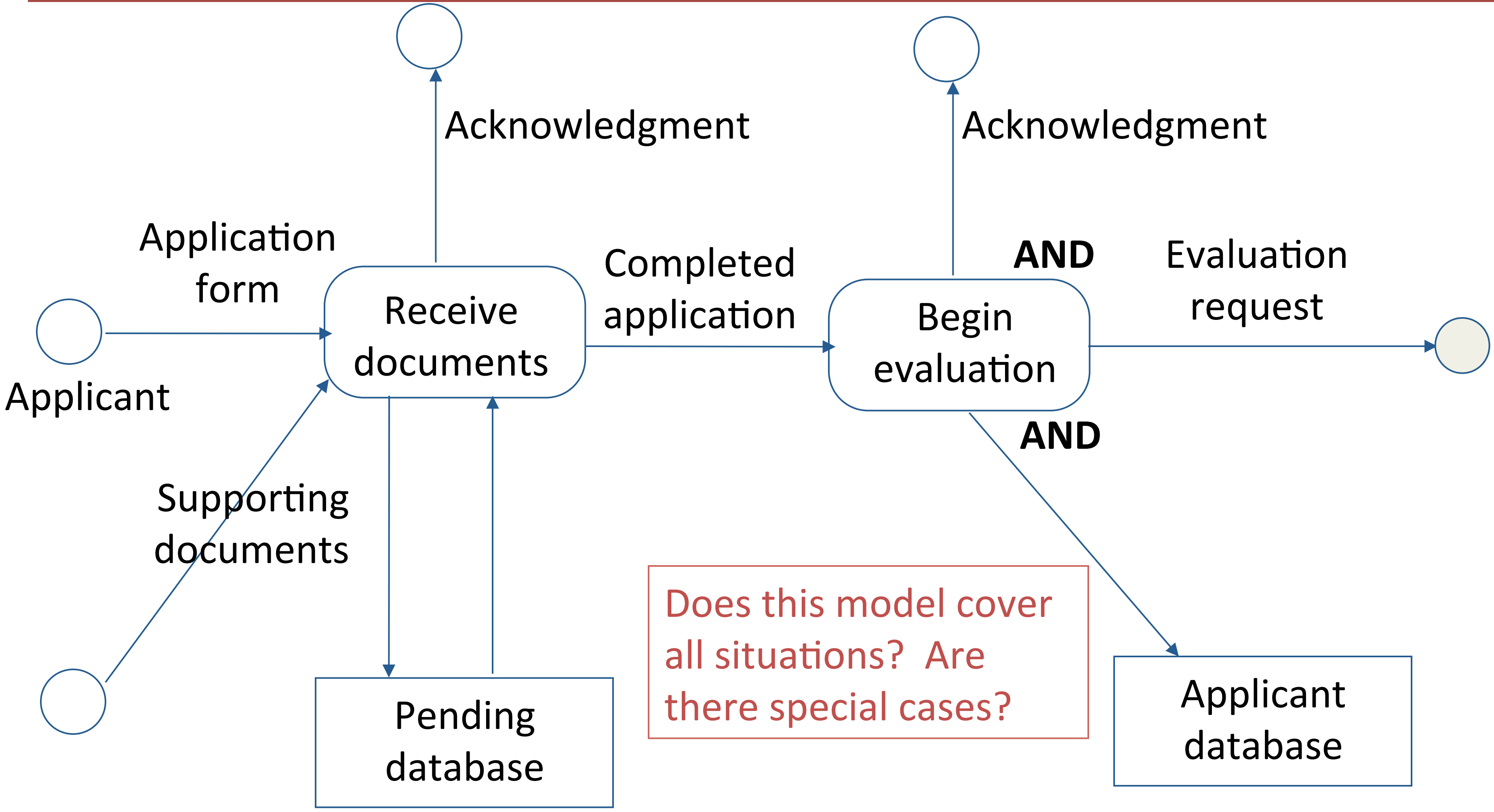
○	External entities
▢	Processing steps
□	Data stores or sources
→	Data flows

Data-Flow Model
Example: University Admissions (first attempt)



Shows the flow, but where is the data stored? Is there supporting information?

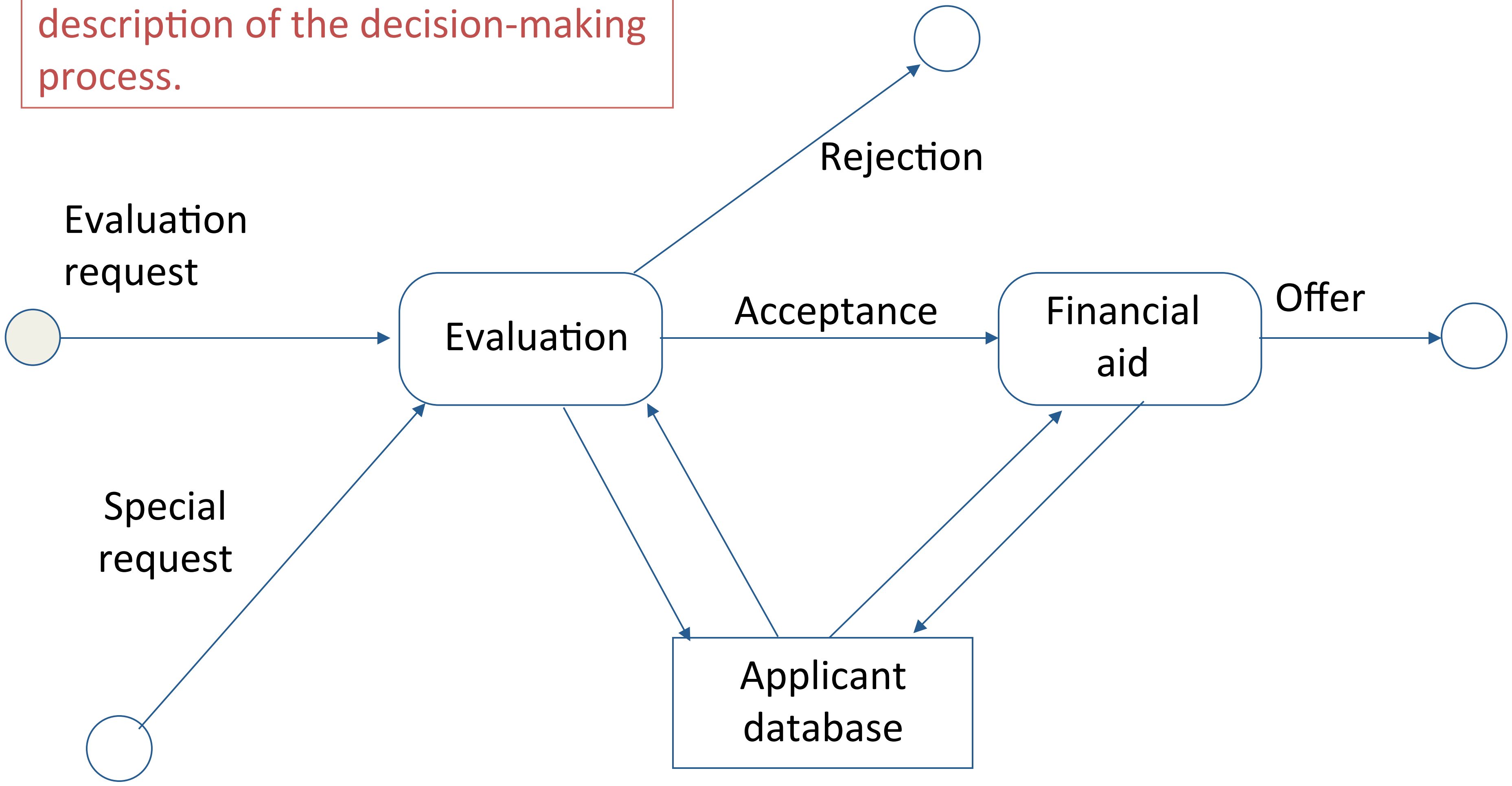
Data-Flow Model
Example: Assemble Application



Does this model cover all situations? Are there special cases?

Data-Flow Model
Example: Process Completed Application

The requirements will need a description of the decision-making process.



Decision Table Model

University Admission Decision

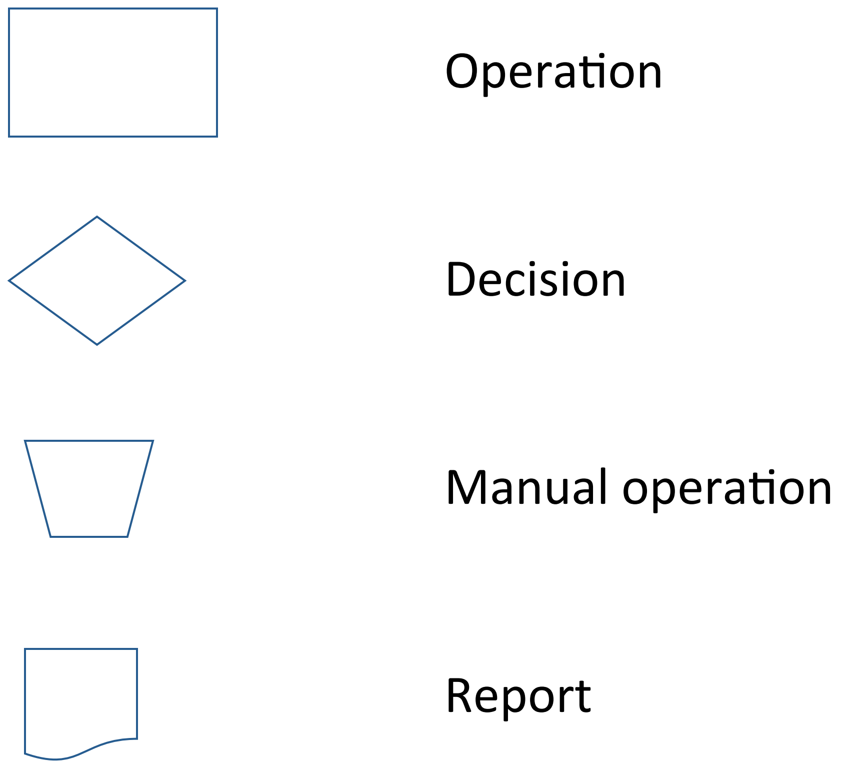
SAT > S1	T	F	F	F	F	F
GPA > G1	-	T	F	F	F	F
SAT between S1 and S2	-	-	T	T	F	F
GPA between G1 and G2	-	-	T	F	T	F
Accept	*X*	*X*	*X*			
Reject				*X*	*X*	*X*

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

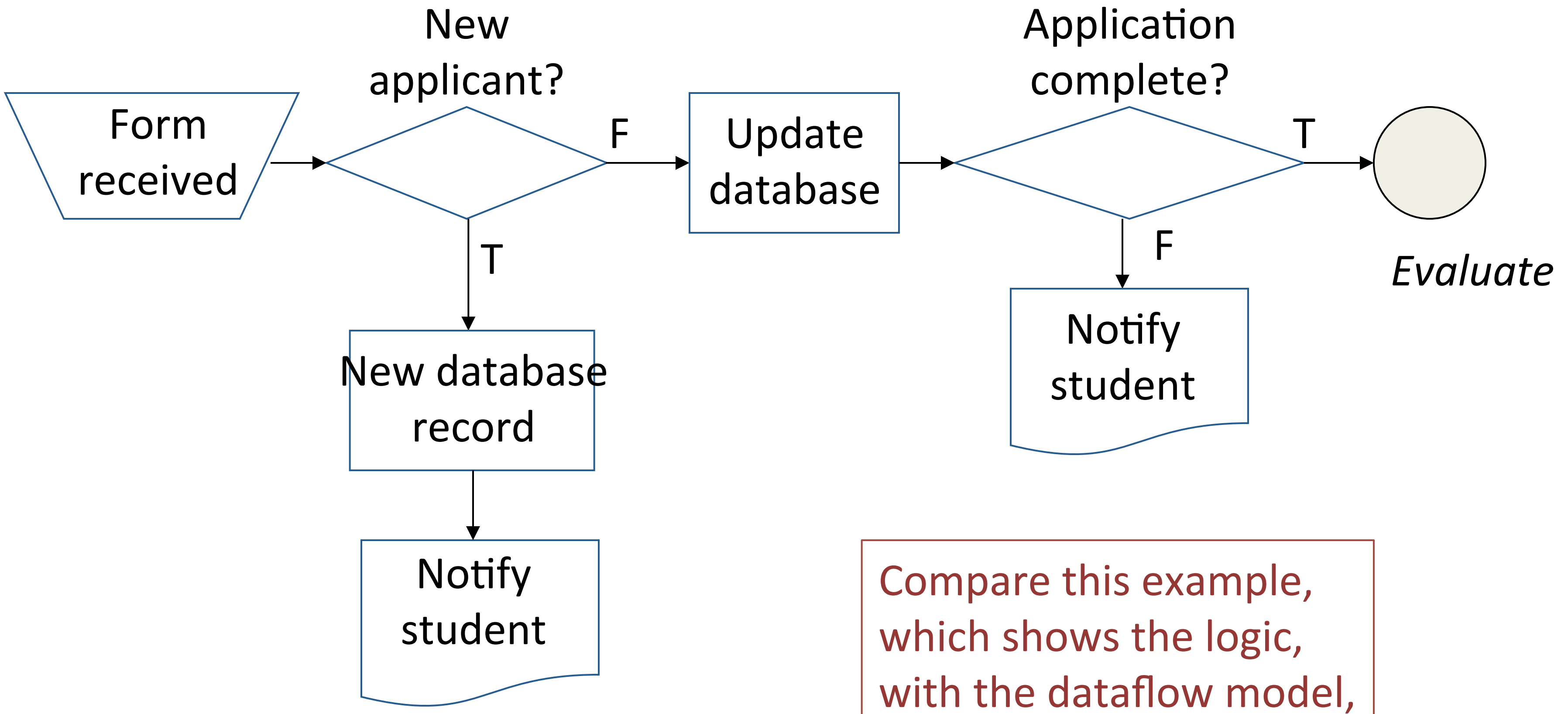
Note that the rules are specific and testable.

Flowchart Models

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



Flowchart Model
Example: University Admissions Assemble Application



Compare this example, which shows the logic, with the dataflow model, which shows the flow of data.

Modeling Tools: Pseudo-code

An informal modeling technique to show the logic behind part of a system.

Example: University Admission Decision

```
admin_decision (application)
    if application.SAT == null then error (incomplete)
    if application.SAT > S1 then accept(application)
    else if application.GPA > G1 then accept(application)
    else if application.SAT > S2 and application.GPA > G2
        then accept(application)
    else reject(application)
```

The notation used for pseudo-code can be informal, or a standard used by a software development organization, or based on a regular programming language. What matters is that its interpretation is understood by everybody involved.

Modeling Tools: Transition Diagrams

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

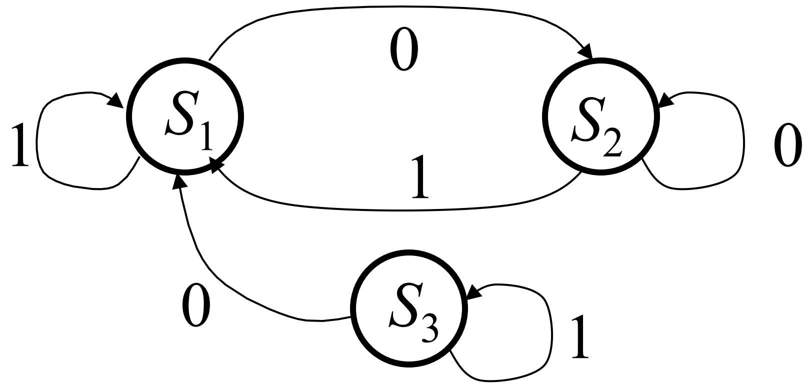
A **transition** is a change from one state to another.

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



Finite State Machine Model Example: Therapy Control Console

Example: Radiation Therapy Control Console

You are developing requirements for the operator's control console. In an interview, the client describes the procedures that the operator must follow when operating the machine.

You use a finite state machine model to specify the procedures.

This shows the client that you understand the requirements and specifies the procedures for the developers.

This scenario and state diagram are based on a published example. Unfortunately I have no record of the source. If you know it, please contact me so that I can acknowledge the author.

Finite State Machine Model
Therapy Control Console: Scenario

Scenario

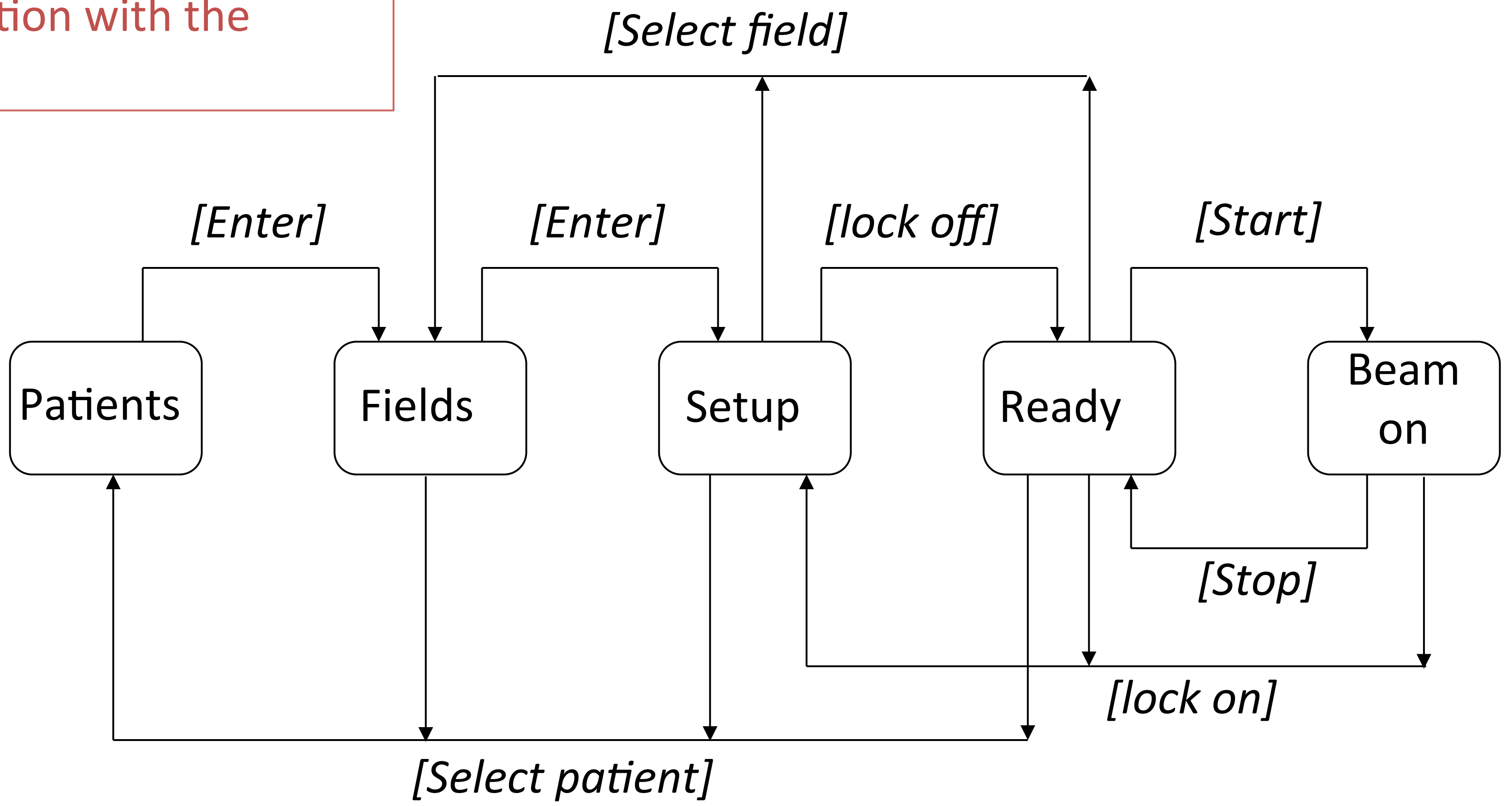
The client provides the following rough scenario.

"The set up is carried out before the patient is made ready. The operator selects the patient information from a database. This provides a list of radiation fields that are approved for this patient. The operator selects the first field. This completes the set up.

"The patient is now made ready. The lock is taken off the machine and the doses with this field are applied. The operator then returns to the field selection and chooses another field."

Finite State Machine Model
State Transition Diagram

Discuss each state and transition with the client.

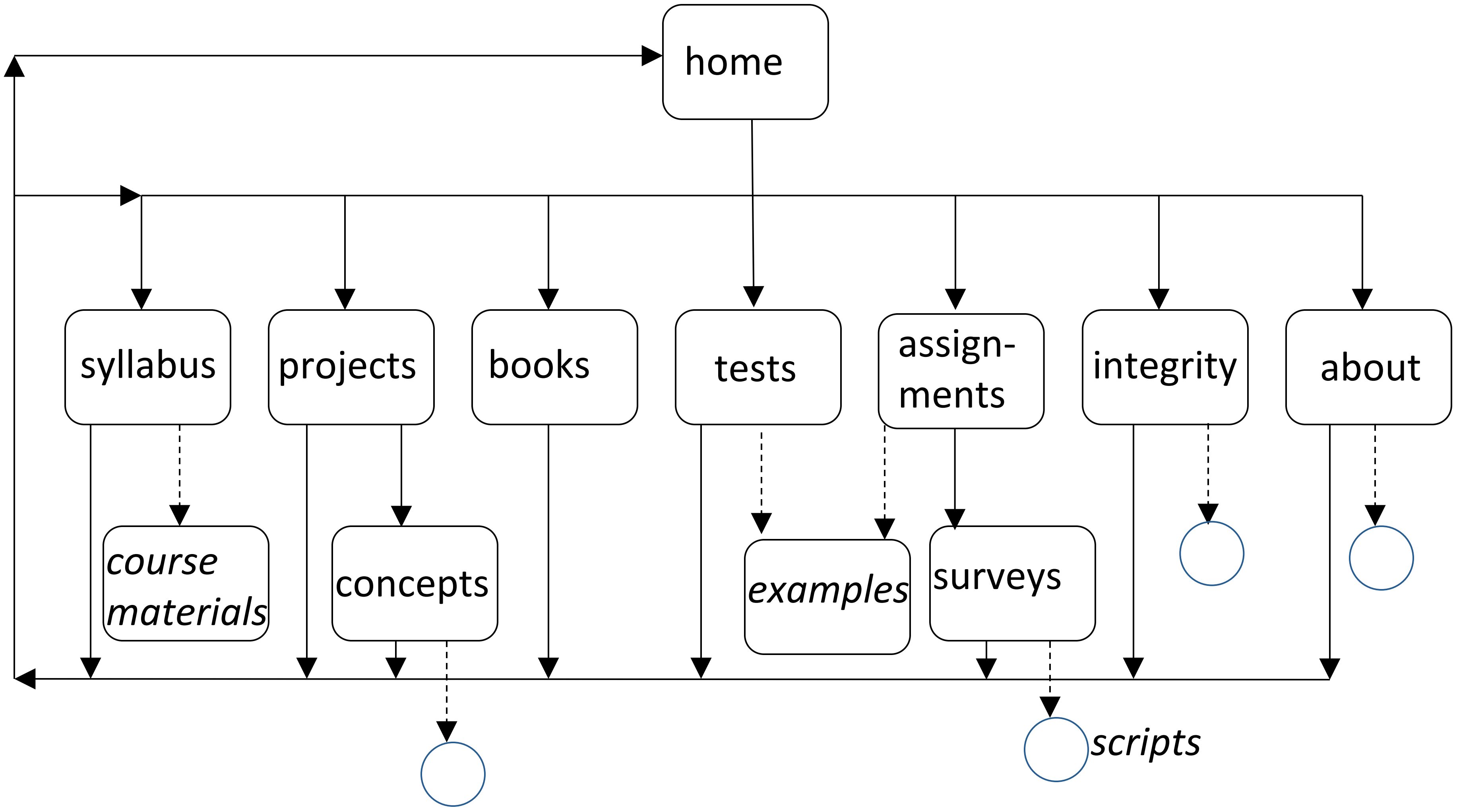


Finite State Machine Model
State Transition Table

	Select Patient	*Select Field*	*Enter*	*lock off*	*Start*	*Stop*	*lock on*
Patients	——	——	Fields	——	——	——	——
Fields	Patients	——	Setup	——	——	——	——
Setup	Patients	Fields	——	Ready	——	——	——
Ready	Patients	Fields	——	——	Beam on	——	Setup
Beam on	——	——	——	——	——	Ready	Setup

This table can be used for requirements definition, program design, and acceptance testing.

Transition Diagram for User Interfaces
Example: CS 5150 Web Site (part)



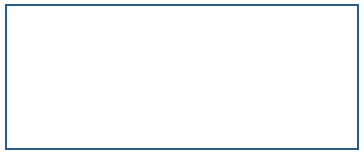
Entity-Relation Model

A requirements and design methodology for relational databases

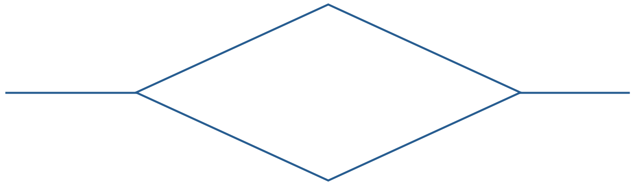
- A database of entities and relations
- Tools for displaying and manipulating entity-relation diagrams
- Tools for manipulating the database (e.g., as input to database design)

Entity-relationship models can be used both for requirements specification and for the design specification.

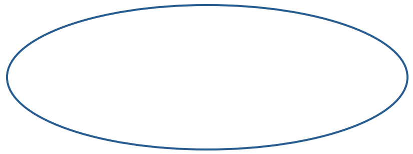
Modeling Tools: Entity-Relation Diagram



An entity (noun)



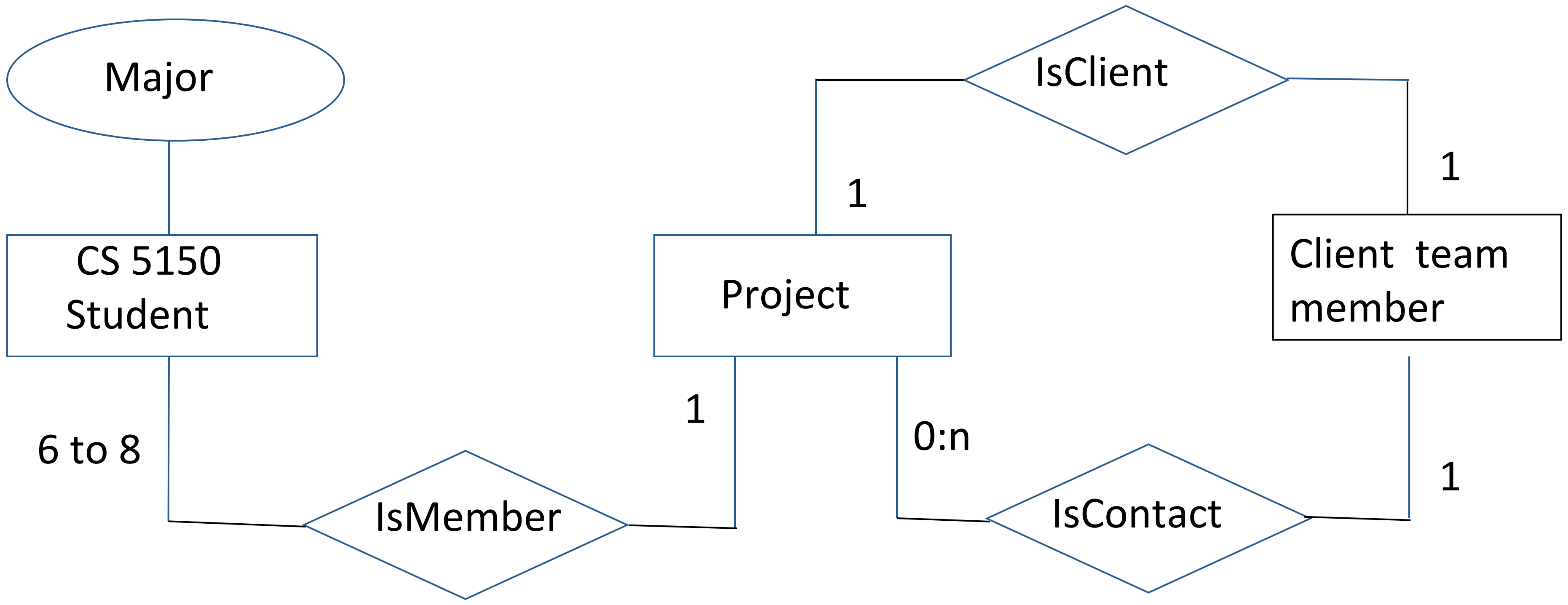
A relation between entities (verb)



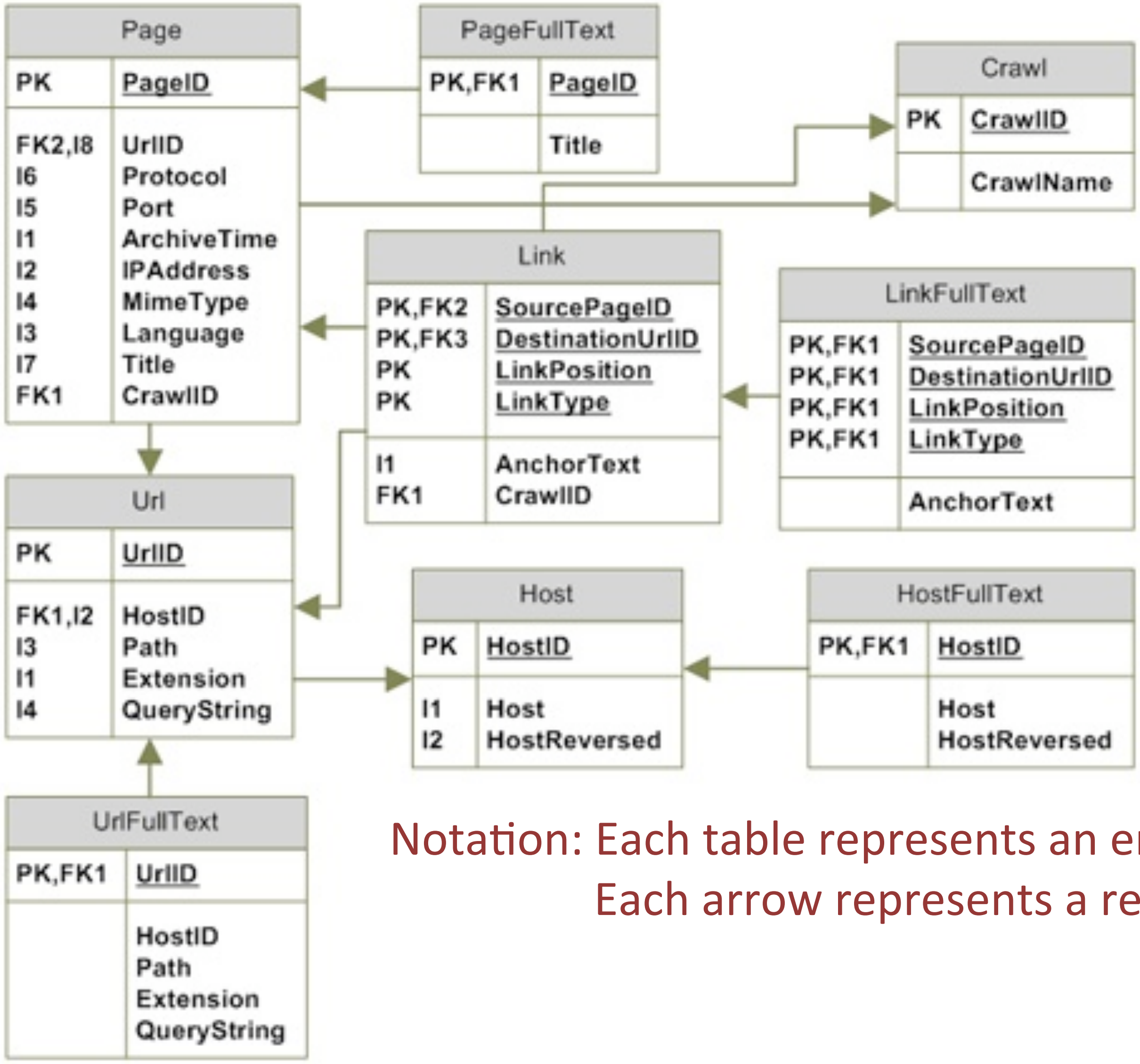
An entity or relation attribute

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

Modeling Tools: Entity Relationship Diagram
Example: CS 5150 Project



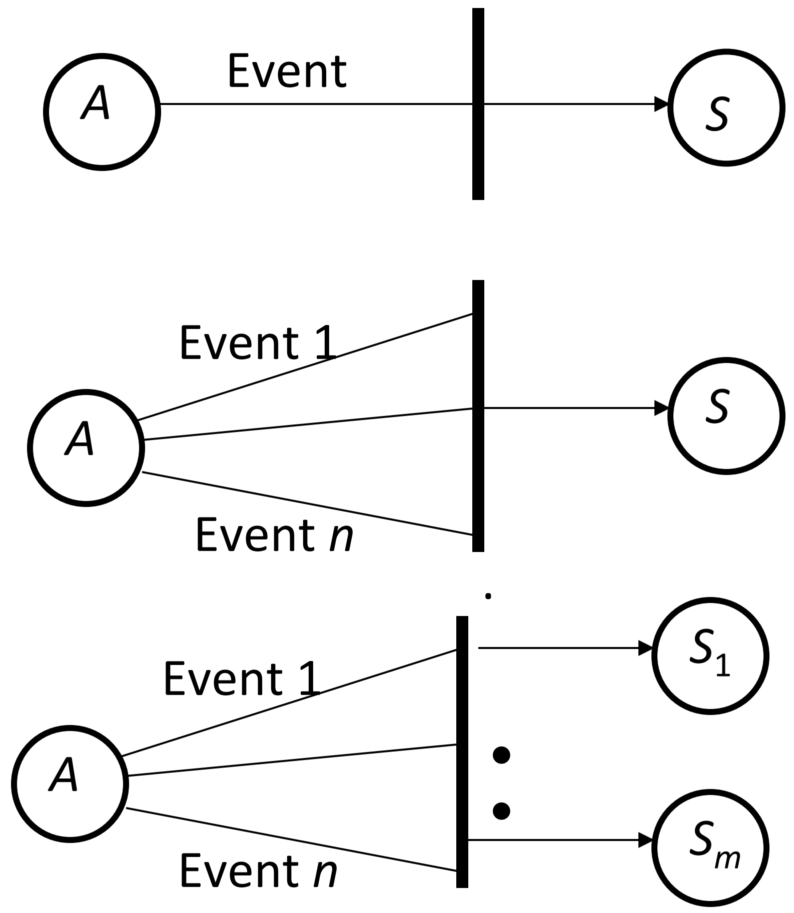
Entity Relationship Diagram as a Design Tool Example: Database Schema for Web Data



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

Petri Net Models

A Petri Net models parallelism



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

Requirements Definition: Data Dictionaries

A data dictionary is a list of names used by the system

- Name (e.g., "start_date")
- Brief definition (e.g., what is "date")
- What is it? (e.g., integer, relation)
- Where is it used (e.g., source, used by, etc.)
- May be combined with a glossary

As the system is developed, the data dictionary in the requirements is the basis of the system data dictionary, which is part of the final specification.

A Note on Class and Object Models

This course teaches class and object models as a tool for **design**, not for modeling requirements.

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

Flow charts and finite state machines are supported by UML as design models, but are equally useful for requirements.