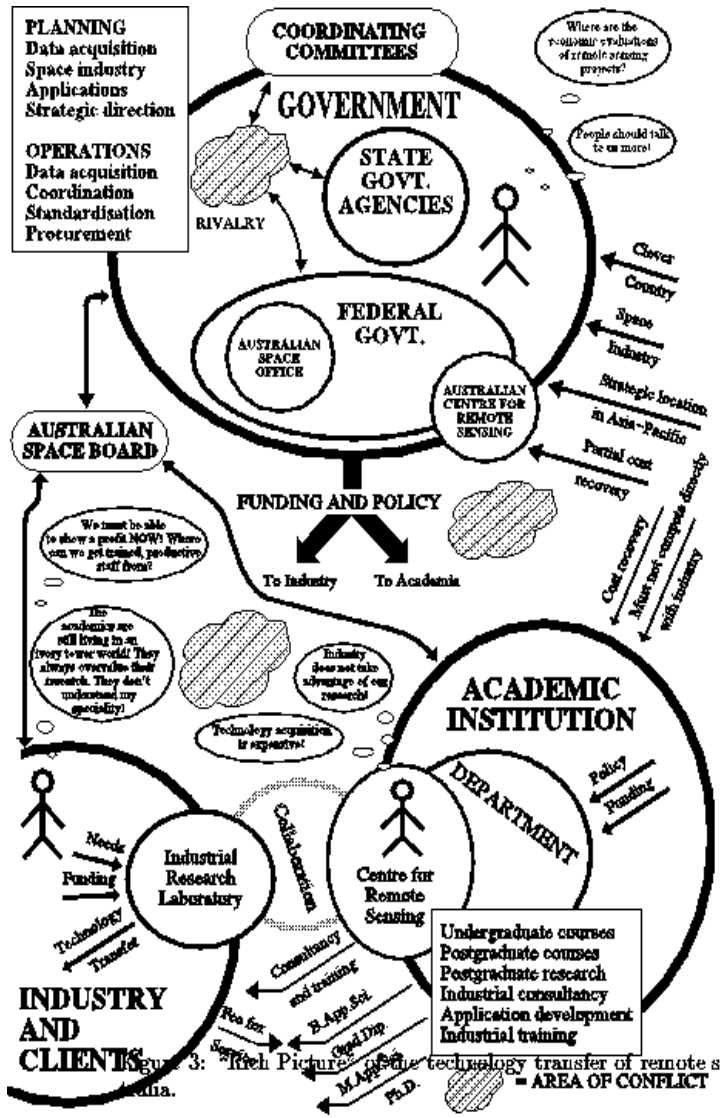


Figure 3: Rich Picture of the technology transfer of remote sensing in Australia.

Software Development

Process, Models,
Methods,
Diagrams
Software Development
Life Cyles

Part - IV

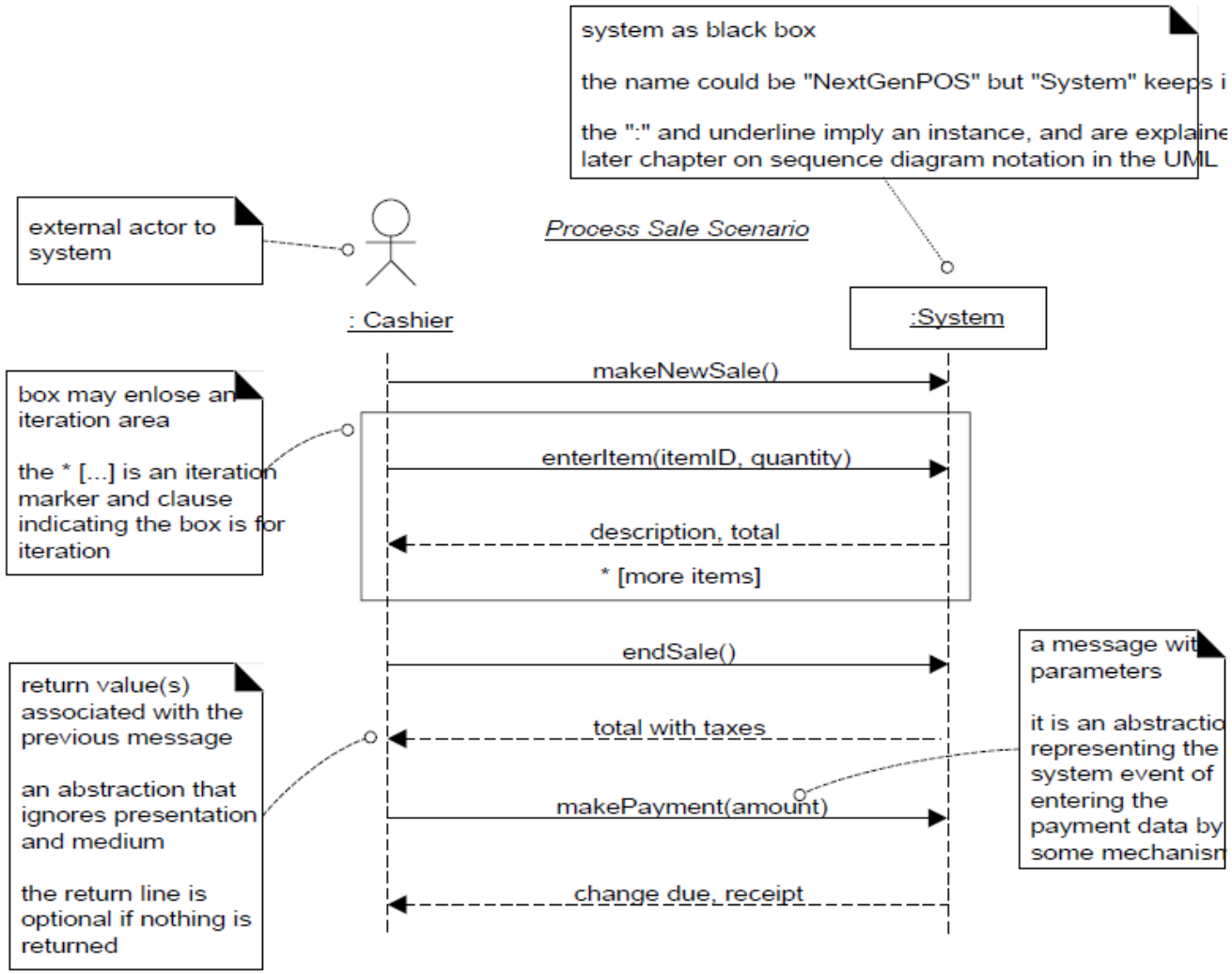
Use-case Model: Drawing System Sequence Diagrams

- A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events

- The terms shown in SSDs (operations, parameters, return data) are terse. These may need proper explanation so that during design work it is clear what is coming in and going out. If this was not explicated in the use cases, the Glossary could be used.

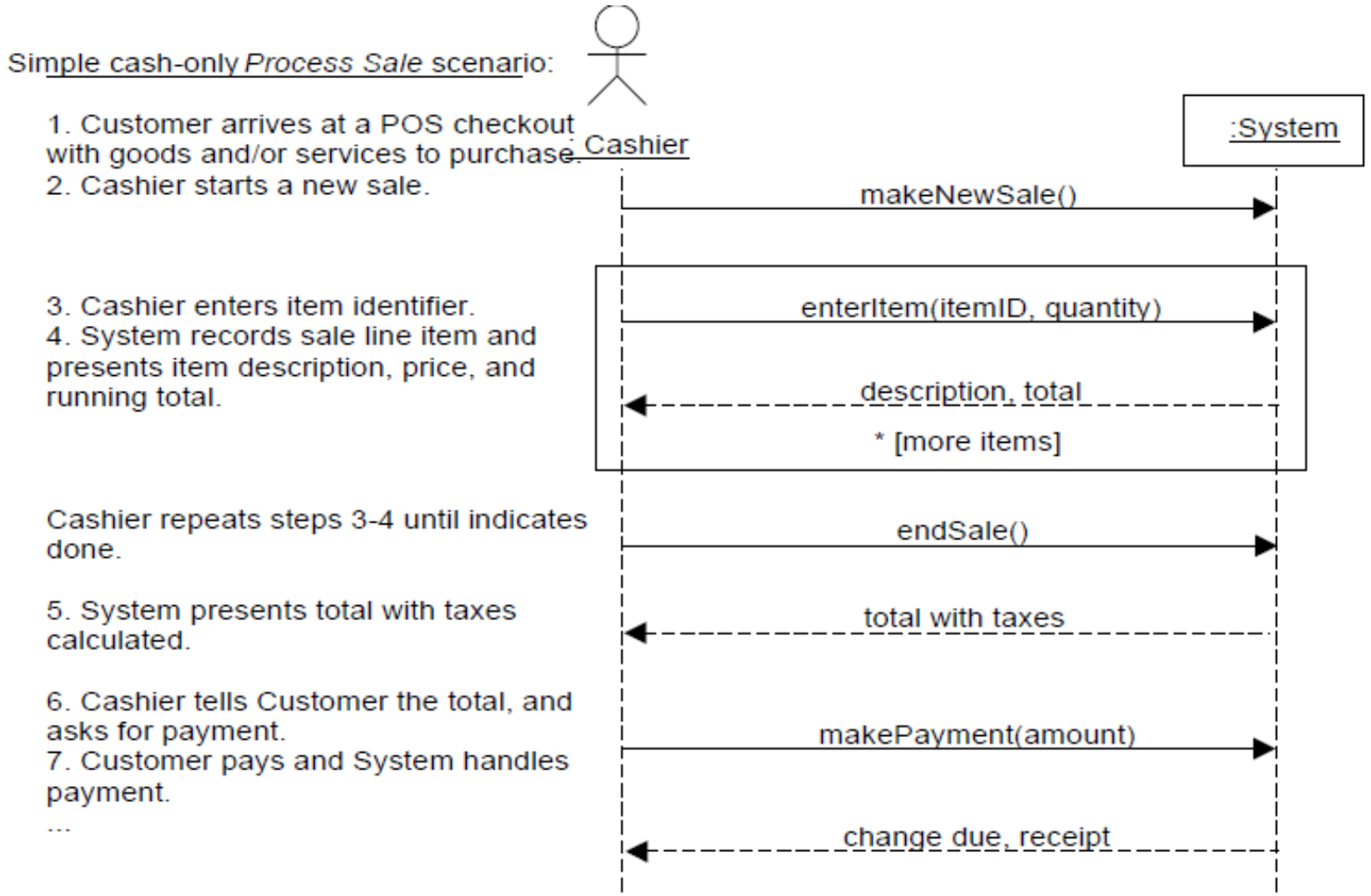
- *Phases*
  - **Inception**—SSDs are not usually motivated in inception.
  - **Elaboration**—Most SSDs are created during elaboration, when it is useful to identify the details of the system events to clarify what major operations the system must be designed to handle, write system operation contracts and possibly support estimation (for example, macroestimation with unadjusted function points and COCOMO II).

Sample



SSD for a *Process Sale* scenario.

SSD with usecase text



SSD with use case text.

DOMAIN MODELVISUALIZING CONCEPTS

- A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest
- Using UML notation, a domain model is illustrated with a set of class diagrams in which no operations are defined. It may show:
    - domain objects or conceptual classes
    - associations between conceptual classes
    - attributes of conceptual classes

- Domain model may be considered a visual dictionary of the noteworthy abstractions, domain vocabulary, and information content of the domain.

Conceptual Class Identification

- In iterative development, one incrementally builds a domain model over several iterations in the elaboration phase
- Strategies to Identify Conceptual Classes

1. Use a conceptual class category list.
2. Identify noun phrases.

Samples for Class Identification

Another useful technique (because of its simplicity) suggested is linguistic analysis: identify the nouns and noun phrases in textual descriptions of a domain, and consider them as candidate conceptual classes or attributes

Main Success Scenario (or Basic Flow):

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale.**
3. **Cashier** enters **item identifier.**
4. System records **sale line item** and presents **item description, price,** and running **total.** Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment.**
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions)** and **Inventory** systems (to update inventory).
9. System presents **receipt.**
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

7a. Paying by cash:

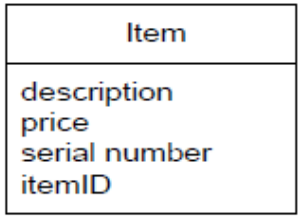
1. Cashier enters the cash **amount tendered.**

Conceptual Class Category	Examples
physical or tangible objects	*Register* *Airplane*
specifications, designs, or descriptions of things	*ProductSpecification* *FlightDescription*
places transactions	*Store* *Airport* *Sale, Payment* *Reservation*
transaction line items	*SalesLineItem*
roles of people containers of other things things in a container	*Cashier* *Pilot* *Store, Bin* *Airplane* *Item* *Passenger*
other computer or electro-mechanical systems external to the system	*CreditPaymentAuthorizationSystem* *AirTrafficControl*
abstract noun concepts organizations	*Hunger* *Acrophobia* *SalesDepartment* *ObjectAirline*
events processes (often *not* represented as a concept, but may be)	*Sale, Payment, Meeting* *Flight, Crash, Landing* *SellingAProduct* *BookingASeat*
rules and policies	*RefundPolicy* *CancellationPolicy*
catalogs	*ProductCatalog* *PartsCatalog*

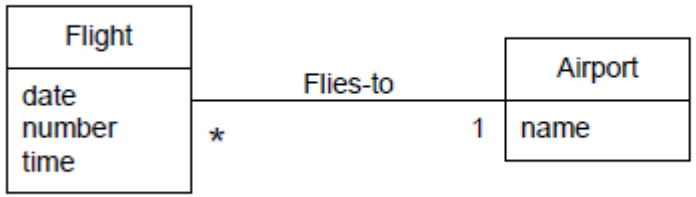
How to make domain model

1. List the candidate conceptual classes using the Conceptual Class Category List and noun phrase identification techniques related to the current requirements under consideration.
2. Draw them in a domain model.
3. Add the associations necessary to record relationships for which there is a need to
4. Add the attributes necessary to fulfill the information requirements

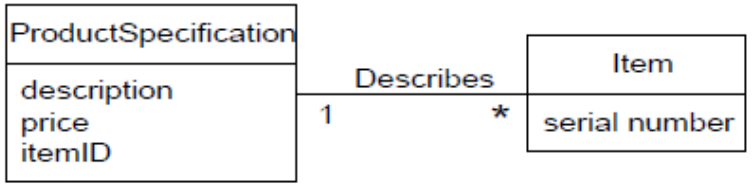
Domain Model



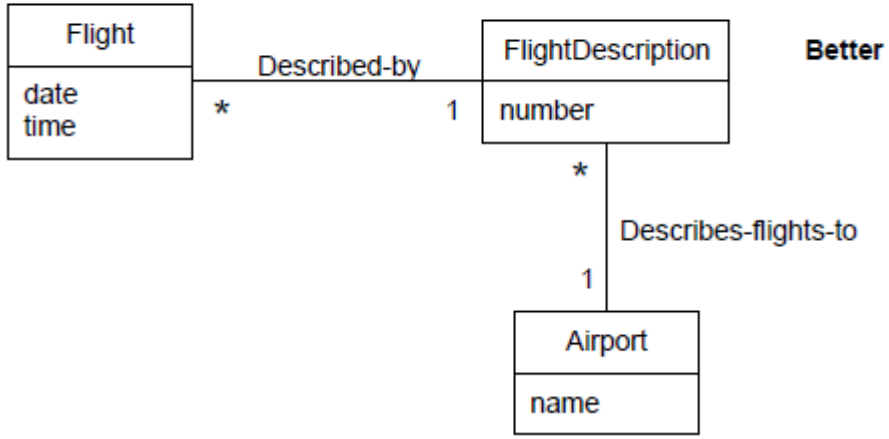
Worse



Worse

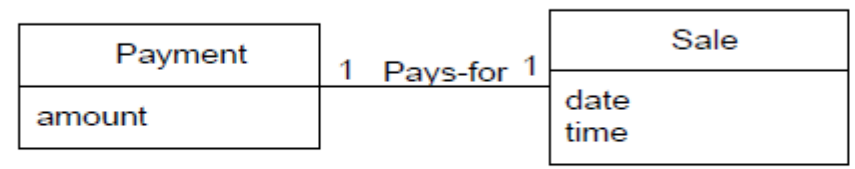


Better



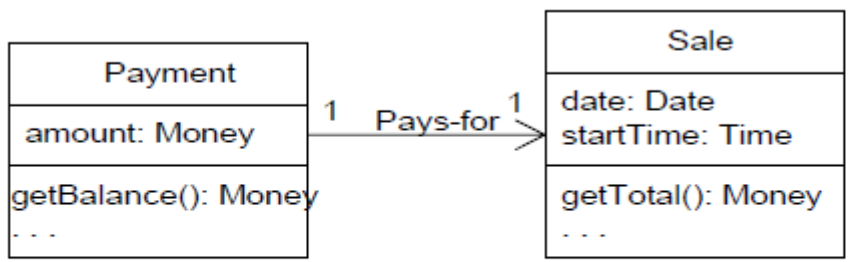
Better

Specifications or descriptions about other things. The "*" means a multiplicity of "many." It indicates that one *ProductSpecification* may describe many (*) *Items*.



UP Domain Model

Raw UML class diagra notation used in an essential model visualizing real-world concepts.



UP Design Model

Raw UML class diagra notation used in a specification model visualizing software components.

Raw UML notation is applied in different perspectives and models defined by a process or method.

Category	Examples
A is a physical part of B	*Drawer — Register (or more specifically, a POST)* *Wing — Airplane*
A is a logical part of B	*SalesLineItem — Sale* *FlightLeg—FlightRoute*
A is physically contained in/on B	*Register — Store, Item — Shelf* *Passenger — Airplane*
A is logically contained in B	*ItemDescription — Catalog* *Flight— FlightSchedule*
A is a description for B	*ItemDescription — Item* *FlightDescription — Flight*
A is a line item of a transaction or report B	*SalesLineItem — Sale* *Maintenance Job — Maintenance-Log*
A is known/logged/recorded/reported/captured in B	*Sale — Register* *Reservation — FlightManifest*
A is a member of B	*Cashier — Store* *Pilot — Airline*
A is an organizational subunit of B	*Department — Store* *Maintenance — Airline*
A uses or manages B	*Cashier — Register* *Pilot — Airplane*
A communicates with B	*Customer — Cashier* *Reservation Agent — Passenger*
A is related to a transaction B	*Customer — Payment* *Passenger — Ticket*
A is a transaction related to another transaction B	*Payment — Sale* *Reservation — Cancellation*
A is next to B	*SalesLineItem — SalesLineItem* *City— City*
A is owned by B	*Register — Store* *Plane — Airline*
A is an event related to B	*Sale — Customer, Sale — Store* *Departure — Flight*

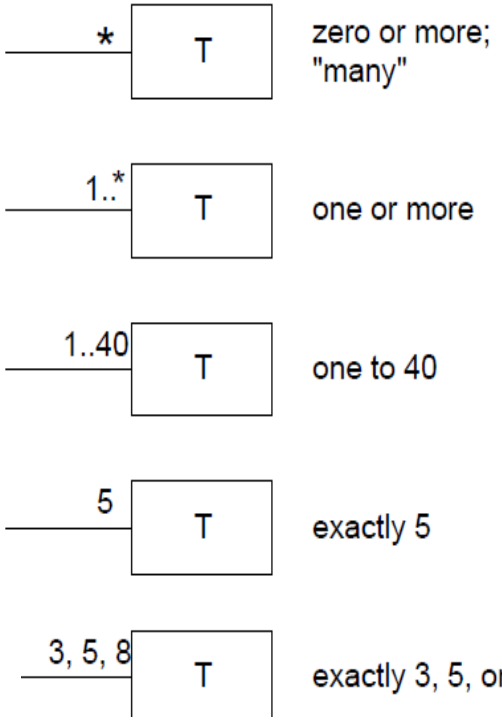
Common Association List

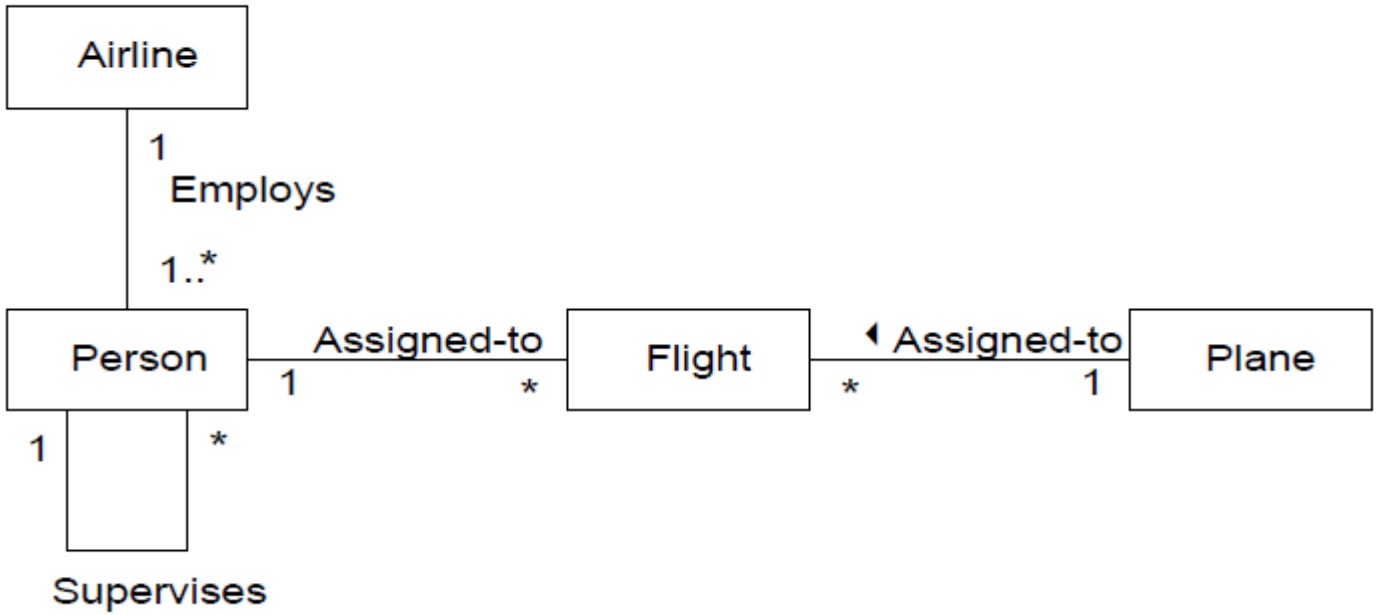
Here are some high-priority association categories that are invariably useful to include in a domain model:

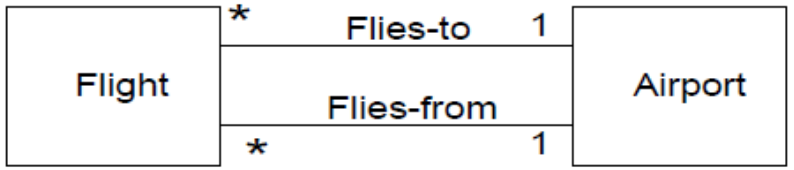
- A is a physical or logical part of B.
- A is physically or logically contained in/on B.
- A is recorded in B.

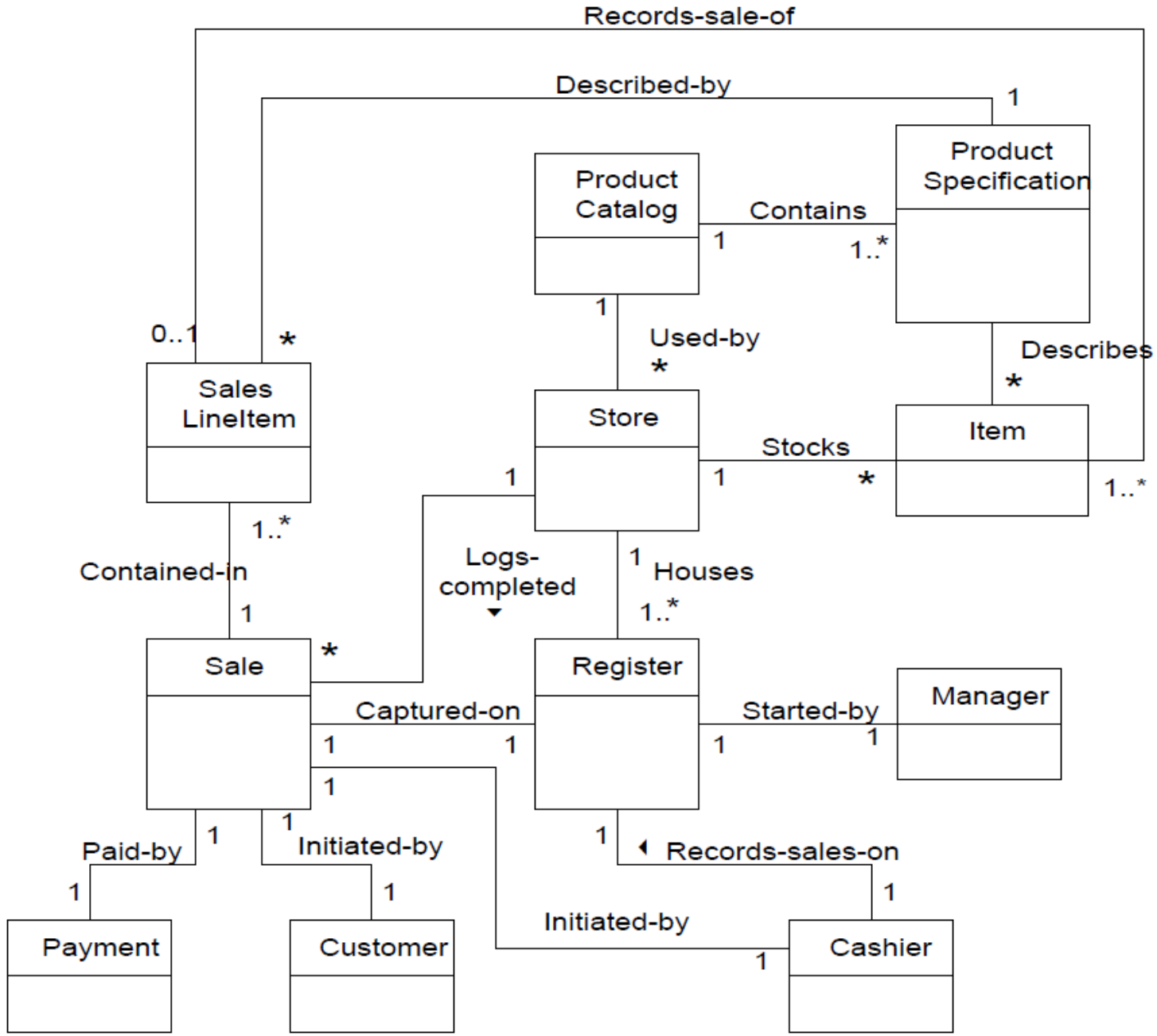
Multiplicities

The multiplicity value communicates how many instances can be validly associated with another, at a particular moment, rather than over a span of time.

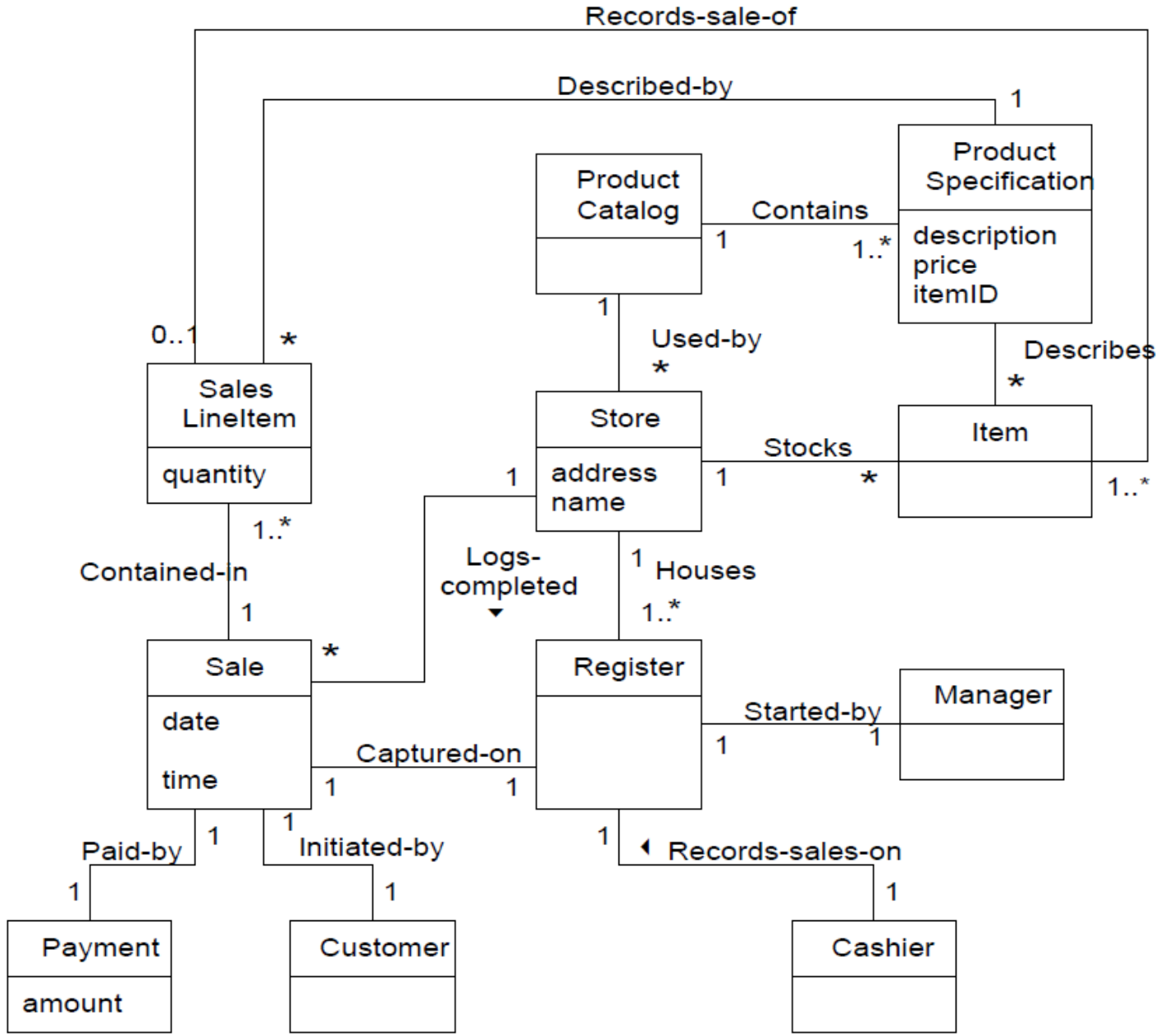








A partial domain model.



A partial domain model.

For the design: Sequence Diagram-200