Software Engineering

STRUCTURED ANALYSIS

History

- 1977
  - Tom de Marco: Structured Analysis and System Specification
  - Gane , Sarson: Structured Systems Analysis
- 1984 – extension
  - McMenamin, Palmer: Essential Structured Analysis
- 1989
  - Yourdon: Modern Structured Analysis
    - integration of Entity-Relationship Models (1976)

Dataflow Diagrams

Terms

- Entity, Terminator
  - source or destination of information
- Process
  - work or task performed on data
- Data Store
  - place where data held between processes
- Data Flow
  - movement of data between above three

Symbols

http://www.cs.mdx.ac.uk/staffpages/sean/teaching/INT1500/BusinessSystems/Lecture02/sld015.htm

Context Diagram

- A Dataflow Diagram where the entire system we want to model is shown as one single process
- Only those external entities which the system shares information with are shown
- The name: noun + verb

Context Diagrams…

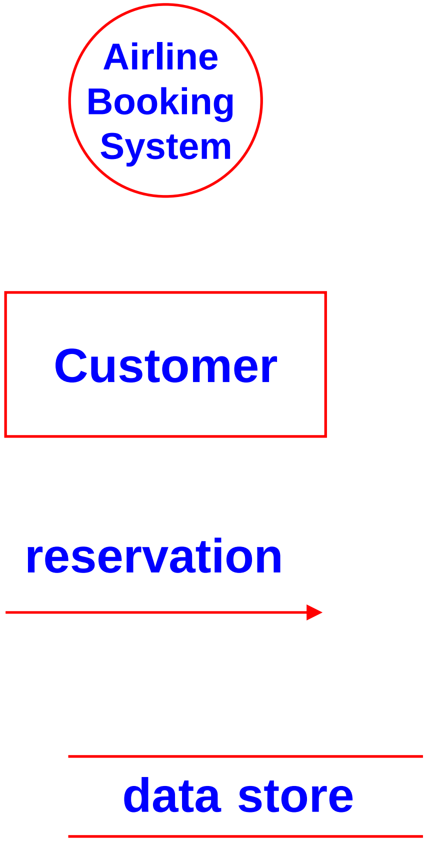
- Show the boundary between the system and the rest of the world
- Indicate the people, organizations and systems which communicate with the system
- Show the data which our system receives from the outside world

Context Diagrams…

- Show the data produced by the system and sent to the outside world
- Show the data which is shared by the system with the outside world

Components

- The process (the system)
- Terminators
  - aka external entities
- Data Flows
- Data Stores



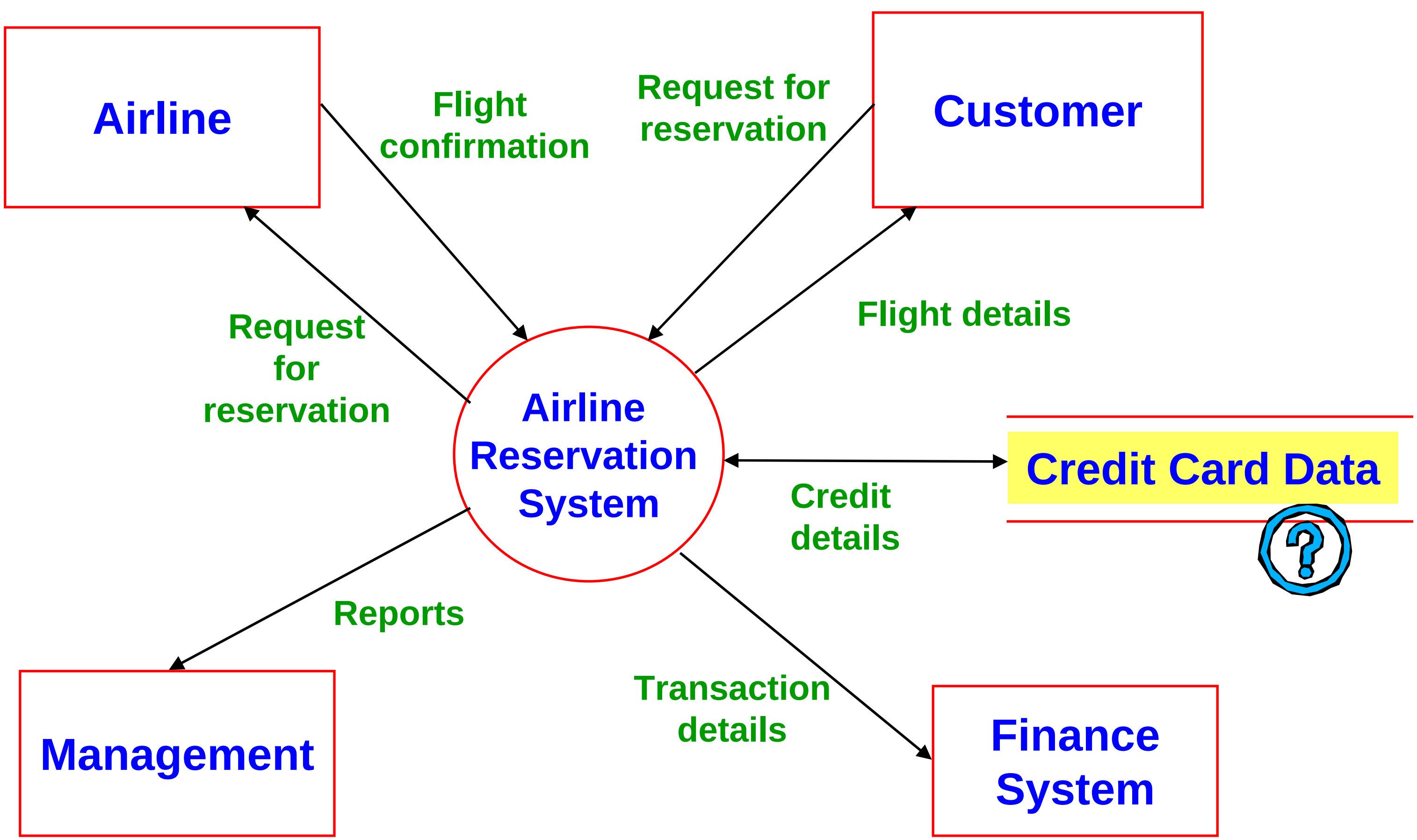
Airline Booking System

Customer

reservation

data store

Reservation System (1)



Airline

Customer

Airline Reservation System

Credit Card Data

Management

Finance System

Flight confirmation

Request for reservation

Request for reservation

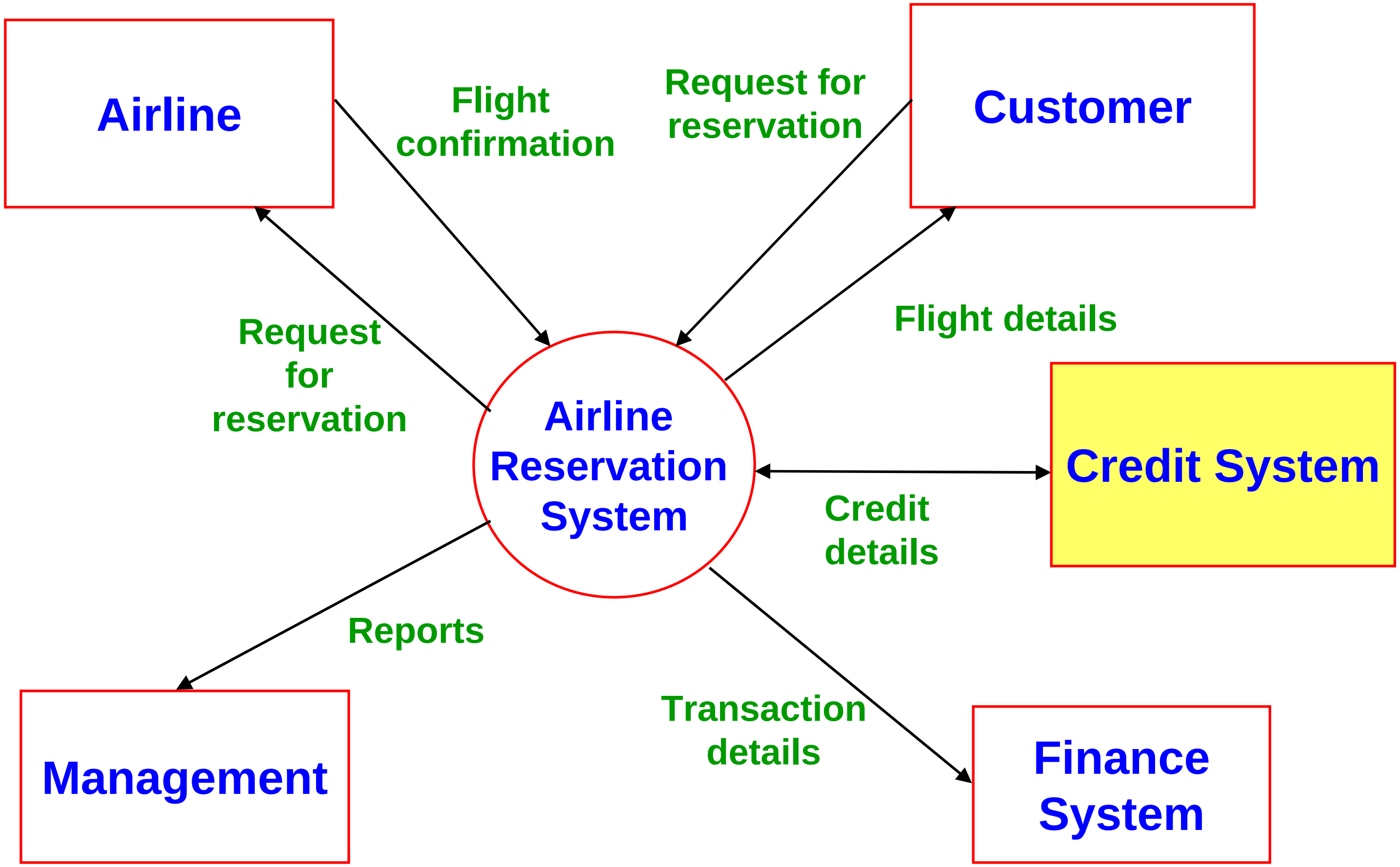
Flight details

Credit details

Reports

Transaction details

Reservation System (2)



Context Diagram: Example

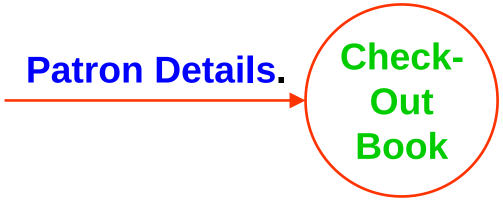
http://www.cs.mdx.ac.uk/staffpages/sean/teaching/INT1500/BusinessSystems/Lecture02/sld009.htm

Data Flow

- Portrays an interface between the components of the dataflow diagram
- A pipeline through which packets of information of known composition flow
- Named to reflect the nature of the data
- All have unique names

Data Flow

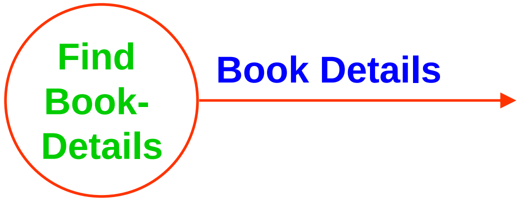
- Indicate movement of information from one part of the system to another part
- Flows are named
  - Input



Patron Details.

Check-Out Book

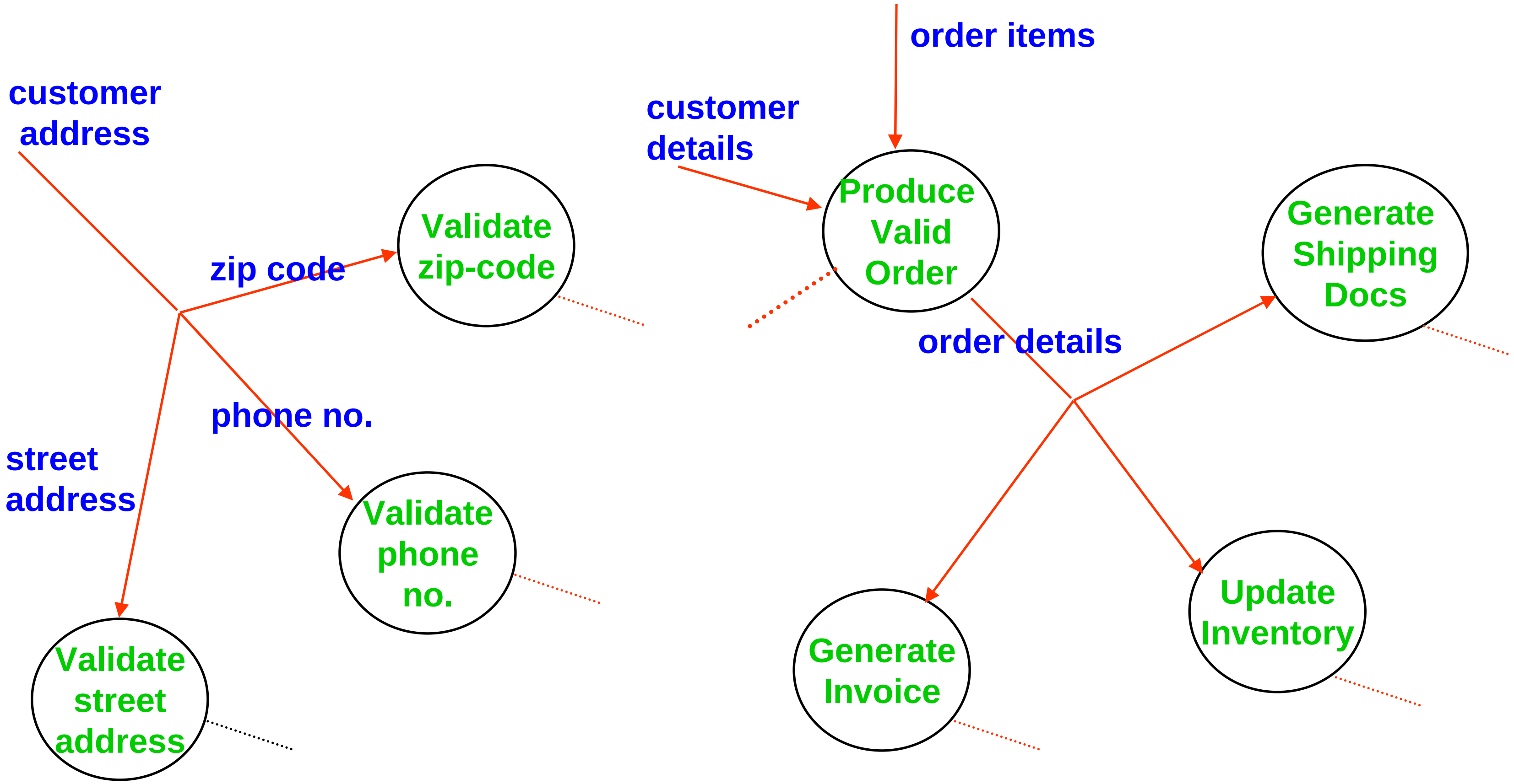
  - Output



Find Book-Details

Book Details

More on Data Flows



Process

- An activity which transforms data
- The name of a process should describe the transformation using only simple verbs and dataflow names
- All have unique names
  - the name: verb + noun

Data Store

- A place where information is put by a process so it can be retrieved later by the same or another process
- In a computer implementation, a data store normally takes the form of a file, a database, or a table
- Names should be kept simple and meaningful

Source or Sink (Terminator)

- An entity outside the context of the system, which either supplies data to it (source) or receives data from it (sink)
- An entity may be both a source and a sink
- A source cannot be directly related to a sink without a process in between

Data Flow Diagram

- Refines the Context Diagram
  - defines the processes which make up a system
- Components
  - Processes
  - Data Flows
  - Data Stores
  - Terminators

} as in context diagram

How to Draw a DFD

- Identify all the external entities (terminators) which act as sources or sinks for the system
- Draw a top level, single process, dataflow diagram which shows the above external entities: context diagram

How to Draw a DFD

- Refine the context diagram by decomposing the single process into several more, maintaining the dataflows with the external entities
- Repeat the above step for each subsequent diagram produced
- Write mini-specs for processes that are not refined

How to Draw a DFD

- There is no algorithm for drawing a DFD but there are heuristics...
  - starting from the sources, ask *what process needs this input?*
  - draw that process, then ask what output does this process produce?
    - gives a clue as to the next processes in a chain
  - repeat the first question to give the identity of the next process
    - connect the processes by a dataflow

How to Draw a DFD

- Chains to graph
  - possibly use the previous questions to produce some chains or fragments, the common links (processes, flows and entities) in the chain can then be collapsed to produce a directed graph

How to Draw a DFD

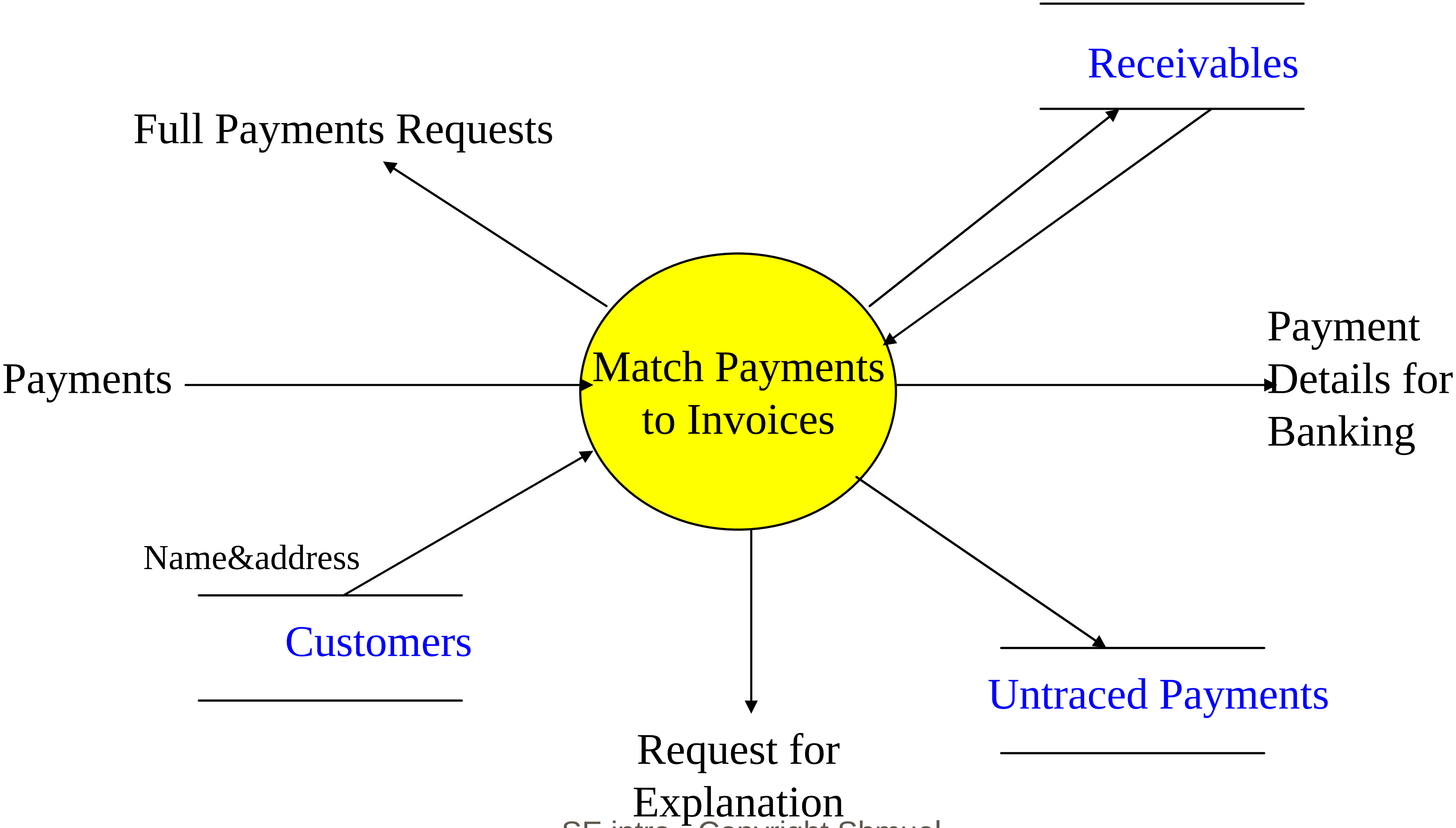
- Do not draw a single layer with more than approximately 5-7 processes
  - avoid overly complex DFDs
- Do not connect a source directly to a sink
- Use meaningful names

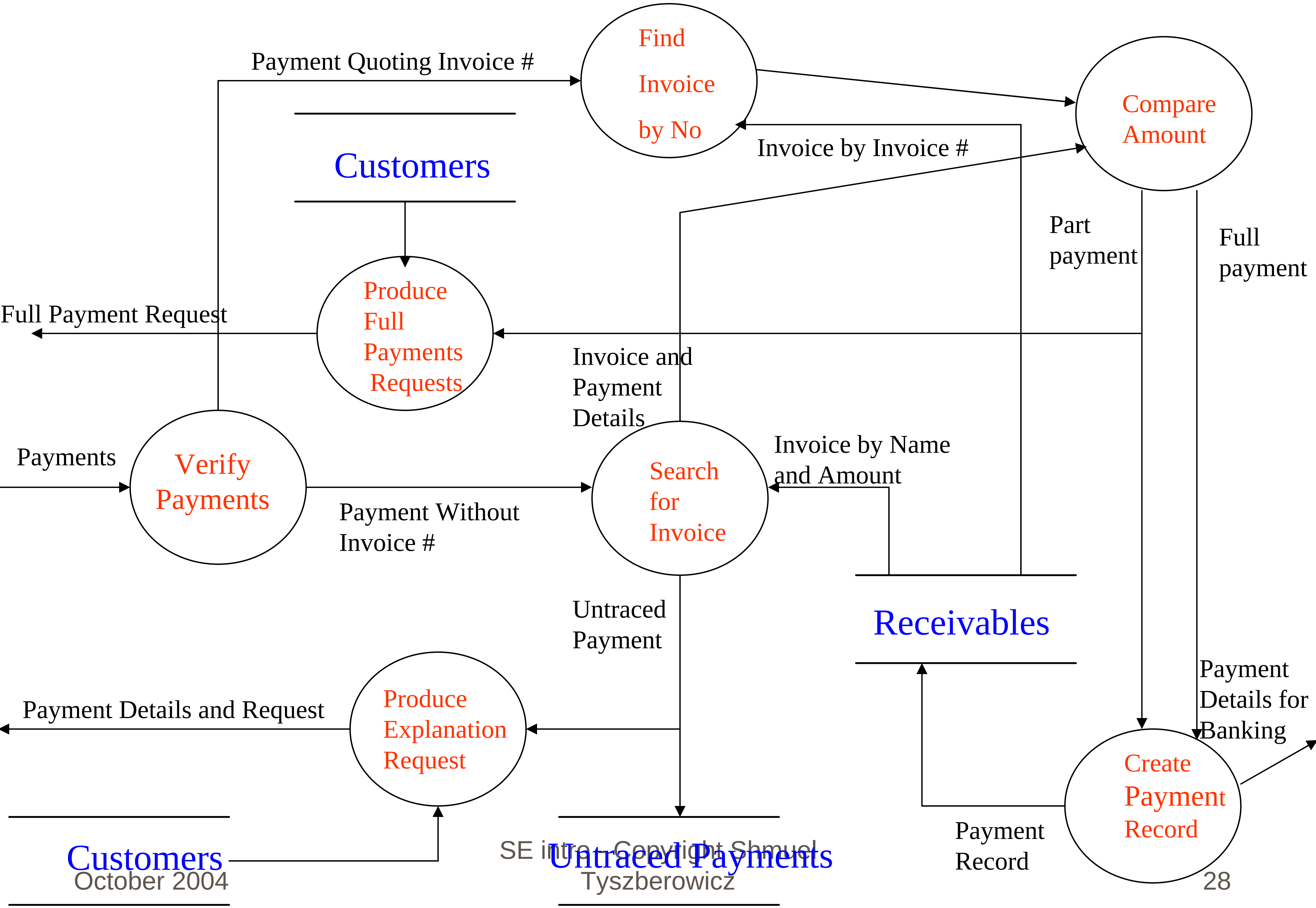
How to Draw a DFD

- Make sure the DFD is internally consistent and consistent with any associated DFDs
  - No blacks holes processes with inputs but no outputs
  - No spontaneous generation processes: processes with outputs but no inputs
    - possible exception is a random number generator
  - Beware of unlabelled flows and processes
  - Beware of read-only/write-only stores
  - Make sure that incoming and outgoing flows from the DFD match those on the DFD at the level above

How To Draw a DFD Example

http://www.cs.mdx.ac.uk/staffpages/sean/teaching/INT1500/BusinessSystems/Lecture02/sld019.htm


Receivables
Full Payments Requests
Payment
Details for
Banking
Payments
Match Payments
to Invoices
Name&address
Customers
Untraced Payments
Request for
Explanation



Payment Quoting Invoice #

Find Invoice by No

Compare Amount

Customers

Invoice by Invoice #

Part payment

Full payment

Produce Full Payments Requests

Full Payment Request

Invoice and Payment Details

Payments

Verify Payments

Payment Without Invoice #

Search for Invoice

Invoice by Name and Amount

Receivables

Untraced Payment

Payment Details for Banking

Payment Details and Request

Produce Explanation Request

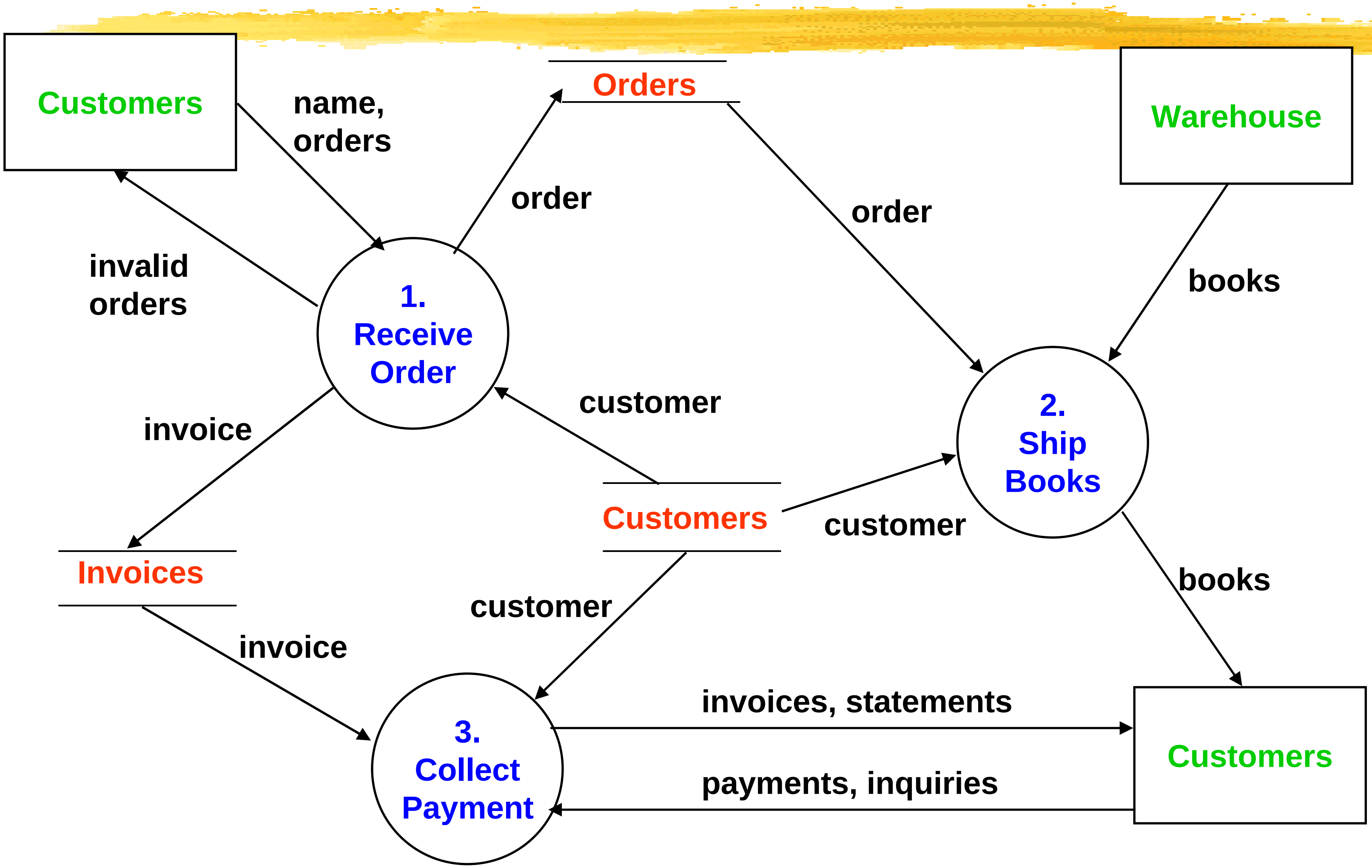
Create Payment Record

Payment Record

Customers

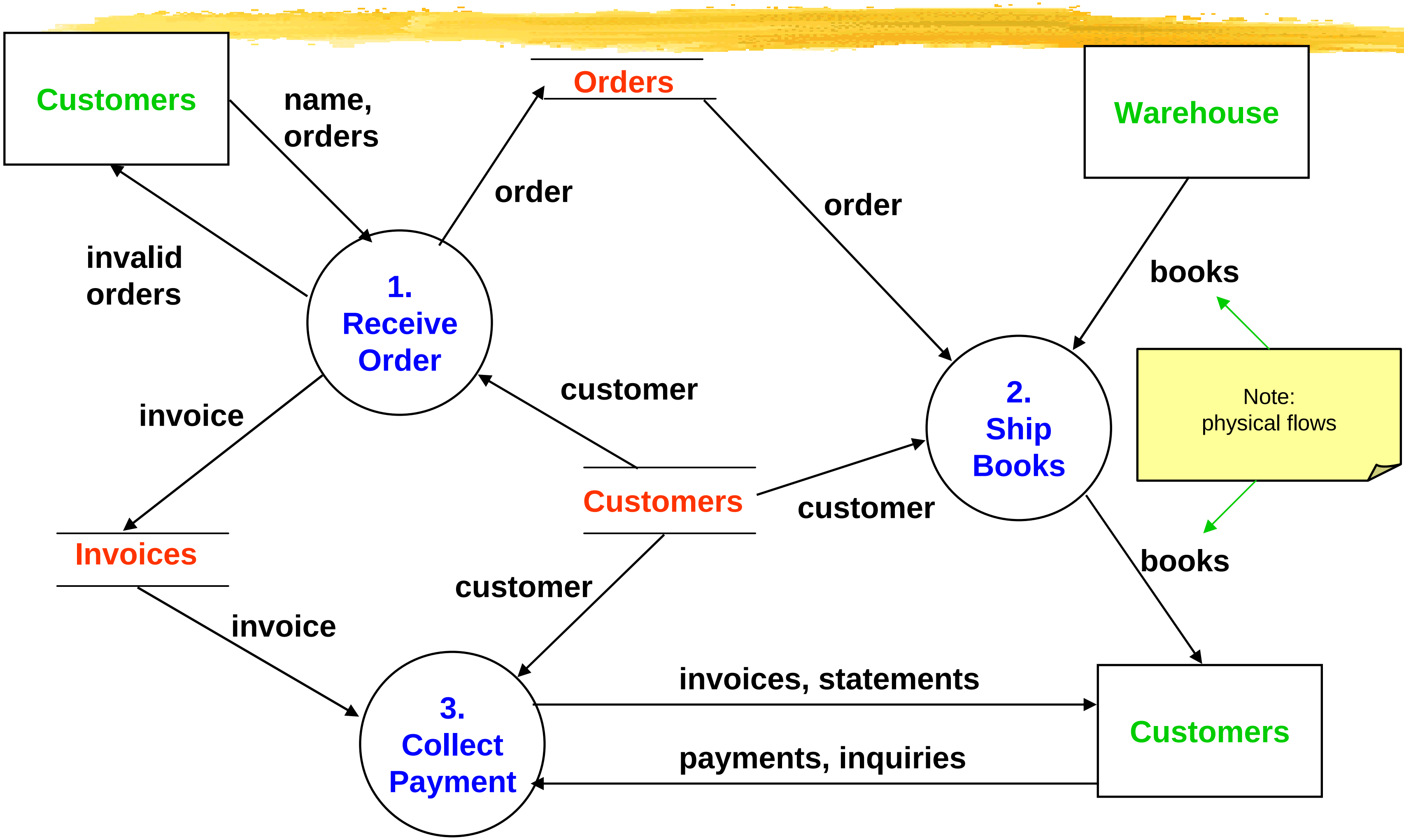
Untraced Payments

DFD-0


Customers
name, orders
Orders
order
Warehouse
order
invalid orders
1. Receive Order
books
customer
2. Ship Books
invoice
Customers
customer
Invoices
books
customer
invoice
invoices, statements
3. Collect Payment
Customers
payments, inquiries

DFD-0

Note:
some do not
show terminators on the
Figure 0 DFD



Customers

name,
orders

Orders

Warehouse

order

order

invalid
orders

1.
Receive
Order

books

customer

2.
Ship
Books

Note:
physical flows

invoice

Customers

customer

Invoices

books

customer

invoice

invoices, statements

3.
Collect
Payment

Customers

payments, inquiries

Student Enrolment System

- System
  - Student Enrolment system
- Terminators
  - Student
  - University Management
  - University Staff
- Data Stores
  - Student Results

Student Enrolment System

- Data Flows
  - enrolment details (from student)
  - confirmation of enrolment (to student)
  - payment details (from staff)
  - student lists (to staff)
  - student results (from staff to system)
  - student results (from Student Results database to system)
  - reports (to management)

Leveling a DFD

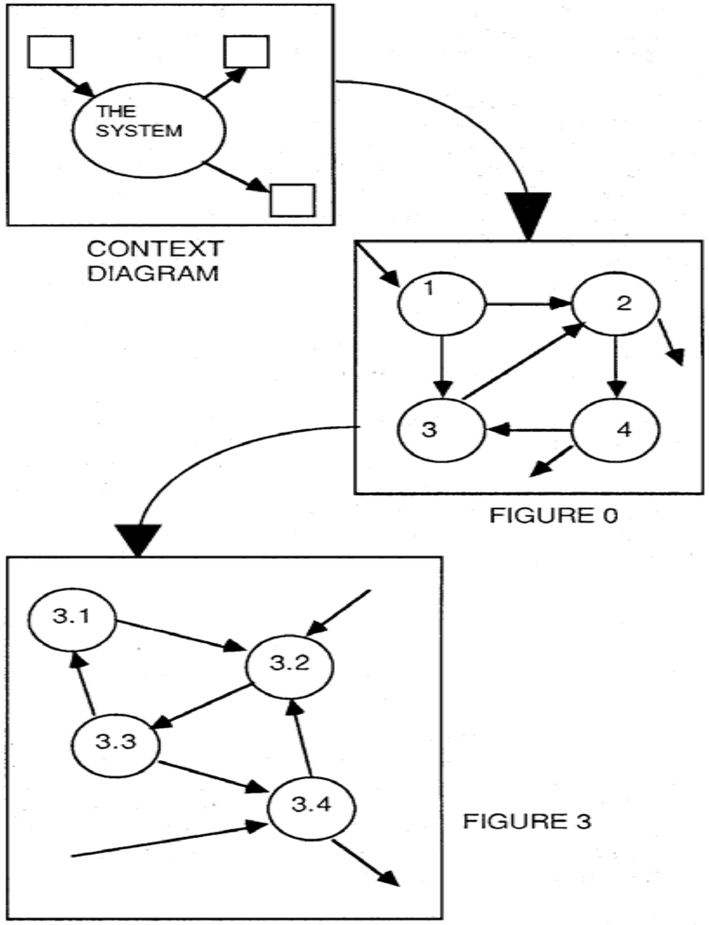
- Most real-life systems are too complex to represent as a single DFD

Leveling a DFD

- Start with a context diagram of the system and the external entities
- Partition into subsystems
- If the sub-systems are too large, divide them into sub-sub-systems
- Repeat until we have DFDs which only contain primitive (indivisible) processes

Leveling a DFD

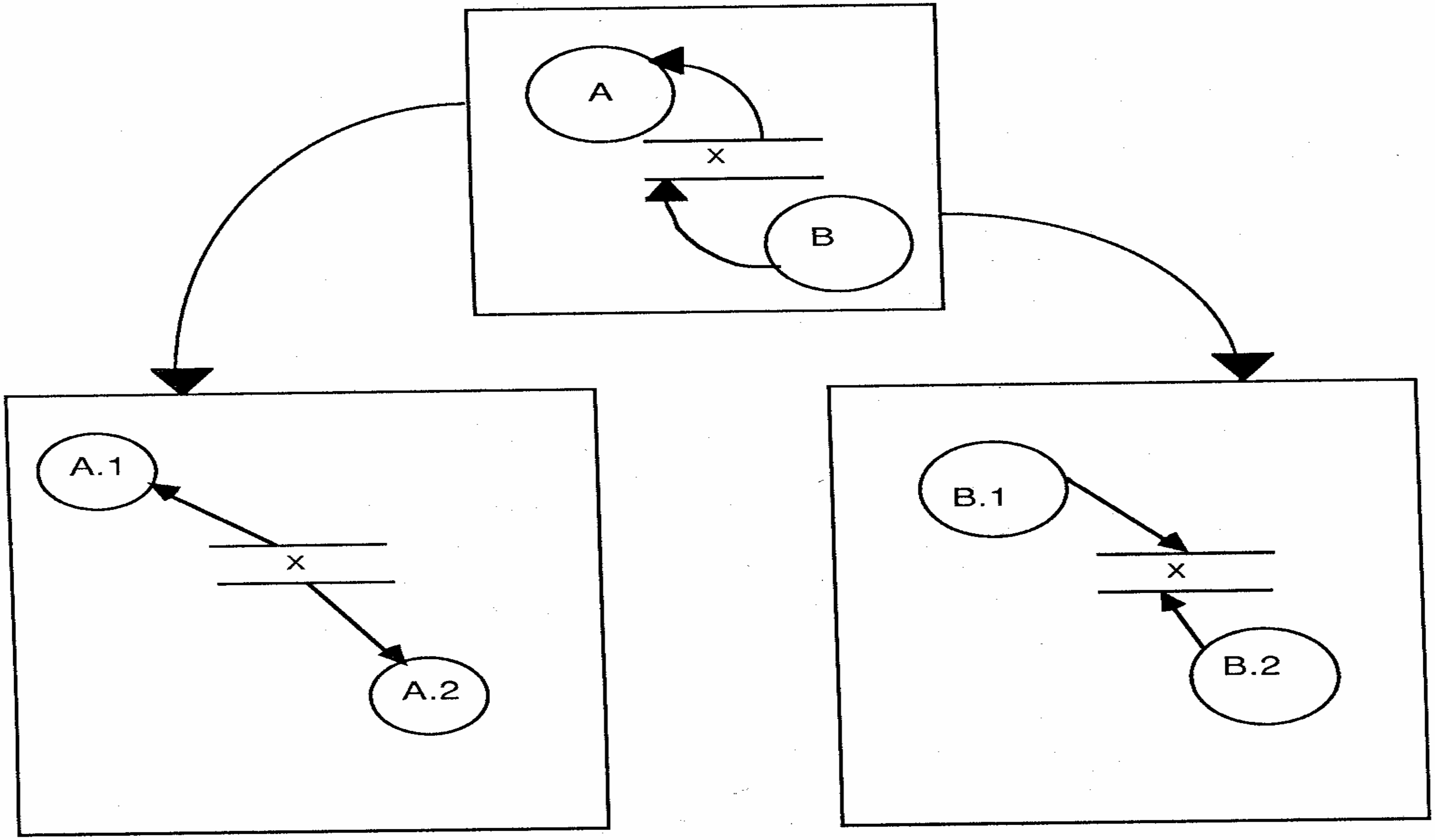
- Numbering of processes indicates their position in the hierarchy of levelled DFDs



Data Stores and Leveling a DFD

- Show the data store at all relevant levels

Data Stores and Leveling a DFD



Validating a DFD

- Does each process precisely state a transformation?
- Is all the data in the system shown?
  - show only data on which a process acts
  - do not show internal data structures
- When a process has been refined into more sub-processes, does the net input/output match the original?

Validating a DFD

- Do the input data and the process determine the output data?
- Can the diagram be redrawn to make it clearer?
- Input data to a process must not appear as output data from a process
- Do data flows pass through processes to reach other processes?

Examples: DFD Level 1

http://www.goldendawn.com/csci270/lan/lan_dfd_1_0.html

http://www.cs.mdx.ac.uk/staffpages/sean/teaching/INT1500/BusinessSystems/Lecture02/sld010.htm

Examples: Full Functional Model

http://www.albany.edu/acc/gangolly/ssa3.html#context

Entity Relationship Diagram

Entity Relationship Diagram (ERD)

- Simplify the representation of large and complex data storage concepts
- ERDs show entities and relationships between them
  - highlights relationships between data stores directly

Entity Relationship Diagram (ERD)

- ERD is intended primarily for the DB design process by allowing the specification of an *enterprise scheme*
  - this represents the overall logical structure of the DB
  - useful in relational databases systems
- Extended ERD (EERD) shows also attributes

Components of ERD

- Rectangles representing entity sets
- Ellipses representing attributes
- Diamonds representing relationship sets
- Lines linking attributes to entity sets and entity sets to relationship sets

Entity Sets

- A collection of entities whose individual members have the following characteristics:
  - Each can be uniquely identified in some manner
  - Each plays a necessary role in the system under development
  - Each can be described by one or more data elements

Entity Sets

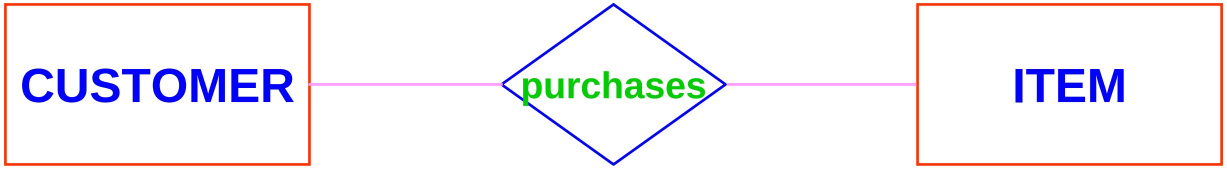
Customer

CHARACTERISTICS

- Has a Customer Number
- Is necessary for a sales system
- Described by elements such as name, address, customer number, phone, number, credit rating etc.

Relationships

- Relationships represent a set of connections between entities



CUSTOMER

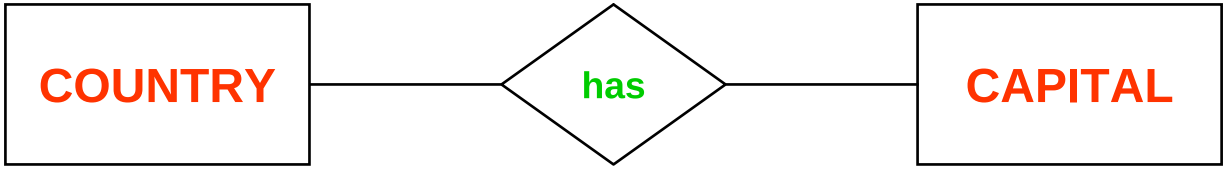
purchases

ITEM

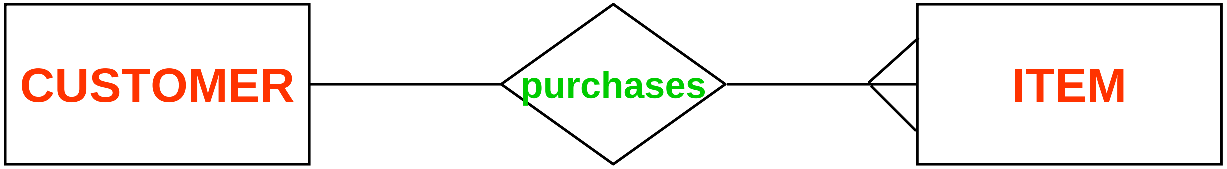
- Relationships have cardinality
  - one to one 1:1
  - one to many 1:n
  - many to one n:1
  - many to many n:m

Relationships

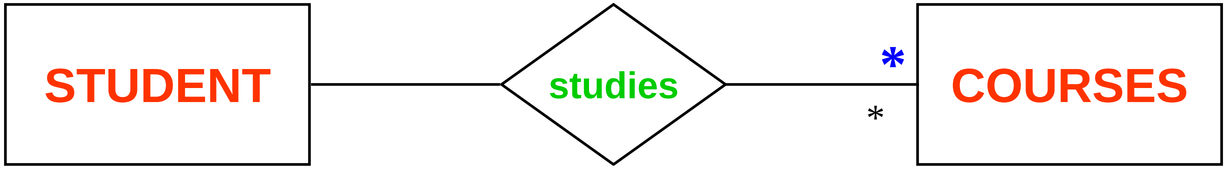
One to One Relationship



One to Many Relationship

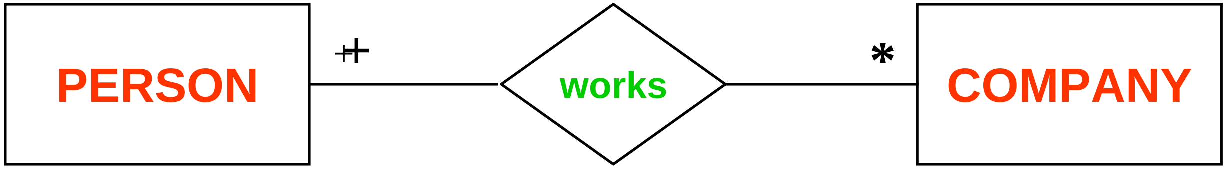


One to Many Relationship

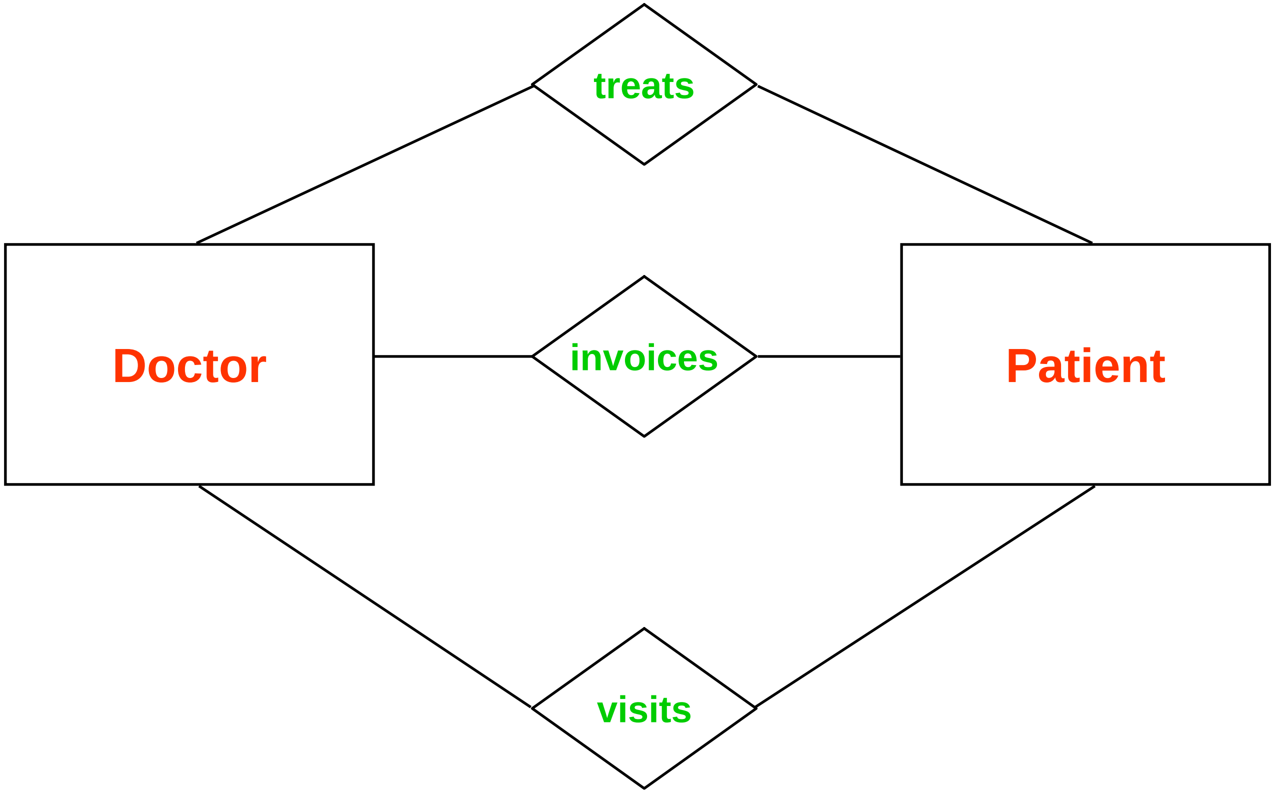


Relationships

Many to Many Relationship

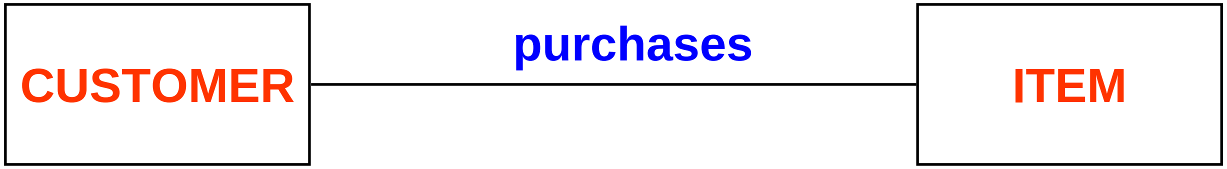


Multiple Relationships



Associative Entities

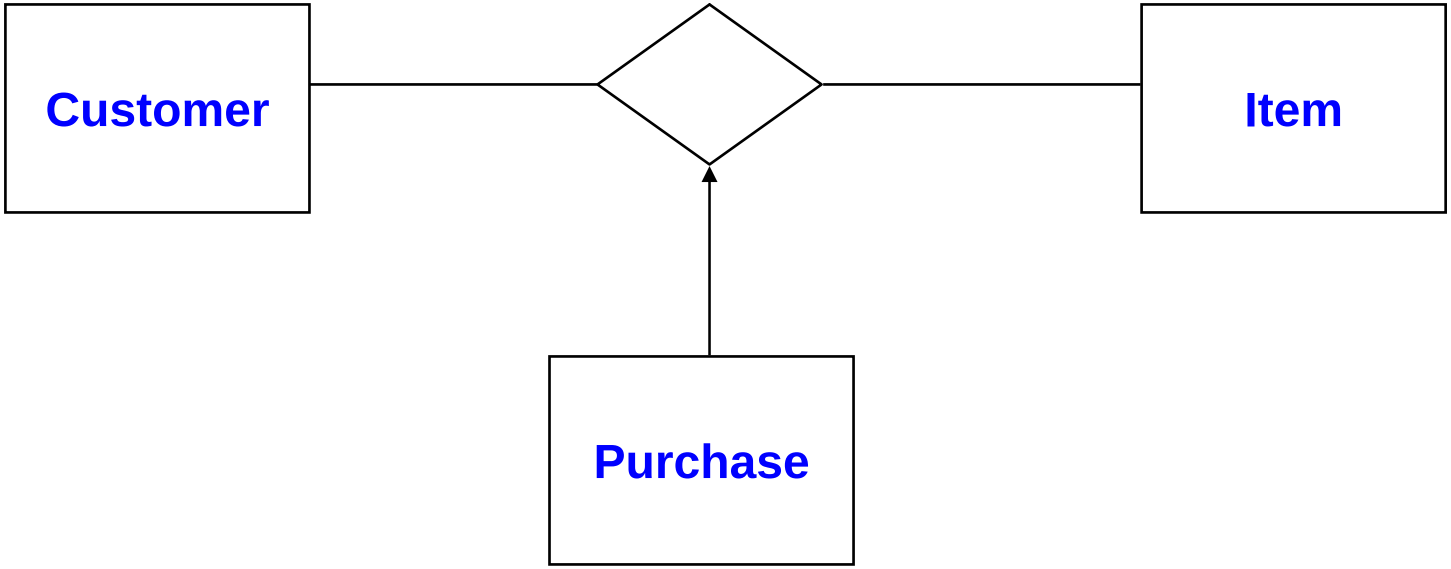
- Sometimes it is necessary to store data about a relationship
  - need to turn the relationship into an entity



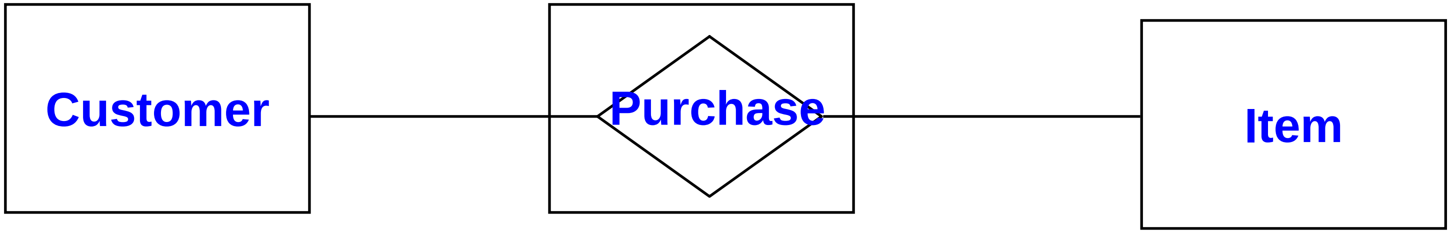
- Keep track of date of purchase, method of purchase, location of purchase
  - none of which fit into the Customer or Item entity

Associative Entities

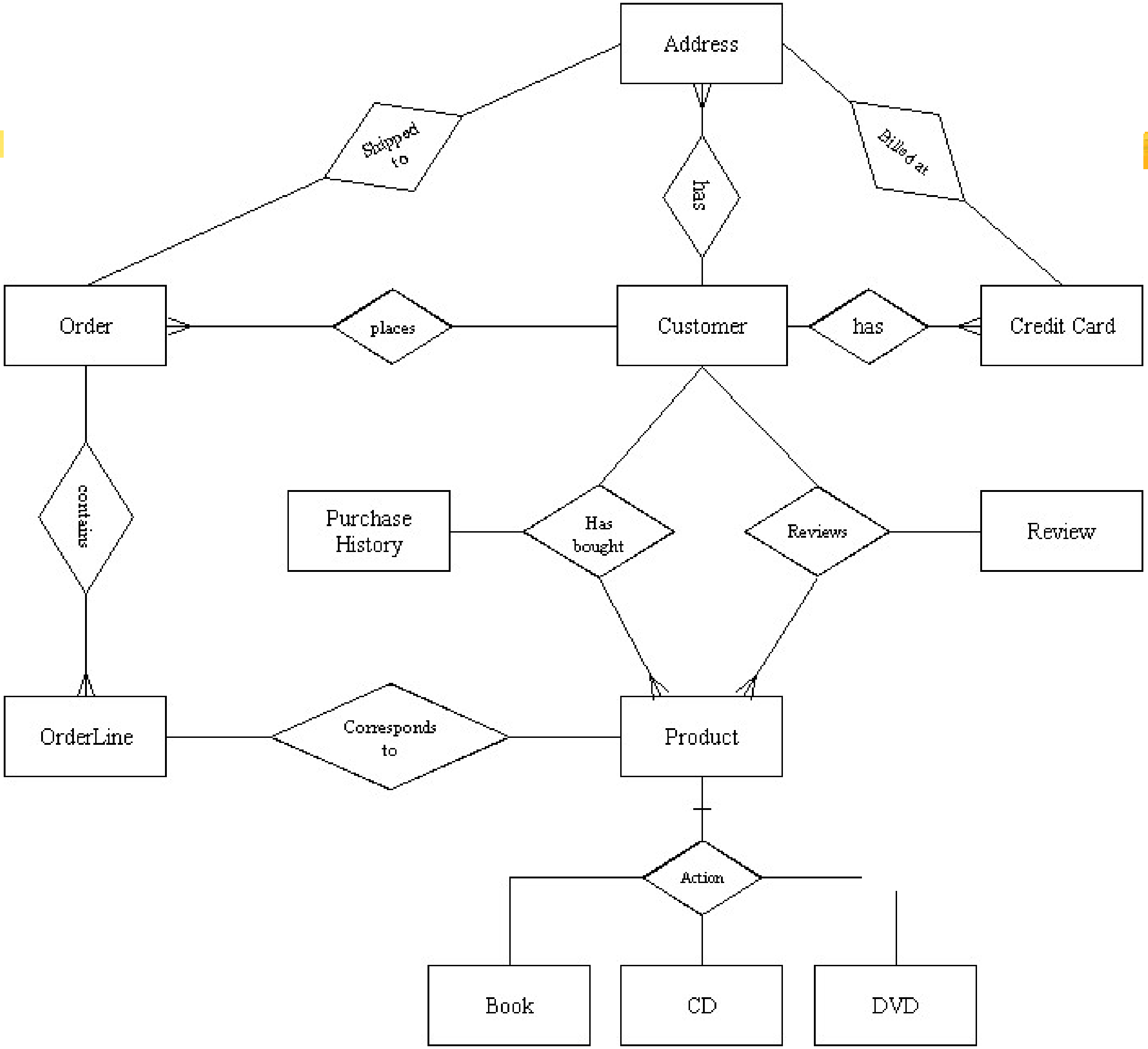
- Create a Purchase entity
- Purchase only exists as a result of the relationship between the other two



Associative Entities



A complete ERD


Address
Shipped to
has
Billed at
Order
places
Customer
has
Credit Card
contains
Purchase History
Has bought
Reviews
Review
OrderLine
Corresponds to
Product
Action
Book
CD
DVD

Examples of ERD

http://www.nsu-cc.northern.edu/schumajb/library/ERdiagram.html

http://www.cs.sfu.ca/CC/354/zaiane/material/notes/Chapter2/node8.html#SECTION00170000000000000000

http://www.csupomona.edu/~llsoe/cis466/samples/erd.htm

Examples of ERD

http://mcs.une.edu.au/~compsad/Overheads/lecture17/sld005.htm

http://mcs.une.edu.au/~compsad/Overheads/lecture17/sld006.htm

http://www.nsu-cc.northern.edu/schumajb/library/ERdiagram.html

Data Dictionary

Data Dictionary

- Notation for representing structure of data items
- Need to express:
  - composition (sequence) - how an item is made up of simpler ones (its attributes)
    - address = street + city + country
  - repetition - items which are repeated in (e.g.) lists, arrays, etc.

Data Dictionary

- selection - values for items chosen from alternatives
- optionality - items which are not always present

Symbol Used

- = means *is defined as*, or *is made up of*
- + means *and*
- {} means zero or more of whatever's inside, i.e. **repetition**
- n{}m means between n and m (inclusive)
- [| |] means **one** of the listed attributes is present

Symbol Used

- () means item inside is **optional**
- " " enclose **literals** (actual values)
- * * enclose **comments** - define meaning of data, informally

Examples

TutorialList = Title + VersionNumber + Date + {TutorialDetails}

TutorialDetails = DayOfWeek + TimeSlot + Room + StudentList

StudentList = {FamilyName + FirstName}

- or…..

TutorialList = Title + VersionNumber + Date + {DayOfWeek + TimeSlot + Room + {FamilyName + FirstName}}

Examples

CoursePlan = DateOfPlan + VersionNumber + Titles + {WeekDetails}

WeekDetails = WeekNo + StartDate + [TeachingWeek |NonTeachingWeek]

NonTeachingWeek = ["admin week"| "induction week"| "student centred learning"|..]

TeachingWeek = 2{LectureDetails}4 + (TutorialDetails) + (PracticalWork)

Examples

LectureDetails = *Description of Lecture Content*

StartDate = Date

DateOfPlan = Date

Date = *date in format "dd-mmm-yy"*

Data Dictionary

- Simple way of describing syntax of composite data
- Meaning or semantics captured informally, by:
  - meaningful names for data
  - comments explaining constraints and usage