**Q1) What are 5 workflows of the iterative and incremental life cycle and what are its aims**

Answer 1

Requirements workflow

* + Determine exactly what the client needs

Analysis workflow

* + Analyze and refine the requirements
    - To achieve the detailed understanding of the requirements essential for developing a software product correctly and maintaining it easily

Design workflow

* + Refine the artifacts of the analysis workflow until the material is in a form that can be implemented by the programmers

Implementation workflow

* + Implement the target software product in the chosen implementation language(s)

Test workflow

* + Testing is carried out in parallel with the other workflows, from the beginning
  + Every developer and maintainer is personally responsible for ensuring that his or her work is correct
  + Once the software professional is convinced that an artifact is correct, it is handed over to the software quality assurance group for independent testing

……………………………………………………………………………………………………………………

**Q2) List 5 fundamental metrics of software process and explain**

Answer 2

Metrics serve as an early warning system for potential problems

Management uses the fundamental metrics to identify problems

More specialized metrics are then utilized to analyze these problems in greater depth

* There are five fundamental metrics

Each must be measured and monitored for each workflow:

1. Size (in lines of code or, better, in a more meaningful metric)

2. Cost (in dollars)

3. Duration (in months)

4. Effort (in person-months)

5. Quality (number of faults detected)

………………………………………………………………………………………………………………

**Q3) Distinguish fault, failure, defect, mistake and error**

Answer 3

A fault is injected into a software product when a human makes a mistake

“An incorrect step, process, or data definition in a computer program”

A failure is the observed incorrect behavior of the software product as a consequence of a fault

The error is the amount by which a result is incorrect

* + “A difference…between a computed result and the correct result”

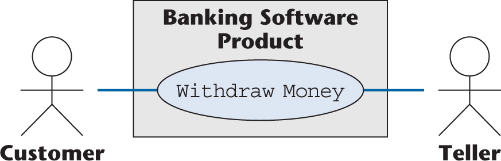
The word defect is a generic term for a fault, failure, or error

……………………………………………………………………………………………………………………………..

**Q5) What is actor? Can software be an actor ? what is use case**

Answer 5

A use case models an interaction between the software product itself and the users of that software product (actors)

* 

An actor is a member of the world outside the software product

It is usually easy to identify an actor

* + An actor is frequently a user of the software product

In general, an actor plays a role with regard to the software product. This role is

* + As a user; or
  + As an initiator; or
  + As someone who plays a critical part in the use case

A user of the system can play more than one role

Example: A customer of the bank can be

* + A **Borrower** or
  + A **Lender**

Conversely, one actor can be a participant in multiple use cases

Example: A **Borrower** may be an actor in

* + The Borrow Money use case;
  + The Pay Interest on Loan use case; and
  + The Repay Loan Principal use case

Also, the actor **Borrower** may stand for many thousands of bank customers

An actor need not be a human being

Example: An e-commerce information system has to interact with the credit card company information system

* + The credit card company information system is an actor from the viewpoint of the e-commerce information system
  + The e-commerce information system is an actor from the viewpoint of the credit card company information system

………………………………………………………………………………………………………………………….

**Q6) Two Types of interviews and explain**

There are two types of questions

* + Close-ended questions require a specific answer
  + Open-ended questions are posed to encourage the person being interviewed to speak out

There are two types of interviews

* + In a structured interview, specific preplanned questions are asked, frequently close-ended
  + In an unstructured interview, questions are posed in response to the answers received, frequently open-ended

………………………………………………………………………………………………………………………………

**Q8) Three kinds of classes in OOA ? how to extract them**

Entity class

* + Models long-lived information

Boundary class

* + Models the interaction between the product and the environment
  + A boundary class is generally associated with input or output

Control class

* + Models complex computations and algorithms

**Finding Entity Objects**

* Some things to look for. These may be candidates for objects, or they may help identify objects:
  + Terms that are domain-specific in use cases
  + Recurring nouns
  + Real-world entities and activities tracked by system
* Use good naming conventions. Good to use names from the application domain -- they understand their own terminology best
* Example: In a ReportEmergency use case -- "A field officer sumits information to the system by filling out a form and pressing the 'Send Report' button"
  + FieldOfficer is a real world entity that interacts with the system
  + This is also likely an actor from the use case
  + As an actor, FieldOfficer is an external entity
  + But we see that the field officer submits information -- here's data to be tracked
  + We'll create the entity object type EmergencyReport, as that's the more common name for the information the officer submits (according to client)

**Finding Boundary Objects**

* Identify general user interface controls that initiate a use case
  + Note: Don't bother with the visual details here. This will evolve later
* Identify forms or windows for entering data into a system
* Identify messages used by system to respond to a user

**Finding Control Objects**

Control objects can help manage communication and interaction of other objects

* If a use case is conplex and involves many objects, create a control object to manage the use case
* Identify one control object per actor involved in a use case
* Life span of control object should last through the use case

…………………………………………………………………………….

**Q9) Strengths and weakness of informal specifications**

* Informal specifications are written in a natural language
  + Examples: English, Mandarin, Kiswahili, Hindi
  + Example

“If the sales for the current month are below the target sales, then a report is to be printed, unless the difference between target sales and actual sales is less than half of the difference between target sales and actual sales in the previous month, or if the difference between target sales and actual sales for the current month is under 5%”

Natural language specifications • Natural language is wrought with ambiguity – Syntax and semantics open to interpretation • Implementation is often difficult – One is never sure of completeness/correctness • Easy to write – Not too much special training required

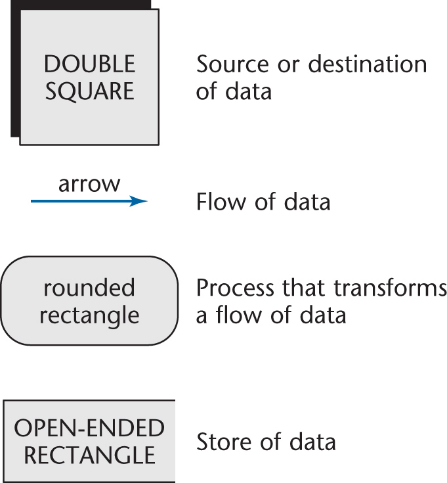
Problems with Informal Specs • Informal specs of any size inevitably suffer from serious problems – Omissions • Something missing – Ambiguities • Something open to multiple interpretations – Contradictions • Spec says “do A” and “do not do A” – Amalgamation • Different requirements mixed together These problems will be faithfully implemented in the software unless found in the spec

…………………………………………………………………………………………………………………………………

Answer 4

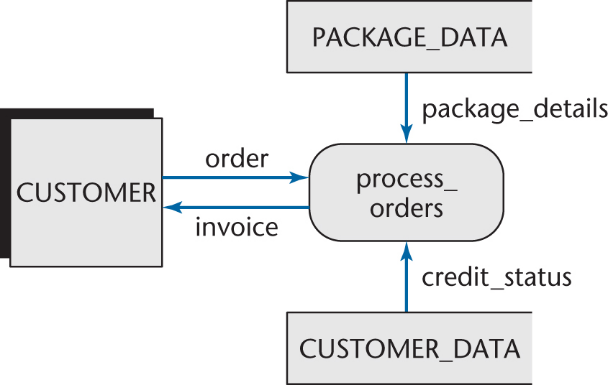
12.3.1  Sally’s Software Shop Mini Case Study

* Sally’s Software Shop buys software from various suppliers and sells it to the public.  Popular software packages are kept in stock, but the rest must be ordered as required. Institutions and corporations are given credit facilities, as are some members of the public.  Sally’s Software Shop is doing well, with a monthly turnover of 300 packages at an average retail cost of $250 each. Despite her business success, Sally has been advised to computerize.  Should she?
* The fundamental issue
  + What is Sally’s objective in computerizing her business?
* We assume: Sally wishes to computerize “in order to make more money”
  + We need to perform cost–benefit analysis for each section of her business
* Gane and Sarsen’s method (i.e., structured systems analysis)
  + Nine-step method
  + Stepwise refinement is used in many steps
* The data flow diagram (DFD) shows the logical data flow
  + “What happens, not how it happens”
* Symbols:

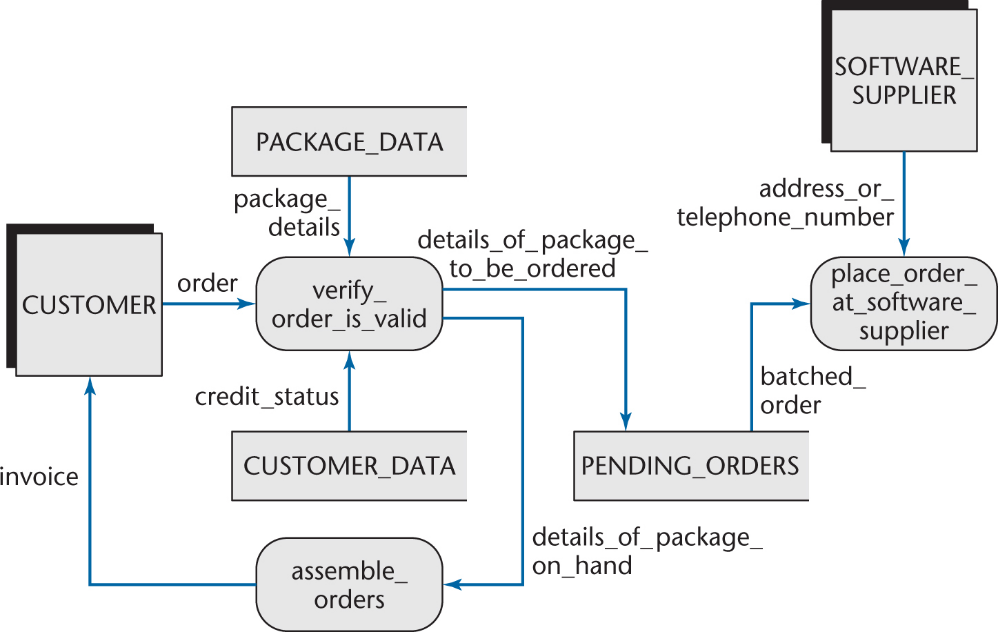


**Step 1.  Draw the DFD**

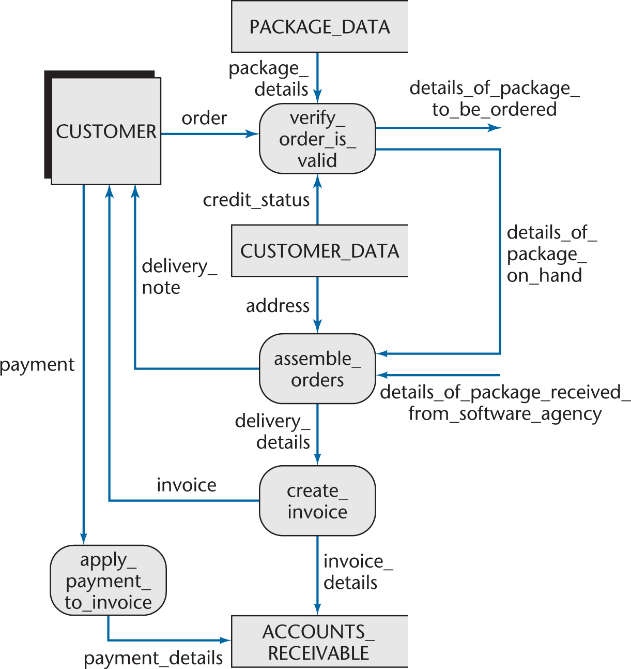
* First refinement
  + Infinite number of possible interpretations



* Second refinement
  + PENDING ORDERS is scanned daily



* Portion of third refinement



* The final DFD is larger
  + But it is easily understood by the client
* When dealing with larger DFDs
  + Set up a hierarchy of DFDs
  + A box becomes a DFD at a lower level

**Step 2. Decide What Parts to Computerize and How**

* It depends on how much client is prepared to spend
* Large volumes, tight controls
  + Batch
* Small volumes, in-house microcomputer
  + Online
* Cost/benefit analysis

**Step 3. Determine the Details of the Data Flows**

* Determine the data items for each data flow
* Refine each flow stepwise
  + Example;

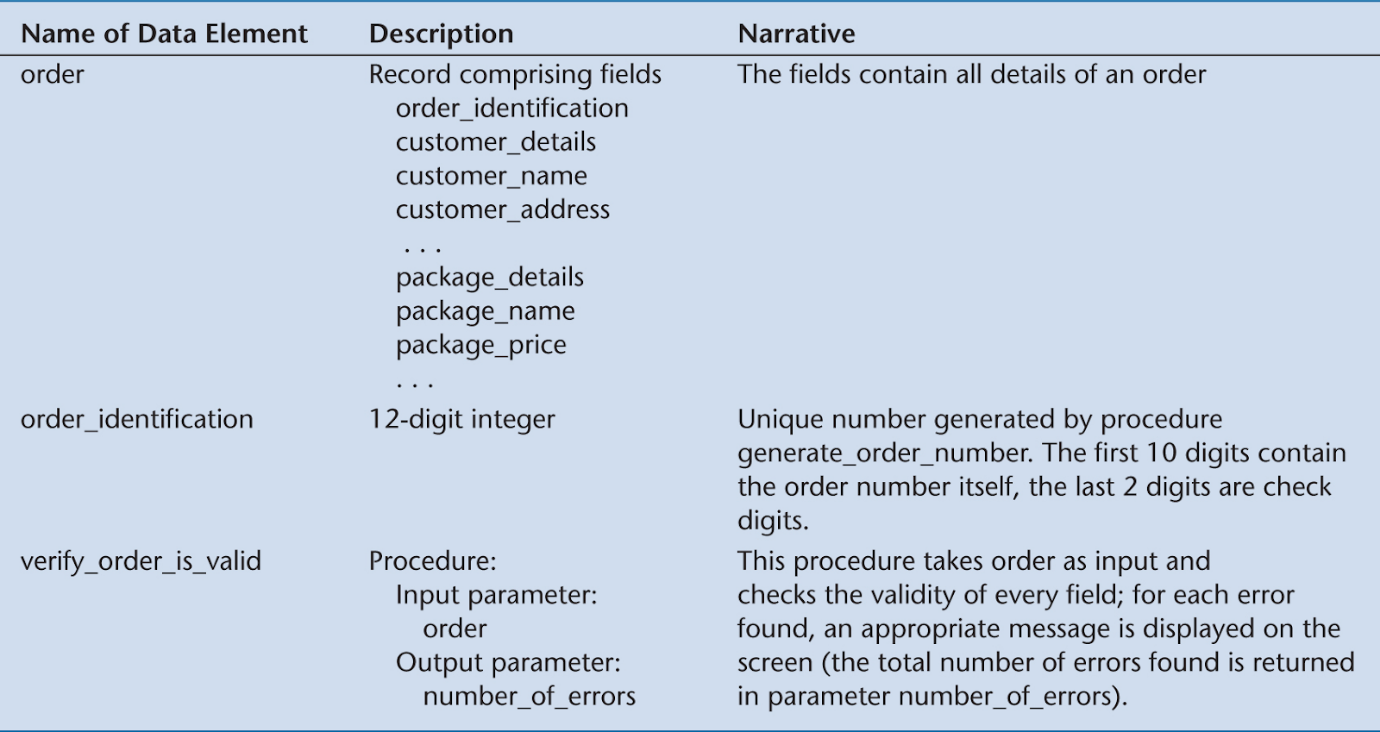
order:

order\_identification

customer\_details

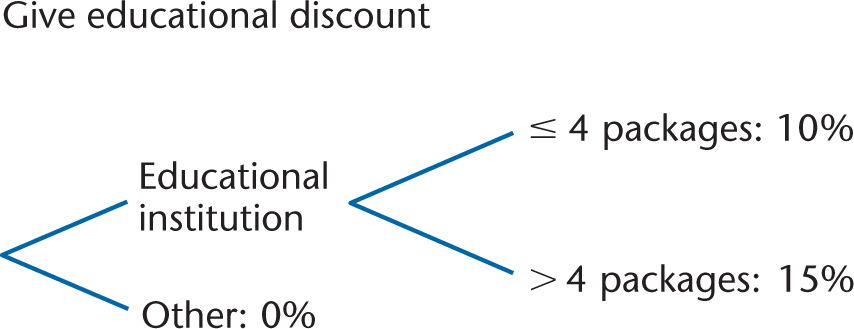
package\_details

* We need a data dictionary for larger products

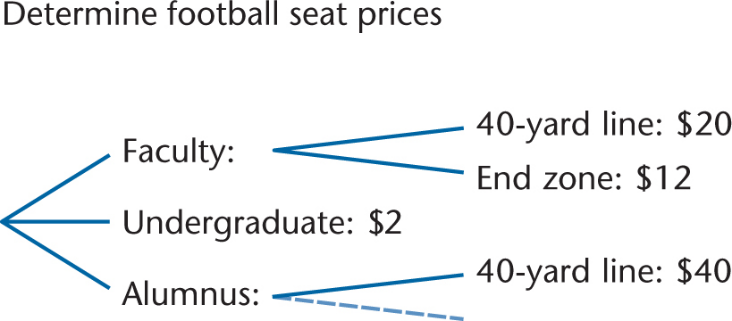
Sample Data Dictionary Entries

**Step 4.   Define the Logic of the Processes**

* We have process give educational discount
  + Sally must explain the discount she gives to educational institutions
    - 10% on up to 4 packages
    - 15% on 5 or more
* Translate this into a decision tree

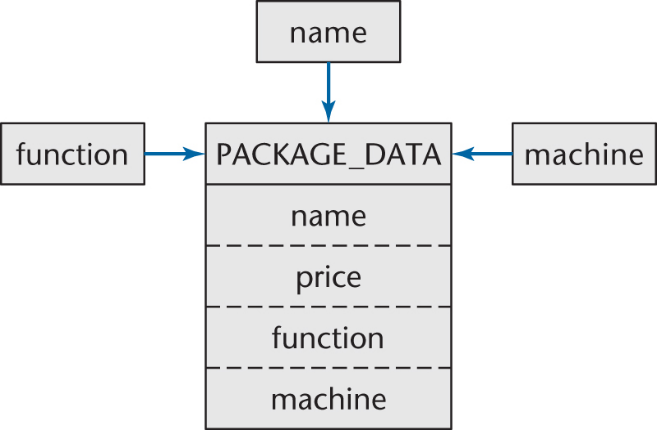


* The advantage of a decision tree
  + Missing items are quickly apparent



**Step 5.  Define the Data Stores**

* Specify where immediate access is required
  + Data immediate-access diagram (DIAD)
  + For example, immediate access to PACKAGE DATA is required by name, function and machine:



**Step 6.  Define the Physical Resources**

* For each file, specify
  + File name
  + Organization (sequential, indexed, etc.)
  + Storage medium
  + Blocking factor
  + Records (to field level)
  + Table information, if a DBMS is to be used

**Step 7.  Determine Input/Output Specifications**

* Specify
  + Input forms
  + Input screens
  + Printed output

**Step 8.  Determine the Sizing**

* Obtain the numerical data needed in Step 9 to determine the hardware requirements
  + Volume of input (daily or hourly)
  + Size, frequency, deadline of each printed report
  + Size, number of records passing between CPU and mass storage
  + Size of each file

**Step 9.  Determine the Hardware Requirements**

* Mass storage requirements
  + Mass storage for back-up
  + Input needs
  + Output devices
  + Is the existing hardware adequate?
  + If not, recommend whether to buy or lease additional hardware

……………………………………………………………………………………………………………

**Q7) Steps of Object Oriented Analysis**

## Three Steps of OOA

### Class Modeling

Purpose: Determine the classes, their attributes and their interrelationships.  
Tool: **Entity-Relationship Diagram (or IC card)**  
End Product: **Class Model Diagram**

### Dynamic Modeling

Purpose: Determine the actions performed by/to each class or subclass.  
Tool: **Finite-State Diagram**  
End Product: **Dynamic Model Diagram**

### Functional Modeling

Purpose: Determine how the various parts of the product interact.  
Tool: **Data Flow Diagram**  
End Product: **Functional Model Diagram**

### The above three steps are iterated, until a satisfactory specification is found.

**Step 1: ITERATE**

**Step 2: until entity classes have been satisfactorily extracted**

**Step 3: extract boundary classes and control classes**

**Step 4: refine the use cases**

**Step 5: perform use case realisation**

