

SENG321: Requirements Engineering

Domain and Process Modeling (Structured Analysis Techniques)

also known as Traditional 'Structured Analysis' Modelling

Review:

To complement the **Textual Requirements** it also includes:

Data modeling artifacts:

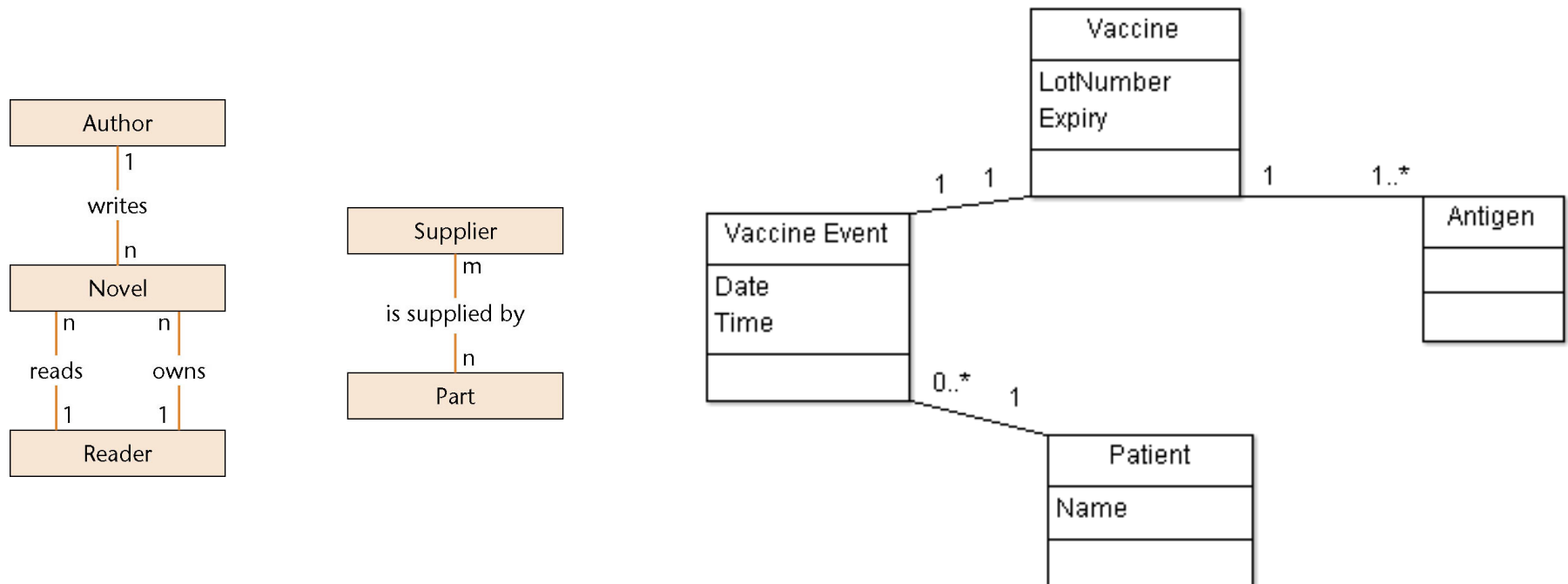
Entity Relationship Diagrams (ERDs)

Data dictionary (DD)

Process modelling artifacts:

Data flow diagrams (DFDs)

Entity-relationship Diagrams (ERDs)



Vaccine: A specific instance of vaccine, containing 1 or more antigens and having an assigned lot number and expiry date.

Data Modeling (Review)

Identify possible domain entities as **nouns** in requirements notes

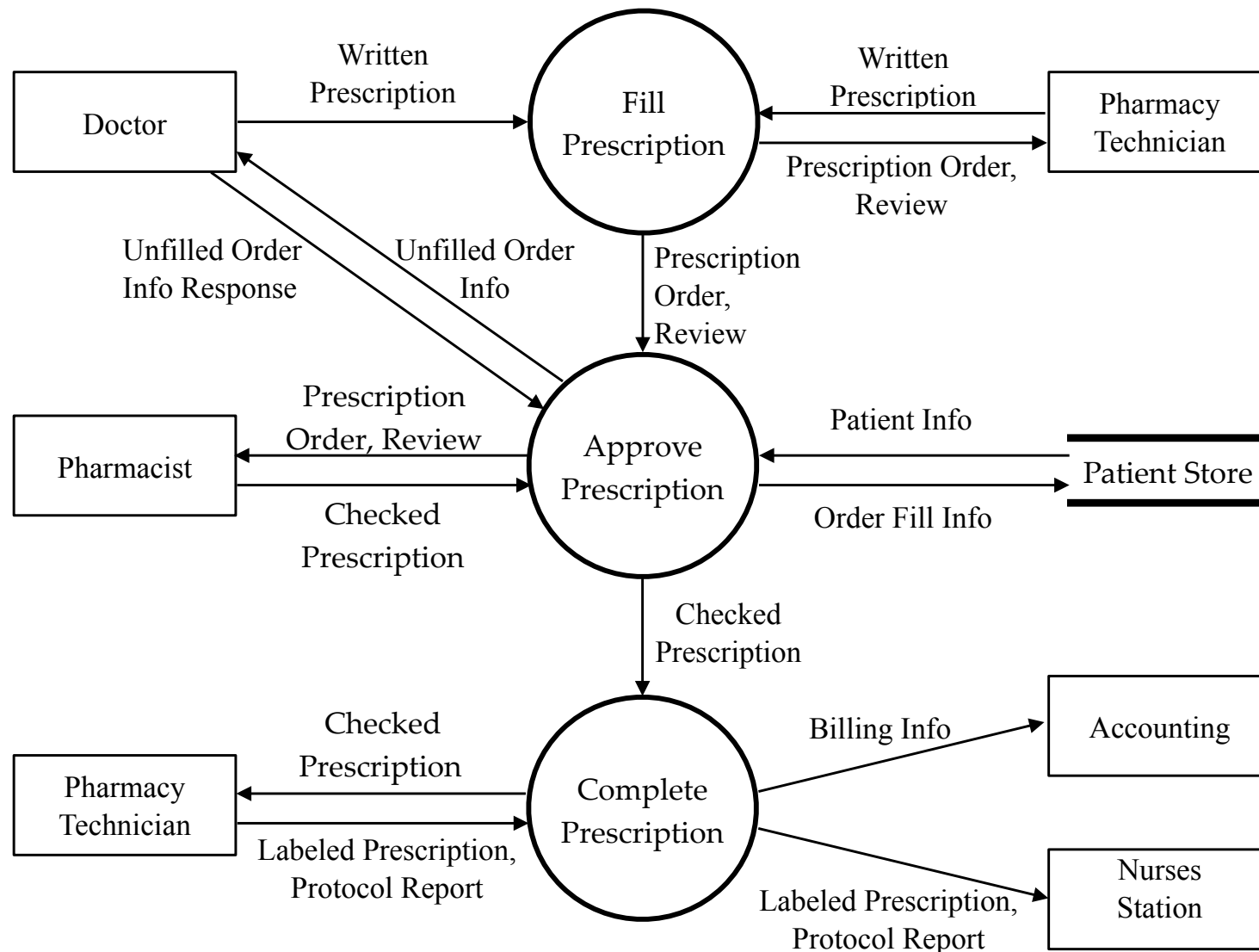
(Narrow this list to just those mentioned in identified use cases)

Draw **data entities** with relationships → **Data Entity Relationship** diagrams

Refine **relationships** with cardinality (numbers of each class compared to their related classes)

Describe each domain entity in the **Data Dictionary**

Hospital Pharmacy System: Data Flow Diagram (DFD)



Process-oriented modelling (data flow modelling)


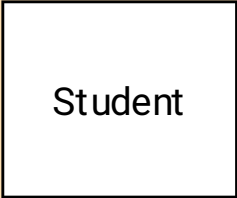

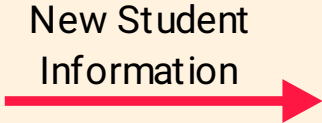
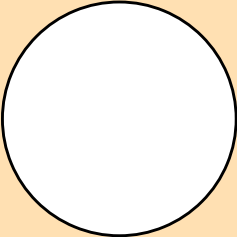
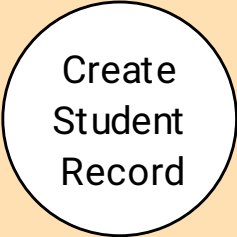
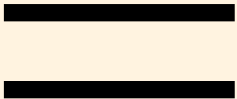

Identify possible processes/functions as **verbs (actions on the data you defined)** in requirements notes

(Narrow this list to just those mentioned in identified use cases)

Draw **process bubbles** with relationships (data flow)→ **Data Flow Diagrams**

Refine **diagrams at different levels of detail**

Data Flow Diagram (DFD) components

Symbol	Meaning	Example
	Entity	
	Data Flow	
	Process	
	Data Store	

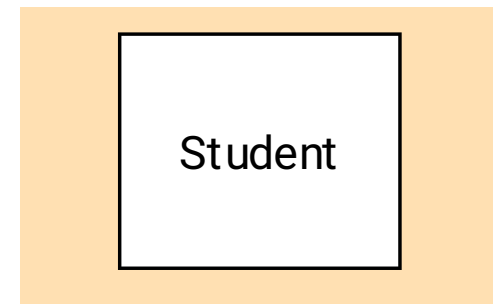
External Entities

Similar to actors in Use Cases

Represent another department, a business, a person, or a machine

A source or destination of data, outside the boundaries of the system

Should be named with a noun



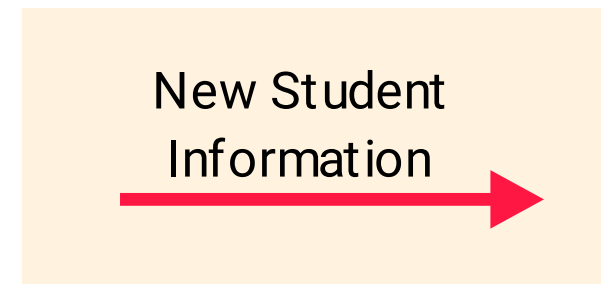
Data Flow

Shows movement of information from one point to another

Described with a **noun**

Arrowhead indicates the flow direction

Represents info about a person, place, or thing



Process

Similar to **use cases** or **steps in a use case**

Each process transforms information

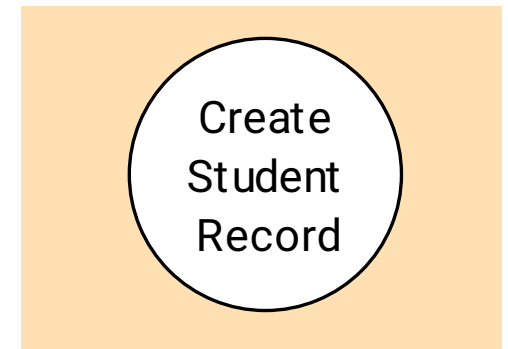
Describes work performed in a system (note: system is not necessarily computer-based)

Naming conventions:

At **context** level – use **system** name

At **subsystem** level, name with subsystem name + the word "subsystem"

At more detailed levels use **verb-adjective-noun** for detailed processes



Data Store

A place where information is stored

Allows examination, addition, and retrieval information

Named with a **noun**, describing the information

Can be given a unique reference number, such as D1, D2, D3

May represent a:

- Database

- File

- Third party storage (e.g. cloud)



A diagram of a data store. It consists of a light orange rectangular box. Inside the box, the text "Student Master" is centered. Above and below this text are two thick, solid black horizontal bars, one above and one below, acting as visual dividers.

Steps in developing DFDs (basics)

1. Make list of business activities and things (potential **entities**)
2. Create **context level diagram** ("DFD 0") including external entities and summary data flows
3. Create a **child diagram** for each complex process on diagram 0. Include local data stores and detailed processes.
4. Check for errors and make sure names of processes and data stores are meaningful..
5. Repeat steps 3 and 4 until no more decomposition is meaningful

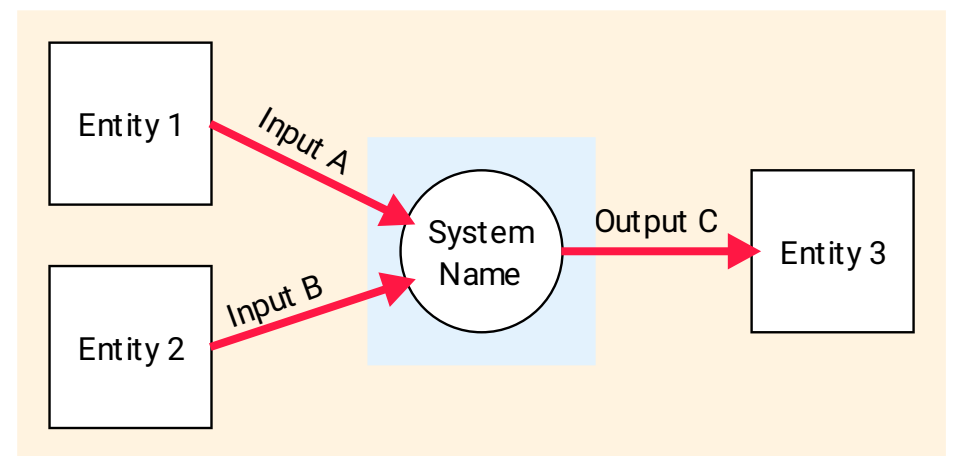
Creating the Context Diagram (DFD 0)

The highest level in a data flow diagram

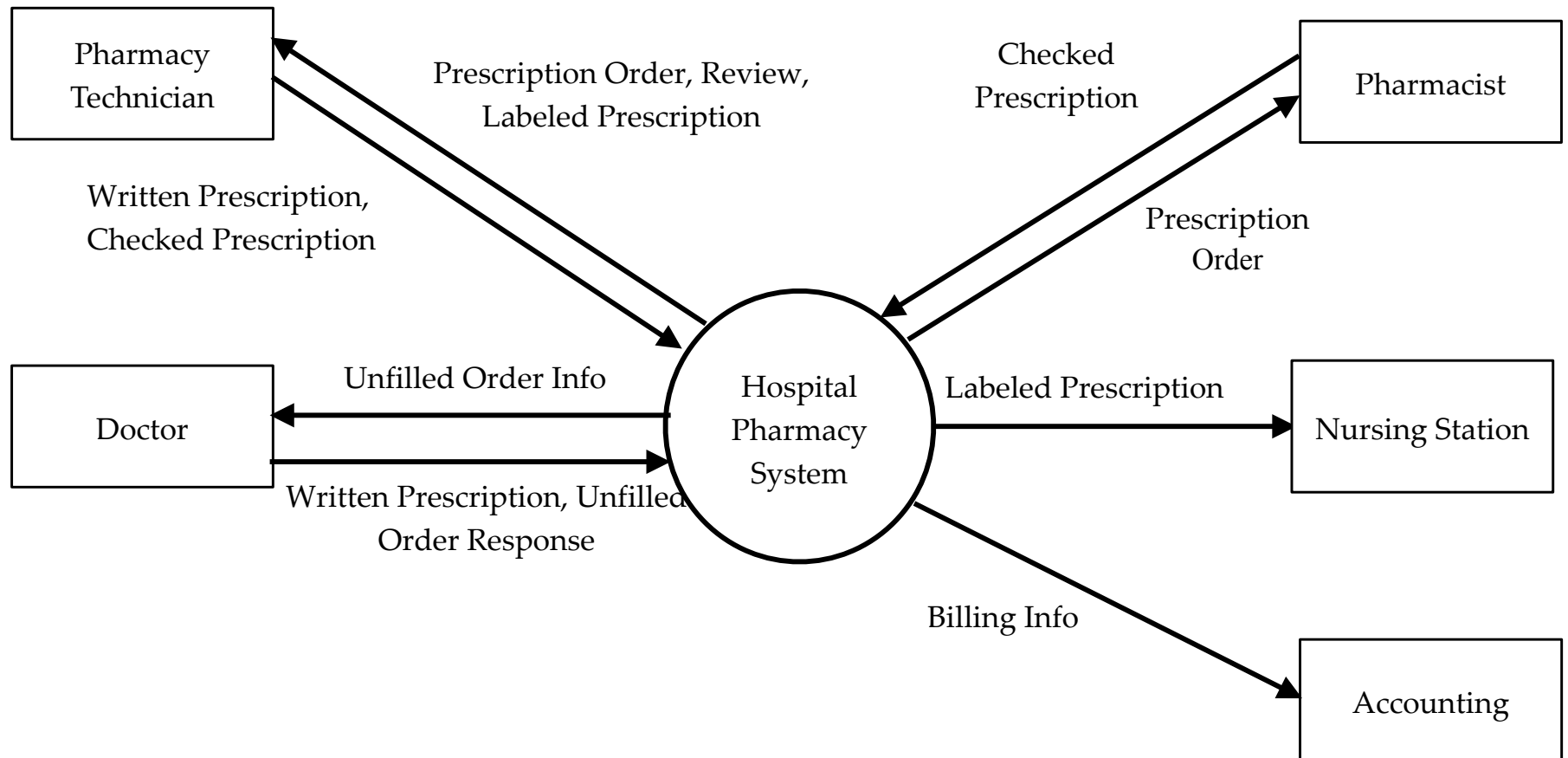
Contains only one process, representing the entire system

The process is given the number 0

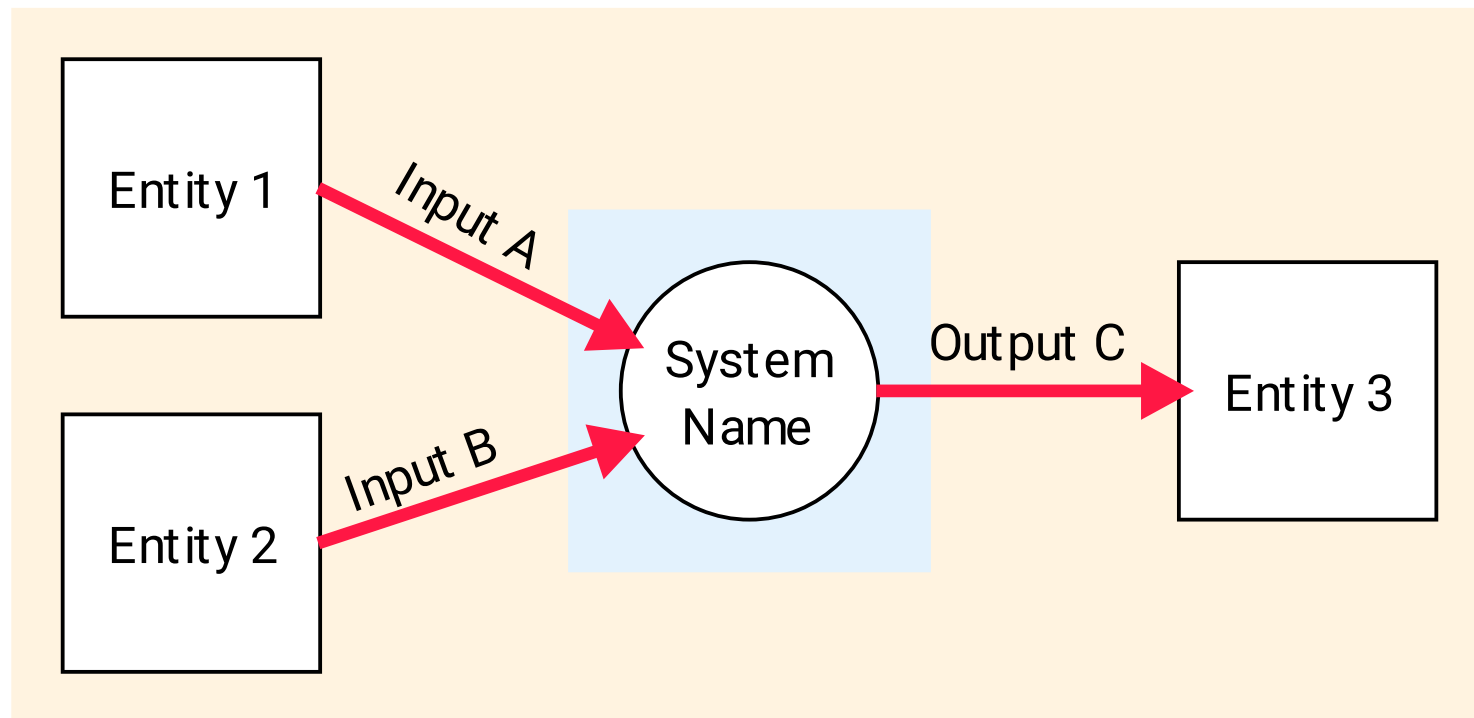
All external entities, as well as major data flows are shown



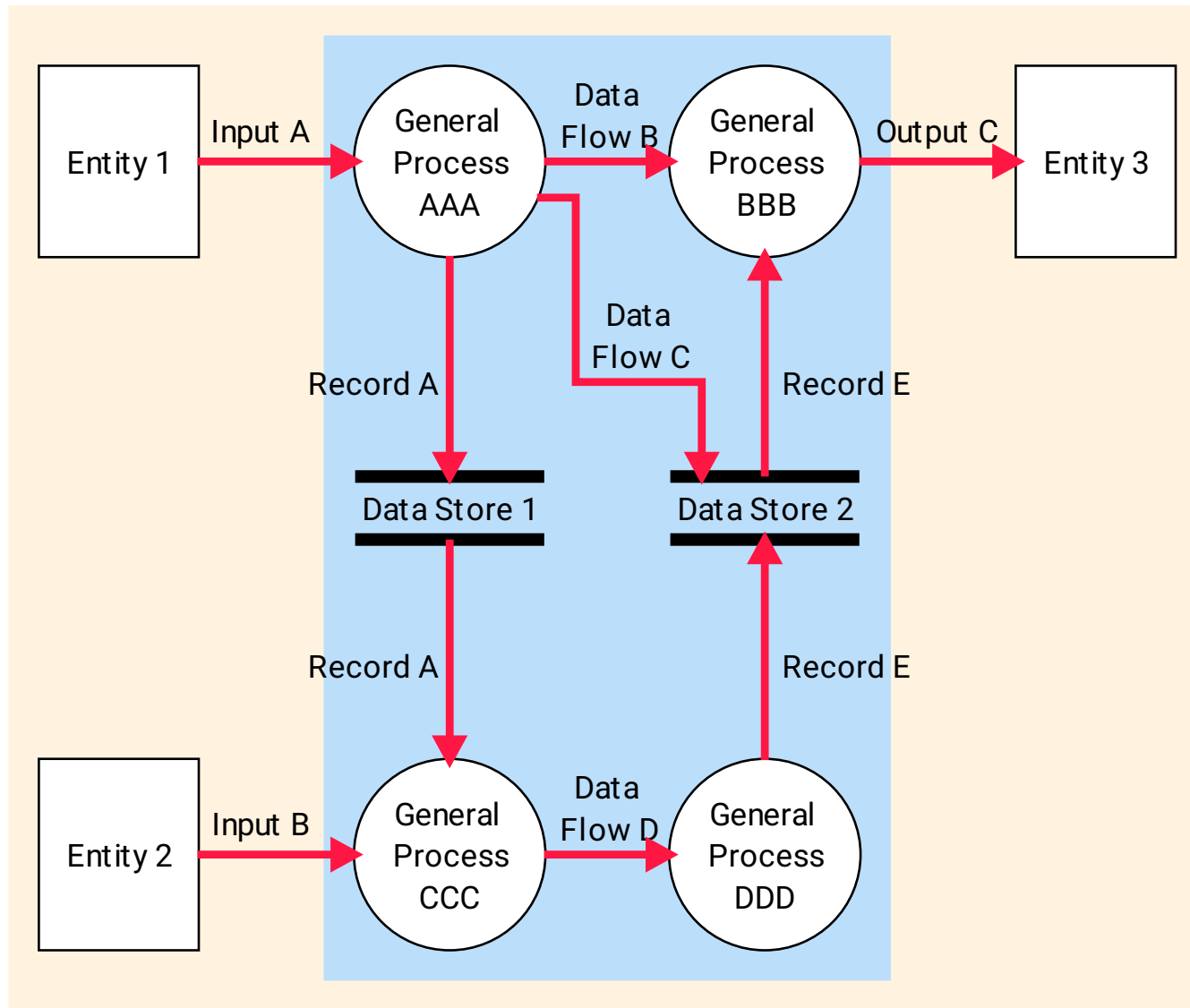
Context DFD example: Hospital Pharmacy



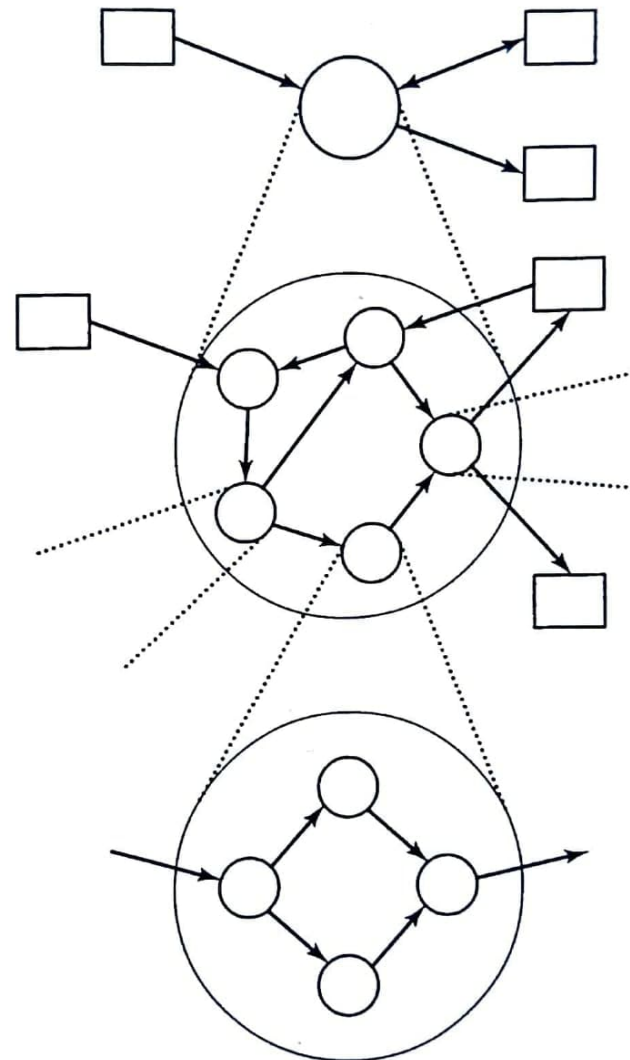
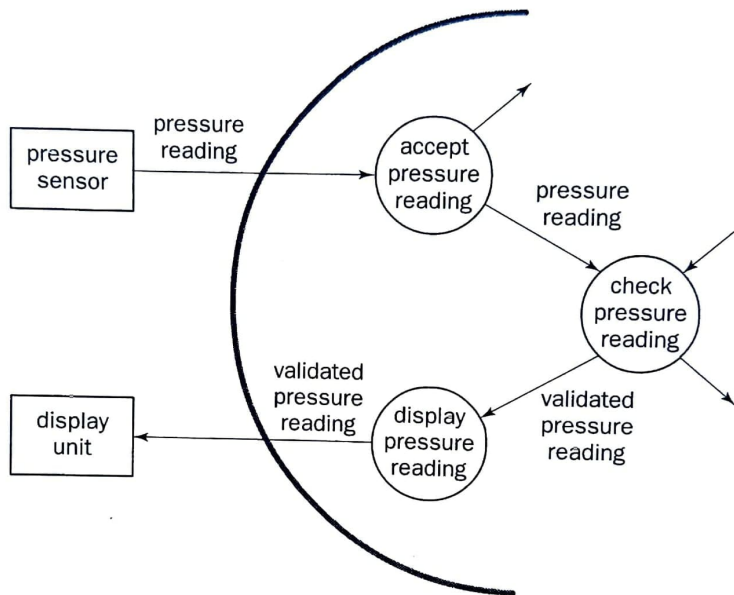
From the Context diagram (DFD 0)...



to DFD Level 1 (example)



Nesting of the DFD diagrams

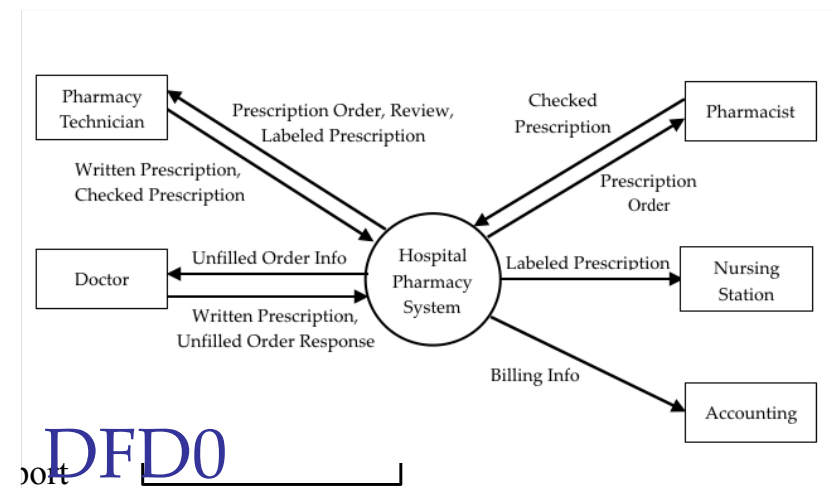
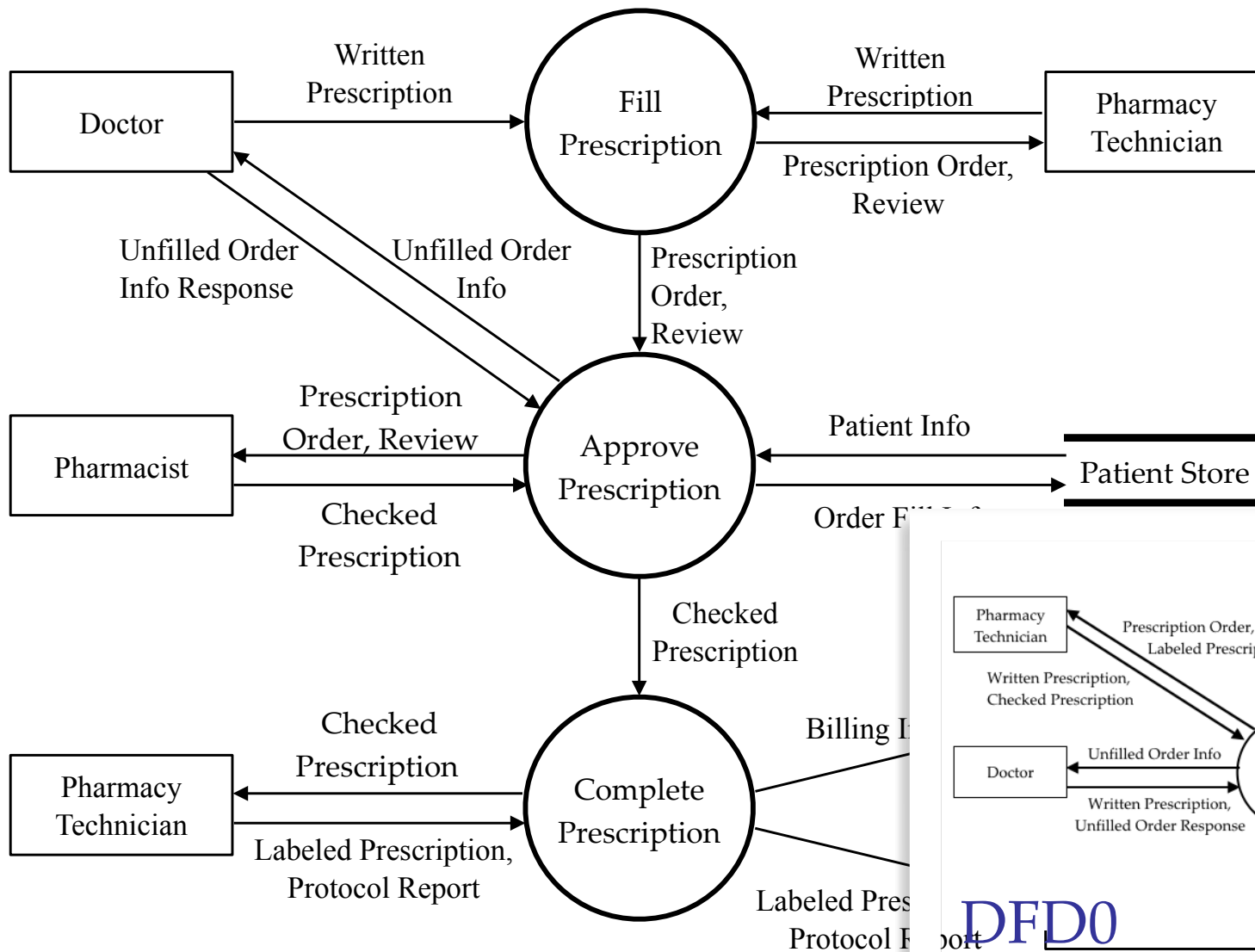


DFD0

DFD1

(part of) DFD2

Example of two levels of DFDs in the Hospital Pharmacy System



DFD0

Data Flow Diagram Levels

Data flow diagrams are built in layers

The top level is the Context level

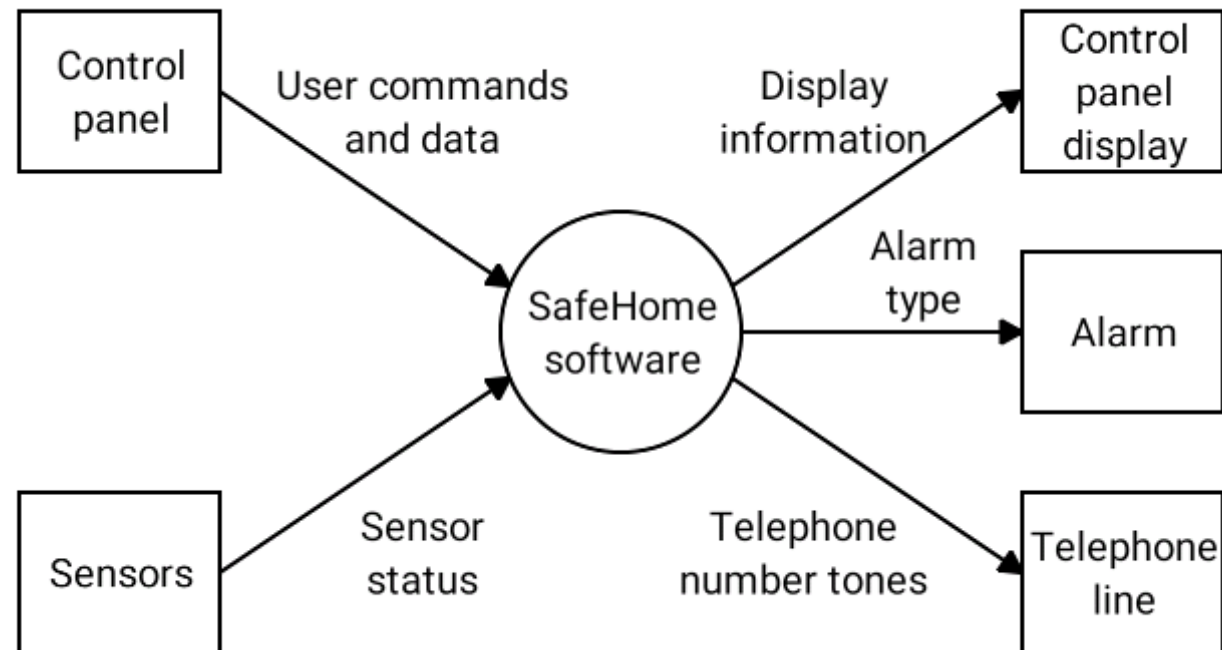
Each process 'blows up' to a lower, more detailed level

The lower level diagram number is the same as the parent process number

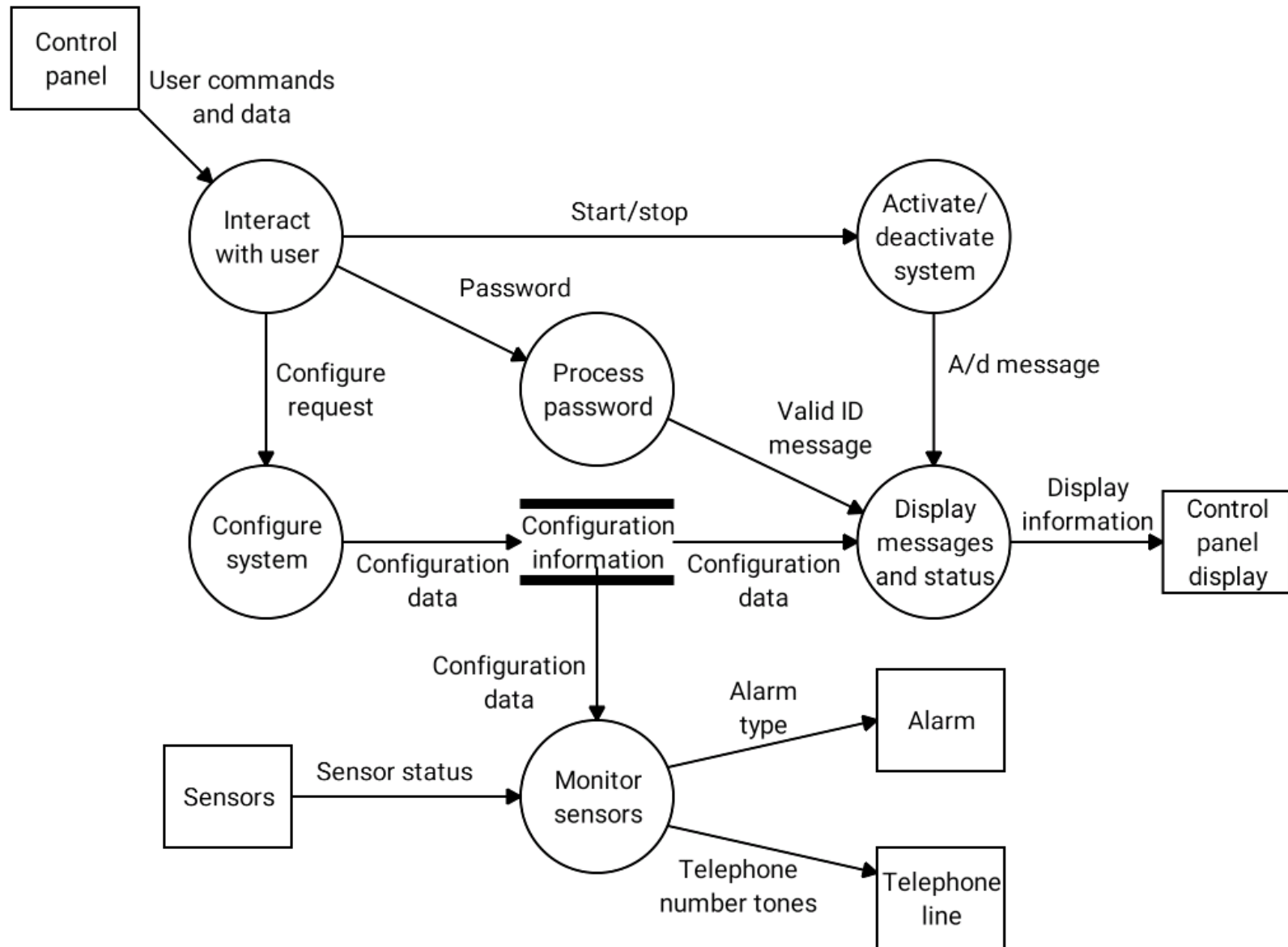
Processes that do not create a child diagram are called primitive

Another Example: SafeHome Software

DFD 0 (Context Diagram)



Associated DFD1 for SafeHome Software



Example from Textbook

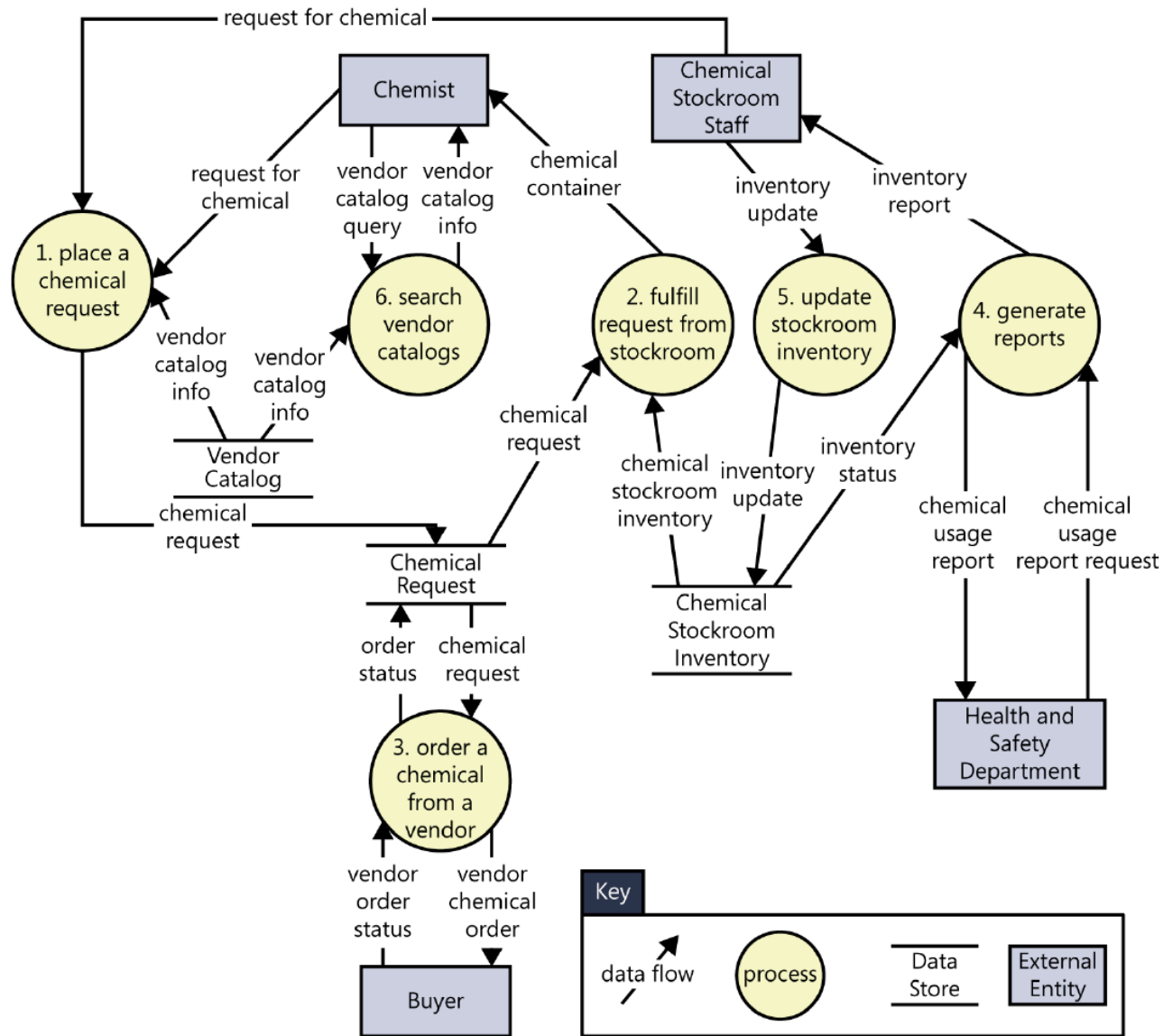
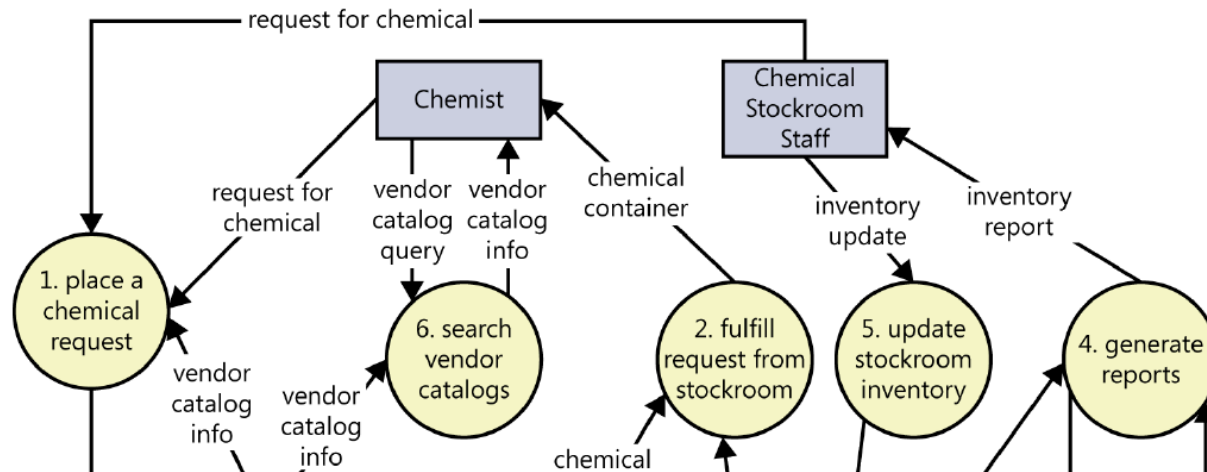


FIGURE 12-1 Partial level 0 data flow diagram for the Chemical Tracking System.

Example from Textbook



Note:
Textbook refers to this as “DFD 0”

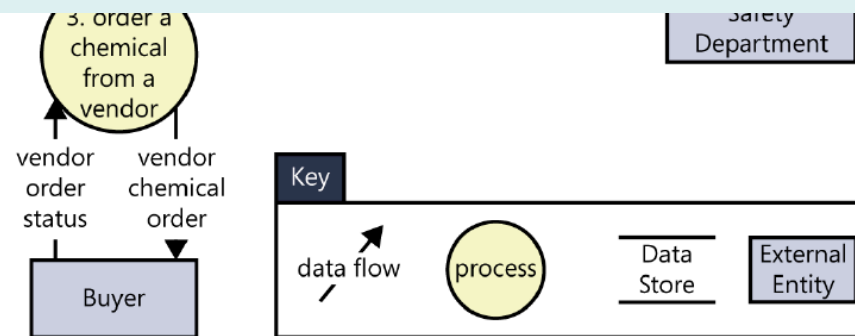


FIGURE 12-1 Partial level 0 data flow diagram for the Chemical Tracking System.

Checking the Diagrams for Errors (Continued)

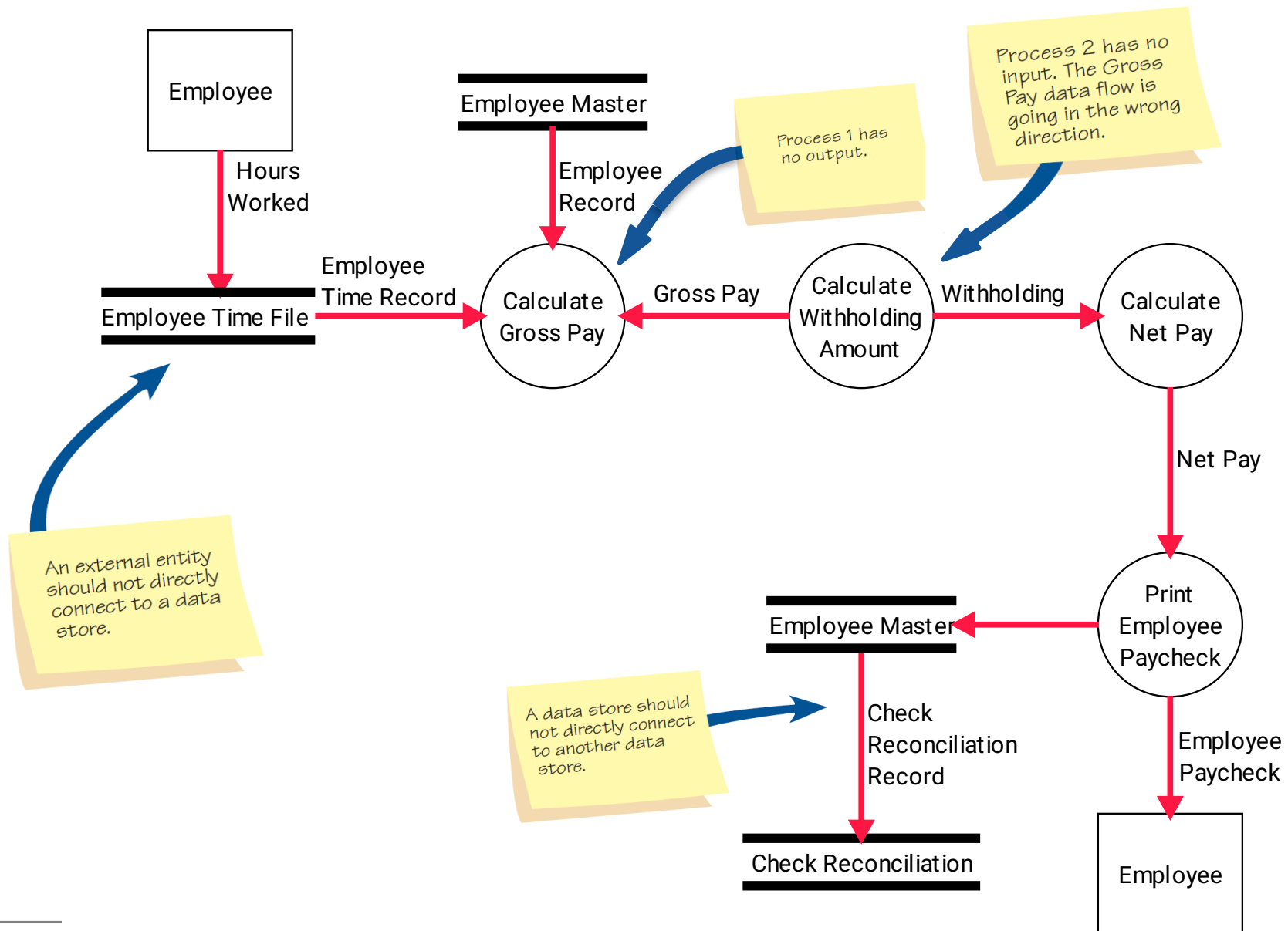
Incorrectly labeling processes or data flow

Including more than nine processes on a data flow diagram

Must ensure diagram is consistent with its parent diagram

Same external entities and input and output flows as parent

Typical errors in a data flow diagram



SWIMLANE diagram

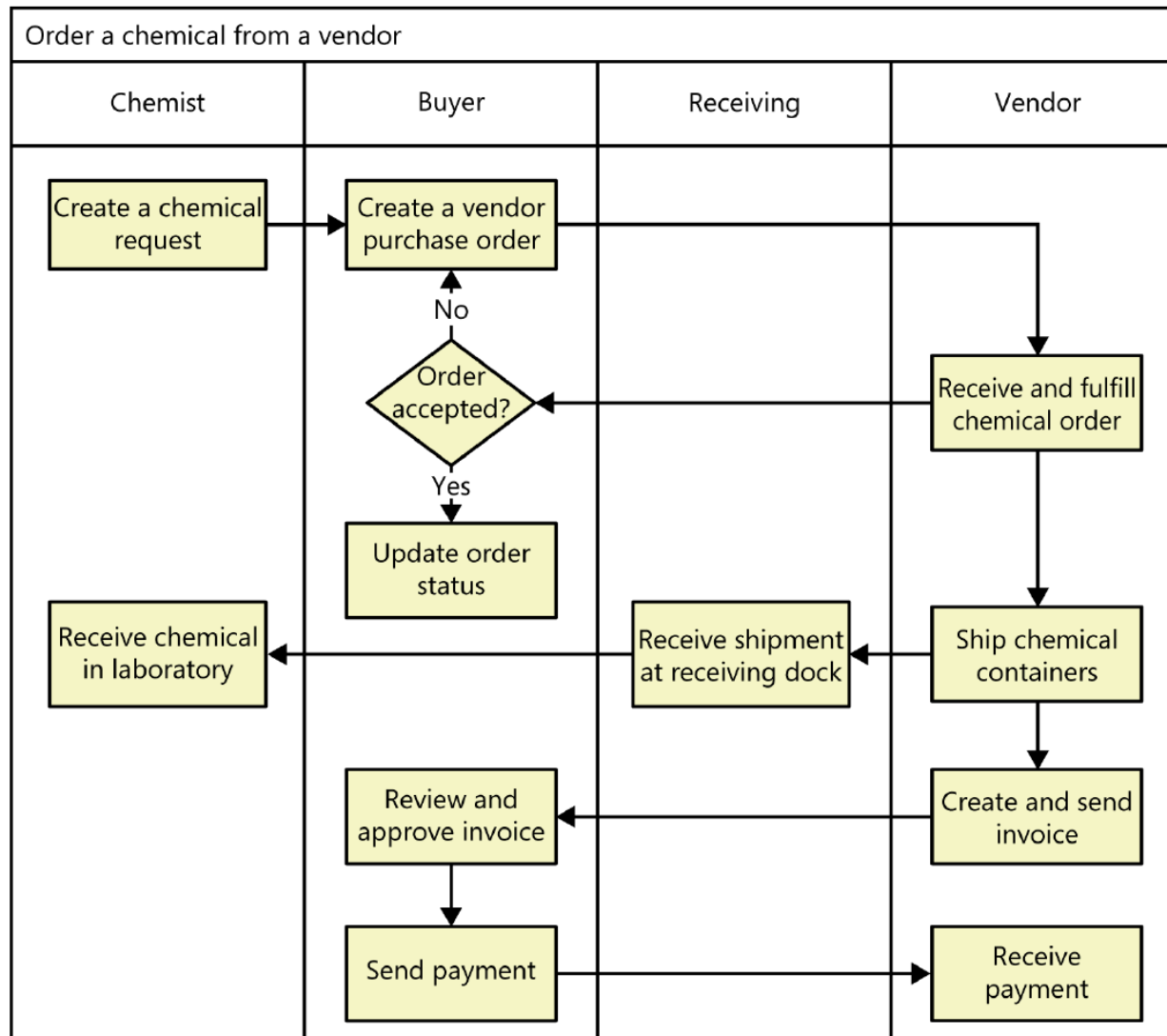
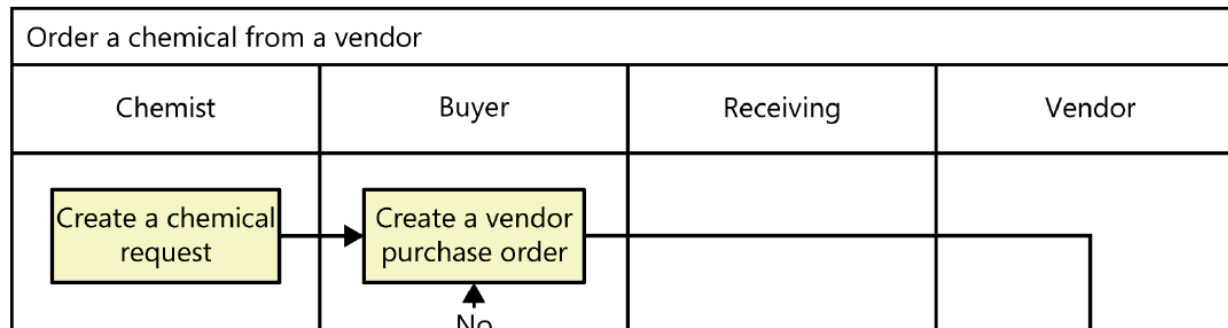


FIGURE 12-2 Partial swimlane diagram for a process in the Chemical Tracking System.

SWIMLANE diagram



In your Project:
Use in conjunction with Use Cases

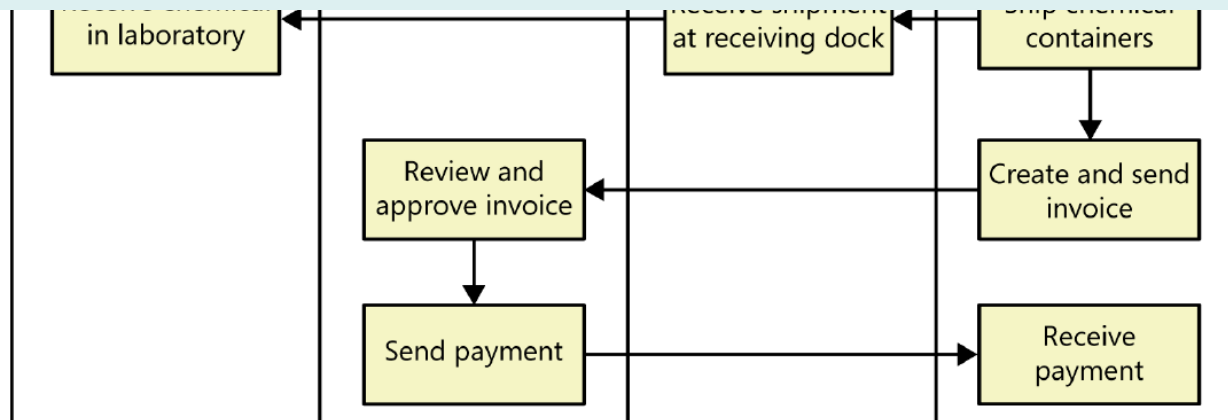


FIGURE 12-2 Partial swimlane diagram for a process in the Chemical Tracking System.

DIALOG MAPS

Useful for UX design

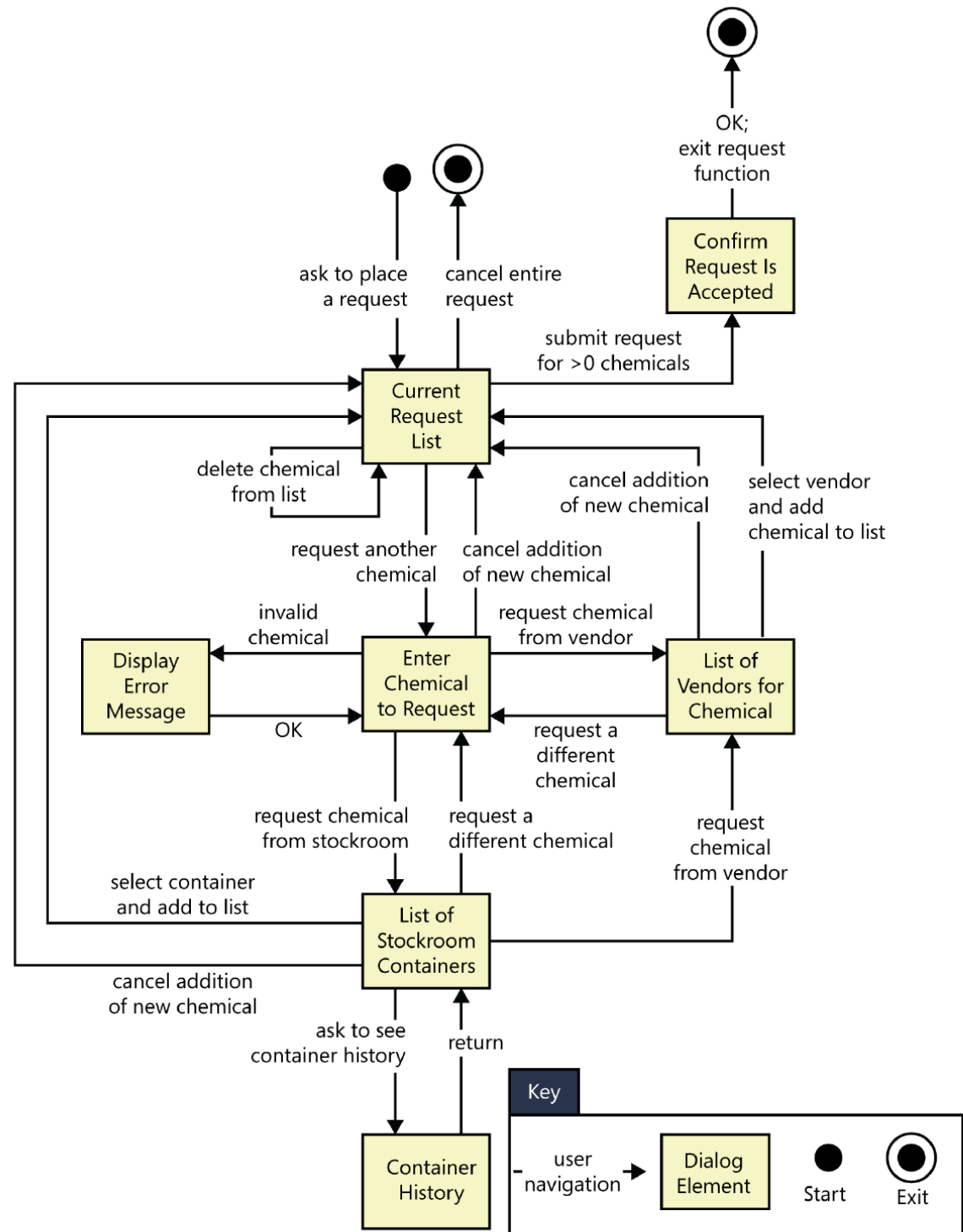
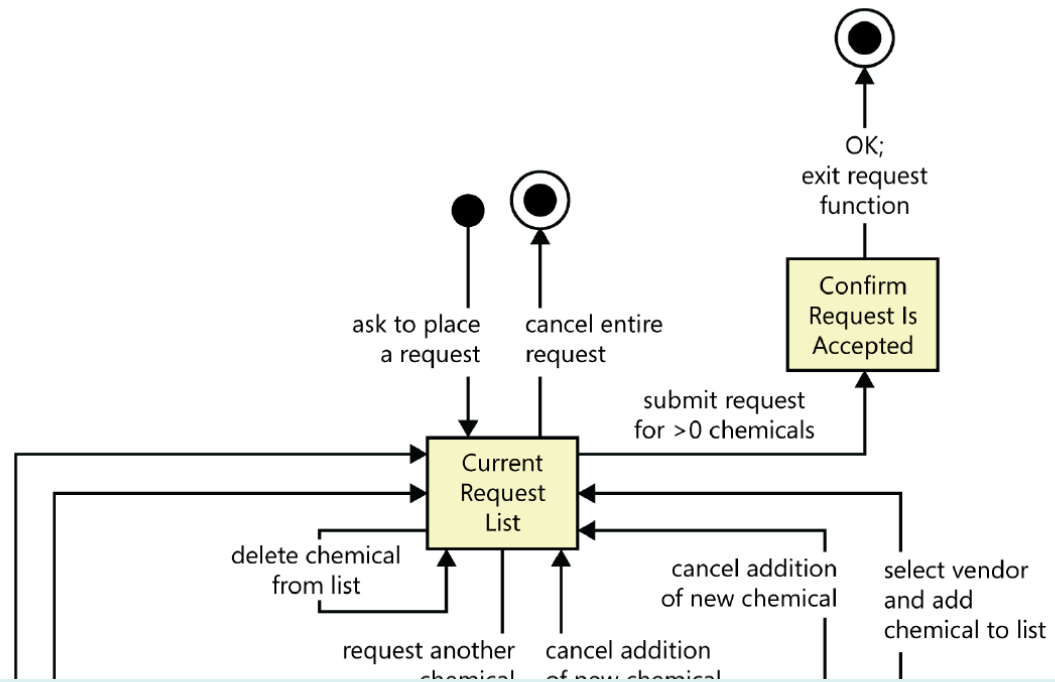


FIGURE 12-5 A partial dialog map for the "Request a Chemical" use case from the Chemical Tracking System.

DIALOG MAPS

Useful for UX design



In your Project:
Use in conjunction with at least 2 Storyboards (not use cases)

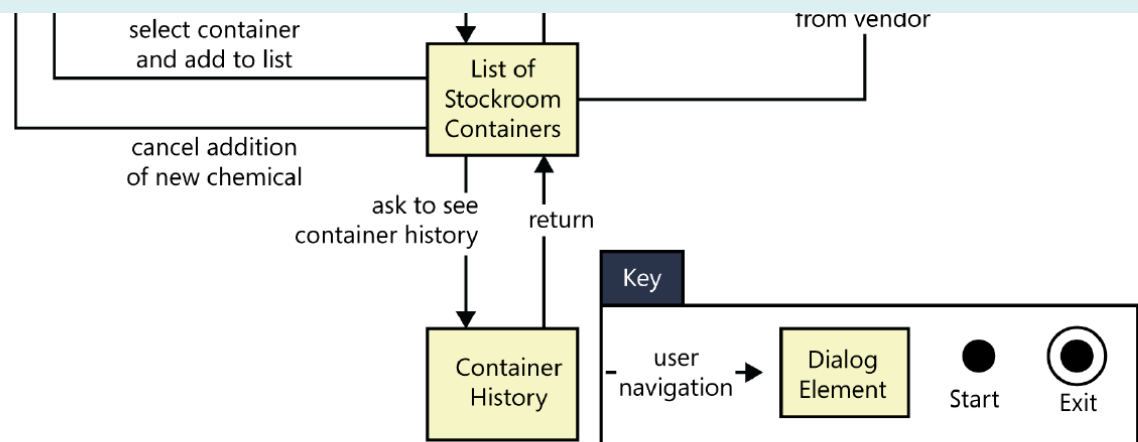


FIGURE 12-5 A partial dialog map for the "Request a Chemical" use case from the Chemical Tracking System.

Other useful diagrams: Decision Trees

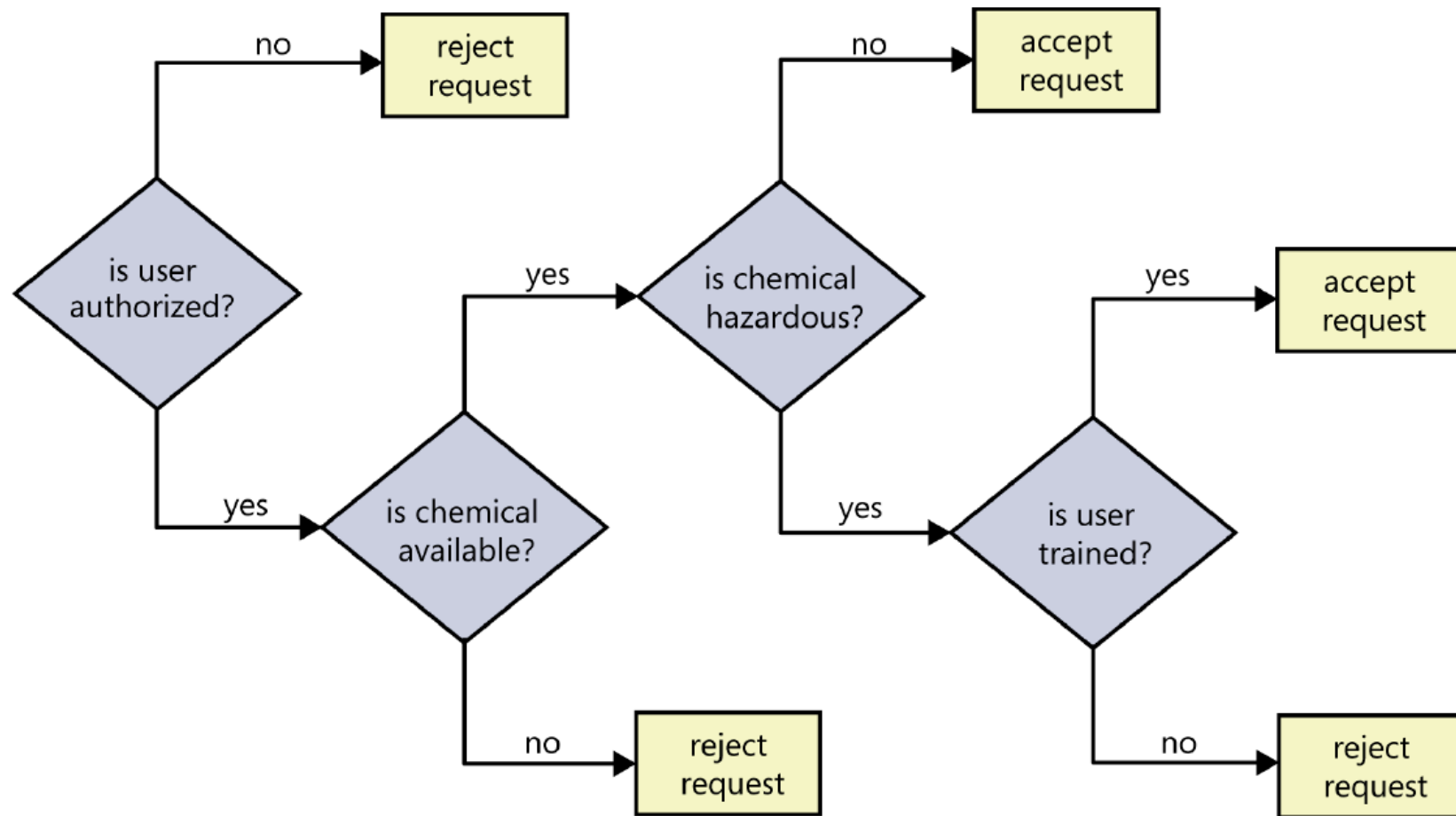


FIGURE 12-7 Sample decision tree for the Chemical Tracking System.

Object Oriented Modelling

Could model processes with DFD

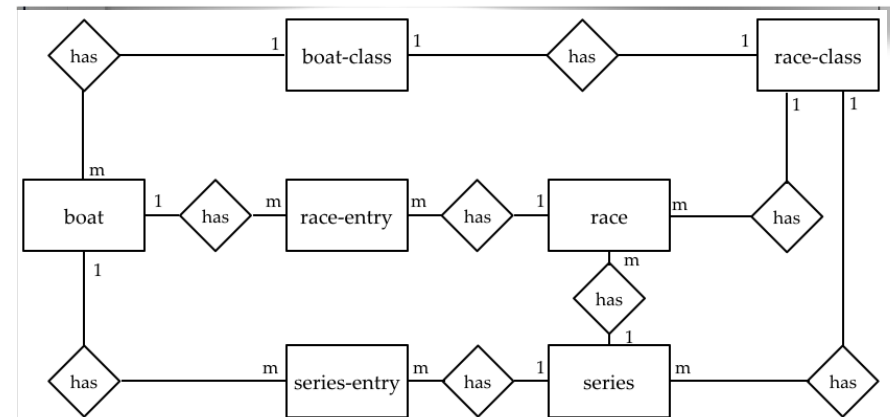
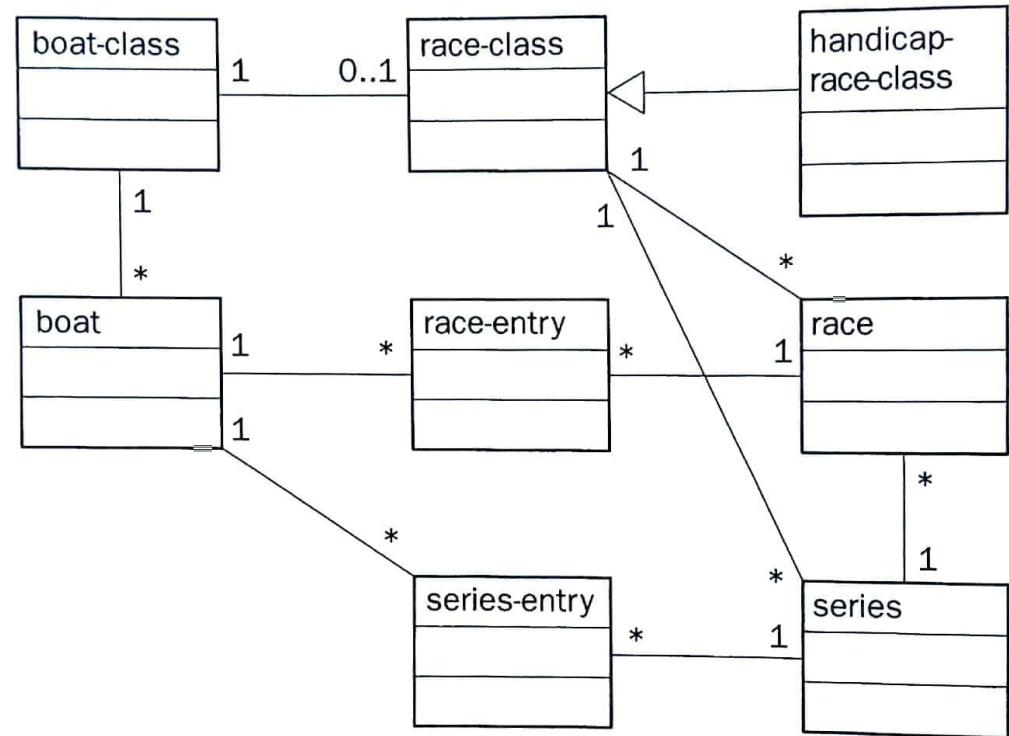
But Data modelling involves **objects**

Difference from ERDs is that there are **actions placed on objects** (detailed during Design phases, not requirements analysis)

Here (in Requirements Analysis), *just define the objects and their attributes*

Objects and their attributes, then relationships

object class	attributes
boat	boat-name, boat-class-name, sail-number, helm
boat class	boat-class-name, handicap-type, handicap-value
race-class	race-class-type, race-class-name,
handicap-race-class	handicap-type, minimum-handicap, maximum-handicap
helm	helm-name
race	race-class-name, race-date, start-time, race-number
race-entry	boat-class-name, sail-number, race-class, race-date, race-time, result
race-officer	
series	series-name, race-class-name, number-of-races, of-races-to-count
series-entry	boat-class-name, sail-number, series-name

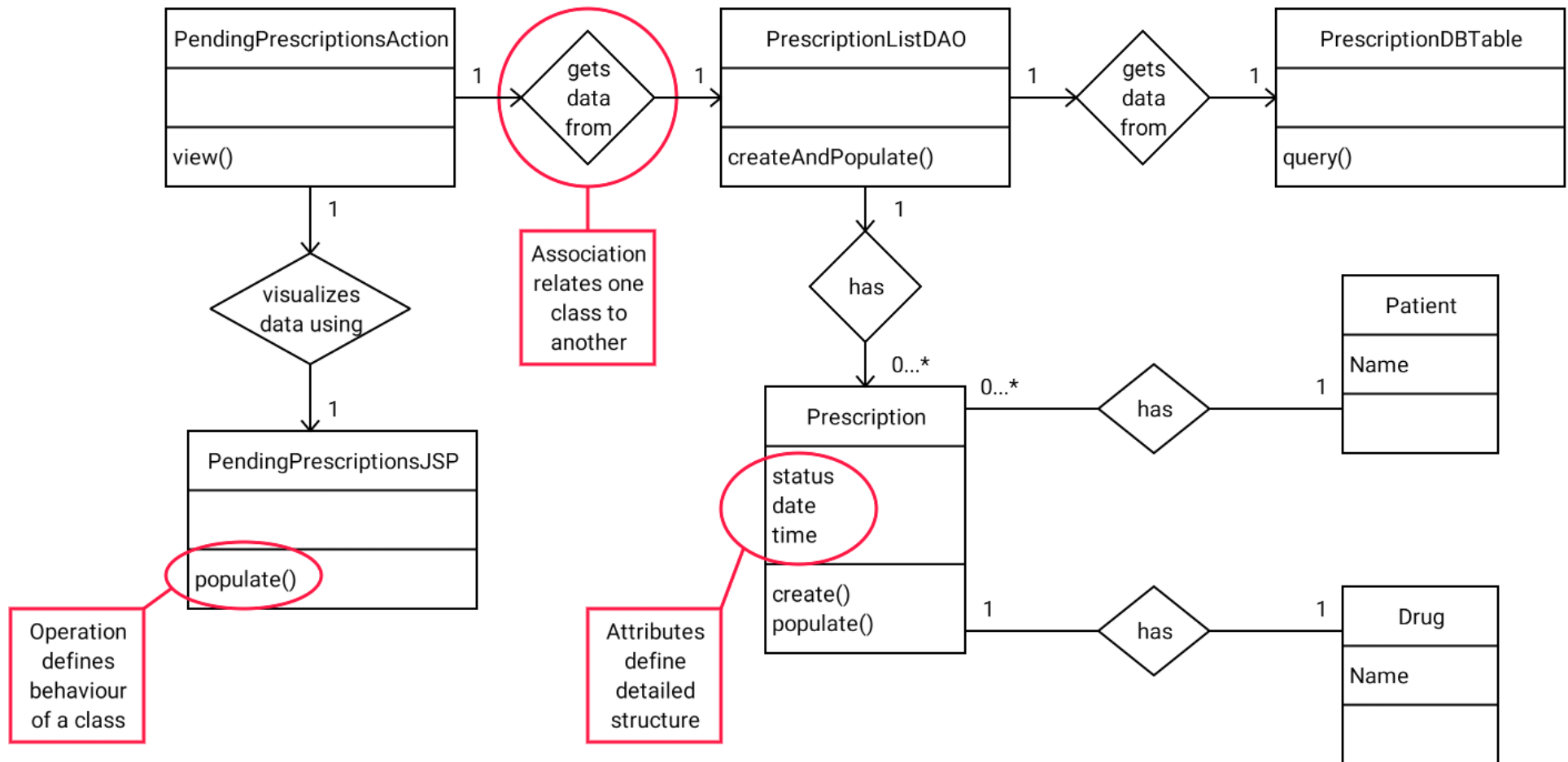


Scenario example

From use case 'Approve Prescription':

1. Retrieve patient pharmacy records
2. Check prescription & review document for errors:
 - A. Check dosage
 - B. Check patient's prescription history
 - C. Check patient record for possible allergies, contraindications, drug interactions
3. Send approved prescription back to pharmacy technician
4. Update patient file

Design Model ER Diagram



High-level Design Sequence Diagram (partial)

