Specification Phase Case Study

Objectives

In this chapter you will:

- Be warned not to use a rapid prototype as the specification document, and not to write a specification document in a natural language such as English.

- Learn about structured system analysis, a graphics–based specification technique.

- Work through a detailed structured systems analysis case study.

- Realize that inspections are an excellent way of checking specification documents.

- The specification document constitutes a major part of the contract between the client who wants the software product built and the developers who are going to build it. Consequently it must be clearly understood by both parties.

- If the specification document is written in a terminology that is unclear to the client — using abstruse mathematics or computer jargon — the client may misunderstand it. Then the delivered product may still satisfy the specification document but the client may not be satisfied $\Longrightarrow$ no further orders, bad propaganda.

- Consequently the specification document must be *clearly and unambiguously understood* by the client.

- If it is unclear to the designers, also problems arise:
  - if it can be interpreted in more than one way (*ambiguity*),
  - if two or more clauses are mutually incompatible (*contradiction*),
  - or if it overlooks some aspect of the product (*omission*).

  This will lead to disputes between the client and the developers as to whether or not the product satisfies its specification – to the detriment of the developers and the client.

- Needed is therefore a mechanism ensuring that the specification document can both be clearly understood by the client and the developers of the product, i.e. that the document is free from ambiguities, contradictions and omissions. To reach this goal a graphical representation of the product will be used, using so–called *structured analysis techniques*.

Two Unsatisfactory Ways of Drawing up the Specification

I. Using Rapid Prototypes

- A rapid prototype is an implementation of only A PORTION OF the functionality of the derived product. However, it is accepted by the client as a way of defining his/her requirements, since it cannot be ambiguous or contradictory (being source code).

- However, it is *incomplete*, and, being a piece of software, cannot serve as a legal document *serving as a contract between client and developers in court*.

$$\Longrightarrow$$

- Consequently a rapid prototype is not suitable for using as specification document (The above reasons are not the only ones. Why this is so: think of it often!)

II. Using Natural Language

- Consider the following specification:

  A part record and an aircraft record are read from the data base. If it contains the number 7 followed by the letter J, then compute the cost of installing that part in that aircraft.

  To what does the "it" in the specification refer to? To the part record, the aircraft record or the data base? $\Longrightarrow$ ambiguity!

- Next consider a natural language specification of part of our C–CC case study:

If the sales for the report month are below the target sales, then a report is to be printed informing the manager that the objective for the month has not been achieved. However, if 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 report month is under 5%, then the report must state that the objective for the report month has been achieved.

- Upon closer inspection, this quite reasonable description turns out to contradict the wishes of Chesterton *in several respects*! For it contains several contradictions and ambiguities with respect to what he told!

  1. Let the target sales for January of some shop be \$100.000, but actual sales be \$64.000, i.e. *36% below target*; consequently the target for that month has not been achieved (or is it, if in December the sales where 75% below target?). Now in February the target figure is \$120.000, and actual sales \$100.00, i.e. *16.7% below target.* The contradiction is now that, *according to these percentages, the objective for that month has been achieved*, because $16.7\% < \frac{1}{2} \times 36$. However, in dollars, the difference between target sales and actual sales was \$36.000 in January, and \$20.000 in February, and $20.000 \not< \frac{1}{2} \times 36.000$, i.e. *according to the paragraph above* the objective has not been reached — *Contradiction*!
  2. In March the target is \$100.00, actual sales \$95.000, and i.e. 5% below target, hence *the objective has not been reached* according to the

paragraph above. However Chesterton said: "*If the shortfall is 5% or less*" the objective is also achieved. *Contradiction*!

- Now one can resort to judicial (lawyer's) language, trying to eliminate such errors. Then *the style is not clear anymore; the paragraph has to be read many times before being understood if this it at all the case* $\Longrightarrow$ an also undesirable situation!

Structured Analysis Applied to the Case Study

- Structured analysis is a nine–steps process for drawing up a specification document.

Step 1. Draw the data flow diagram
Step 2. Decide what sections to computerize
Step 3. Specify the details of the data flows
Step 4. Define the logic of the processes
Step 5. Define the data stores
Step 6. Define the physical resources
Step 7. Determine the input/output specifications
Step 8. Perform the sizing
Step 9. Determine the hardware requirements

Figure 4.1 The nine steps of structured system analysis

- The basis for the structured analysis (SA) technique is a *data flow diagram* that depicts *what* happens in the intended product, as to opposed *how* it happens.

- It is clearly understood, and reduces the chances of ambiguity and contradiction. Its main constituents are shown in Figure 4.2 below and are applied in Figure 4.3 below.

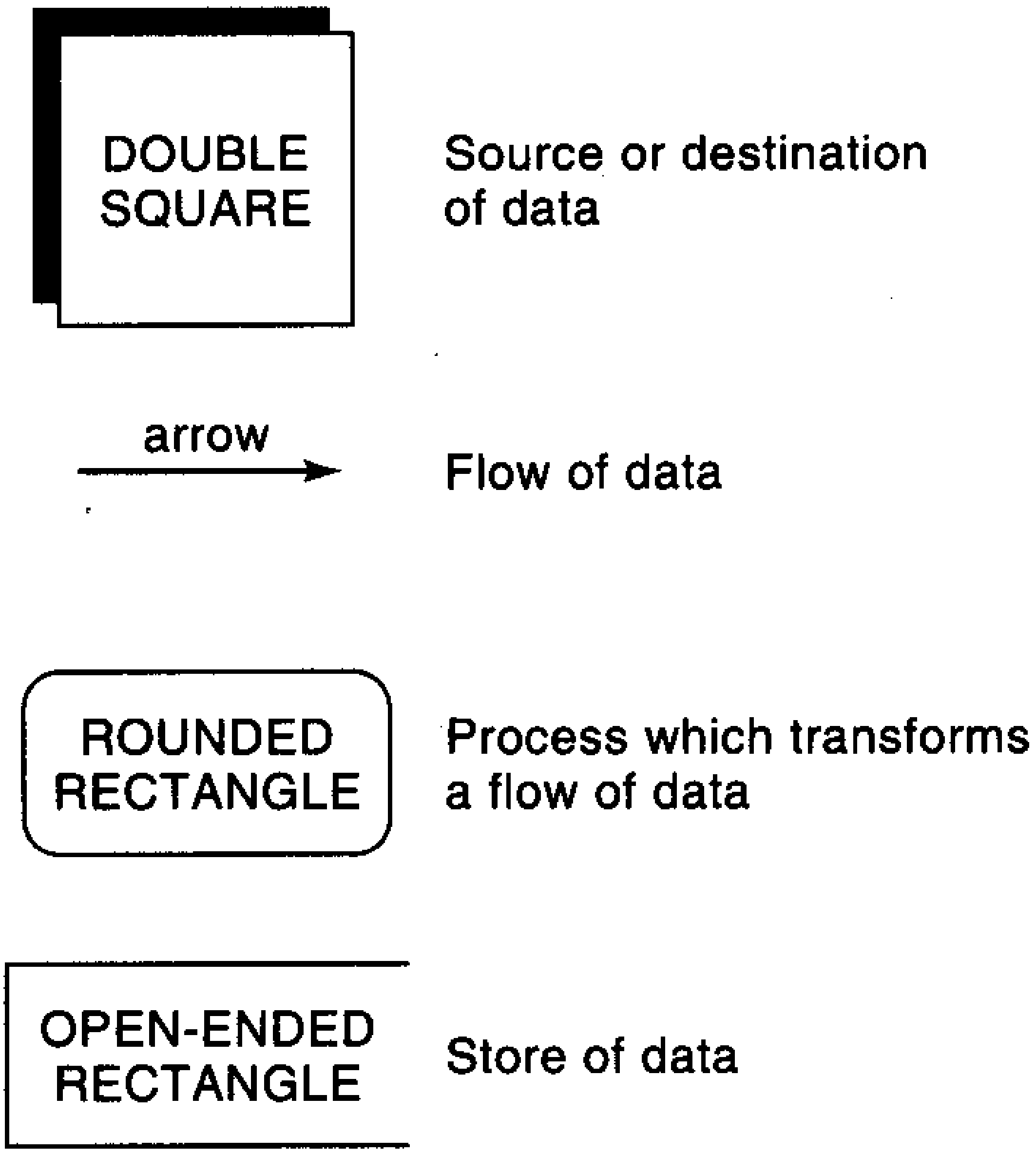

Figure 4.2 Symbols of structured systems analysis

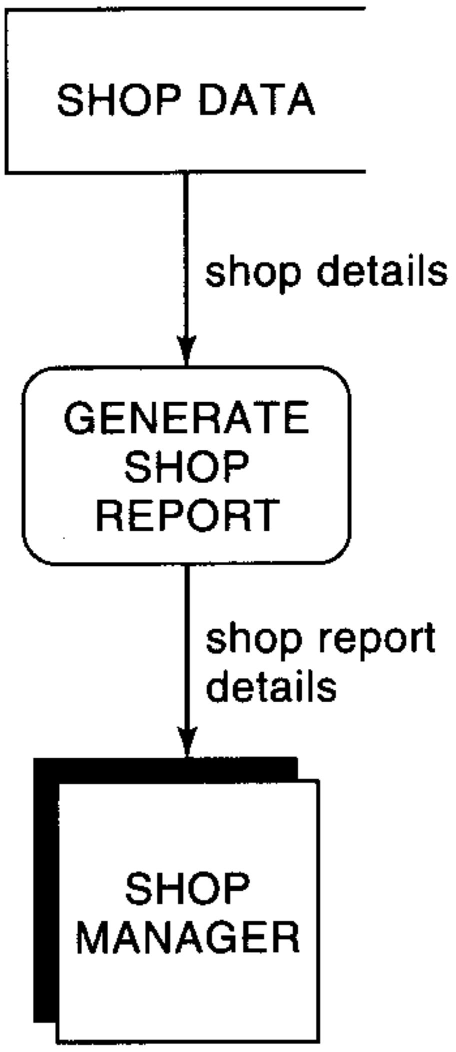

Figure 4.3 Data flow diagram for C–CC Corporation: first refinement

- In fact Figure 4.3 starts already part of *Step 1* of the SA technique *drawing the data flow diagram* (DFD).

Step 1. Draw the Data Flow Diagram (DFD)

- A DFD is built from 4 kinds of symbols, as shown in Figure 4.2.
  - An example of a *destination data* is the SHOP MANAGER, and of a source of data is an OPERATOR (who types in the data).
  - An example of a *flow of data*, represented by an arrow, are the flow transmitting the *shop report details* to the SHOP MANAGER.
  - An example of a *process transforming data* is the process GENERATE SHOP REPORT.
  - An example of a *store of data* is SHOP DATA (or REGION DATA).

- This explains the DFD explaining at a high level the data flow between SHOP DATA and SHOP MANAGER as in Figure 4.3. Similar illustrations are presented by Figures 4.4 and 4.5 below.

- Figure 4.4 illustrates the flow of data between the stores SHOP DATA and REGION DATA and the destinations SHOP MANAGER, REGION MANAGER and VP FOR SALES.

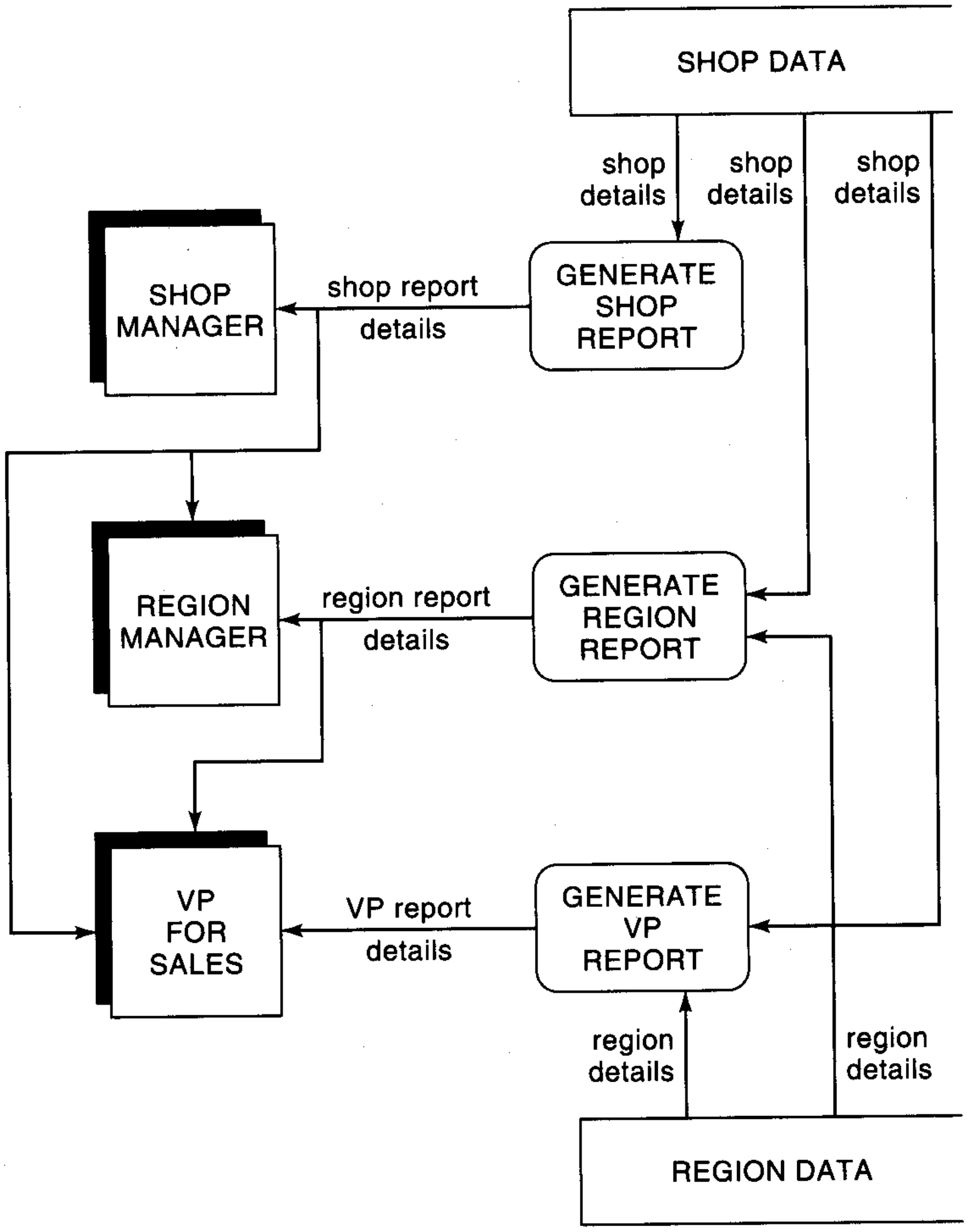

Figure 4.4 Data flow diagram for C–CC Corporation: second refinement

- Figure 4.5 illustrates also how the data reach the *sources destinations* SHOP DATA and REGION DATA.

- Figure 4.5 illustrates a way to visualize where the data in our case study come from:
  - *Target sales* are computed at the beginning of each year by management.
  - *Actual sales* are computed each month by the shop managers.

- However, actually entering these data is done by an OPERATOR, as *source* of data, as depicted.

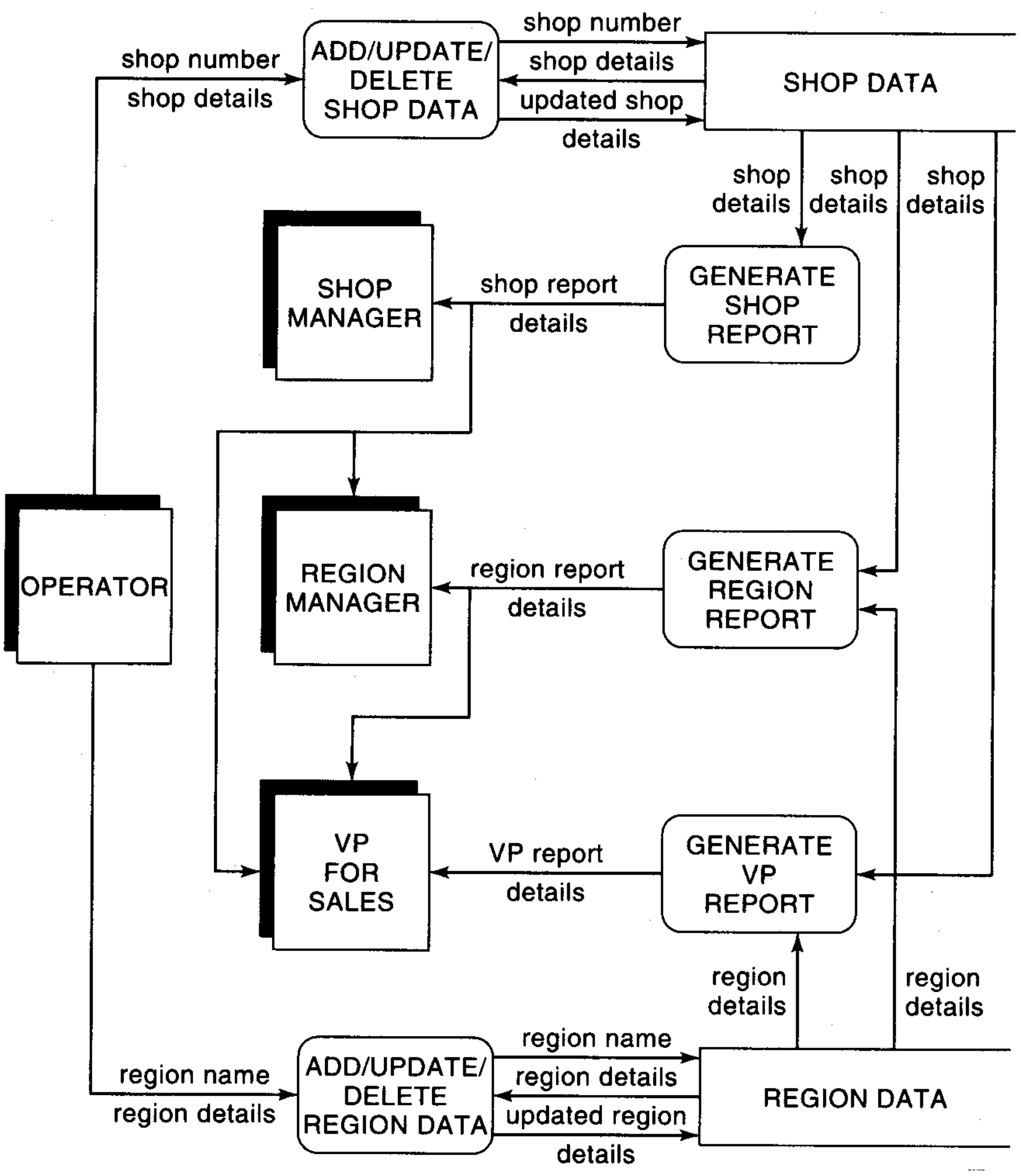

Figure 4.5 Data flow diagram for C–CC Corporation: third refinement

- Actually, with a view towards the future maintenance phase, the system analysts want to cater also for the possibility that shops and regions are *deleted* or *added to*. Consequently as additional operation on *shop details* (and upon *shop number* — consider Figures 6 and 7 on transparencies 116 and 118) the processes ADD SHOP DATA and DELETE SHOP DATA, and upon region details and region name (see Figures 7 and 8 on transparencies 118 and 119) the processes ADD REGION DATA and DELETE REGION DATA are introduced, apart from the operations (processes) UPDATE SHOP DATA and UPDATE REGION DATA.

- Subsequently, these figures are explained to, and discussed with, the client, Chesterton, in order to be completely sure that Figure 4.5 is indeed an accurate representation of the data flow of the proposed procedure. He approves of them.

A DFD is no Commitment to Implementation by a Computer

- Consider the following two implementations of the DFD in Figure 4.5.

  1. A **totally manually operated system**, described below:
     - *data store* SHOP DATA is contained in a *filing cabinet* with one *manilla folder* for each shop.
     - *adding a shop* consists of *inserting* a *new* folder into the filing cabinet, containing the *new* data,
     - *shop deletion* consists of *removing* a folder from the filing cabinet,
     - *updating shop details* consists of *changing* the data in the relevant folder,
     - *REGION DATA* are stored in a *binder*,

  - the OPERATOR performs the *ADD/DELETE/UPDATE REGION* operations similarly, and
  - processes GENERATE SHOP REPORT, GENERATE REGION REPORT, and GENERATE VP REPORT are *performed by a typist* using the informations within the folders.
2. A **fully automated product**, where:
  - SHOP DATA and REGION DATA reside in a data base on a hard disk,
  - the OPERATOR sits at the keyboard of a personal computer,
  - the data are extracted by interactive procedures and written to a file, and
  - a second program, run overnight, uses these files to update the data base containing SHOP DATA and REGION DATA, and prints the reports.

Conclusion:

STRUCTURED SYSTEMS ANALYSIS allows developers to postpone decisions regarding computerization until they know exactly what the product is to do, i.e. until after they have drawn the DFD.

Step 2. Decide what Sections to Computerize

- The team estimates the *costs* and *benefits* of *computerizations.* If benefits exceed the costs, they recommend to develop the product (i.e. they perform a *risk analysis*, as in the spiral model).

- The costs of computerizing the C–CC system are quickly estimated:
  - on the basis of similar products made by SRS the estimate of the costs of the system is \$29.000,
  - costs of the computer operator entering the relevant data is \$10/month, of a supervisor checking the output before distribution is \$20/month,
  - so the total costs over 5 years add up to \$30.800.

- Calculating the benefits of computerization is more complicated:
  - The costs of operating the manual system and checking the reports are estimated on \$450/month, i.e. \$27.000 over 5 years.
  - C–CC management realizes that a report which prints out that July's target has not been met has a greater effect upon improving August's sales, than when it is available by the end of August. Consequently the losses occuring as a result of *slow feed back* are estimated on: \$6000/year, i.e. \$30.000 over 5 years.
  - *The C–CC management still believing in the (unjustified pink) view that computers and its programs are infallible*, also wants to estimate losses/year as a result of mistakes in the manual system (a belief which is totally fallacious, as we have seen): \$12.000 over 5 years, being the costs of hiring a new shop manager, who is needed because a previous shop manager retires/gets fired as a result of wrong reports.

  - Hence the total costs of the manual system/5 years adds up to: $69.000.

- *Consequently it is less expensive to have a fully automated system* (even disregarding the costs of errors).

- Next the next 3 steps of the SSA consist of *refining* the data flows (arrows), processes (rounded rectangles), and data stores (open rectangles).

Step 3. Specify the Details of Data Flows

- All data flows in Figure 4.5 must now be specified.
- To this end the prototype is consulted, specifically the format of the reports output by the prototype (see transparencies 118, 119). This results in identification of the following data flows:
  - *shop number* (3 digits),
  - *region name* (2 letters),
  - *shop details*: see Figure 4.6 below for characterization as a *record* with *fields*:

```
shop details
    shop number (3 digits)
    shop address (25 characters)
    shop city (25characters)
    region name (2 letters)
    shop ZIP (9 digits)
    shop phone (10 digits)
    shop manager name (25 characters)
    actual sales (12x4 digits)
    target sales (12x4 digits)
```

Figure 4.6 Description of shop details

– *shop report details*: see Figure 4.7 below:

```
shop report details
 shop details [see Figure 4.6]
 shop effectiveness details
  monthly target sales (12 x 4 digits)
  monthly actual sales through report month
  (12 x 4 digits)
  monthly effectiveness through report month
  (12 x 3 digits)
  monthly shortfall through report month
  (12 x 3 digits)
  monthly objective achieved through report
  month (12 x 3 characters)
  end of report message (80 characters)
```

Figure 4.7 Description of shop report details

– similarly *region details*, *region report details* and *VP report details* are defined in Figures 4.8 and 4.9 below,

region details
  region name (2 letters)
  region manager name (25 characters)
  region manager address (25 characters)
  region manager city (25 characters)
  region manager ZIP (9 digits)
  region manager phone (10 digits)

```
region report details
  region details [see above]
  shop effectiveness summary line
    shop number (3 digits)
    shop address (25 characters)
    shop city (25 characters)
    region name (2 letters)
    shop ZIP (9 digits)
    target sales for report month (4 digits)
    actual sales for report month (4 digits)
    effectiveness for report month (3 digits)
    objective achieved for report month (3 characters)
  region effectiveness details
    number of shops in region (3 digits)
    number of shops achieving objective for report month
    (3 digits)
    number of shops not achieving objective for report
    month (3 digits)
    number of shops with not data for report month
    (3 digits)
```

Figure 4.8 Description of region details and region report details

```
VP report details
 region effectiveness summary line
  region name (2 letters)
  region effectiveness details [see Figure 4.8]
 C–CC effectiveness summary line
  total number of shops (3 digits)
  total number of shops achieving objective for
  report month (3 digits)
  total number of shops not achieving objective
  for report month(3 digits)
  total number of shops with no data for report
  month (3 digits)
```

Figure 4.9 Description of VP report details

Step 4. Define the Logic of the Processes

- Now that the data elements within the product have been determined, the next step is to define exactly what happens inside each process.
- The best way to to this is to use pseudocode; that is to say, *a notation that is similar to a programming language so that the processes are unbiguously defined.*
- Next follows pseudocode definition of GENERATE SHOP REPORT, GENERATE REGION REPORT and GENERATE VP REPORT

```
GENERATE SHOP REPORT
  For each shop;
   Fetch shop details from SHOP DATA;
   For each month through report month;
    Set effectiveness to (actual sales / target sales) x 100%,
     rounded to nearest integer.
    If effectiveness is $< 100\%$,
     set shortfall to (100% - effectiveness),
    else
     set shortfall to 0%
    If this month's shortfall is $\leq 5\%$, or if this month's
     shortfall is $\leq$ half of last month's shortfall
     this month's objective has been achieved,
    else
     this month's objective has not been achieved.
  Format and output shop report.
```

Figure 4.10 Pseudocode definition of process GENERATE SHOP REPORT

```
GENERATE REGION REPORT
  For each region;
    Fetch region details from REGION DATA
    Fetch shop details of all shops in that region
    from SHOP DATA
    Generate effectiveness data for that region.
    Format and output region report

GENERATE VP REPORT
  For each region;
    Fetch region details from REGION DATA
    Fetch details of all shops in that region from
    SHOP DATA
    Generate effectiveness data for for that region
    Generate effectiveness data for corporation
    as a whole
  Format and output VP report
```

Figure 4.11 Pseudocode definitions of process GENERATE REGION REPORT and process GENERATE VP REPORT

– The process ADD/UPDATE/DELETE SHOP DATA is split in three processes: ADD SHOP, UPDATE SHOP, DELETE SHOP, as depicted in Figure 4.12 below:

```
ADD SHOP
  Read information regarding new shop
  Update SHOP DATA

UPDATE SHOP
  Read shop number
  Retrieve shop details from SHOP DATA
  Make changes to specific fields
  Update SHOP DATA

DELETE SHOP
  Read shop number
  Delete shop record from SHOP DATA
```

Figure 4.12 Pseudocode definition of processes ADD SHOP, UPDATE SHOP, and DELETE SHOP

- Processes ADD REGION, UPDATE REGION and DELETE REGION are similarly defined, and omitted here.

Step 5. Define the Data Stores

- How are SHOP DATA and REGION DATA exactly defined? Checking the rapid prototype results in establishing that there are no other stores needed except these two, and that these two contain all the (component) information needed for defining the C–CC product.
  SHOP DATA serves only to store the data *shop details*; similarly REGION DATA serves only to store *region details*. Since *shop details* and *region details*have been defined before, see Figures 4.6 and 4.8, the following notation is used to express these stores:
  - SHOP DATA
    shop details (see Figure 4.6)
  - REGION DATA
    region details (see Figure 4.8)

- Further analysis is that SHOP DATA are identified by the following keys: *shop number* and *region name*. This is depicted in Figure 4.13 below:

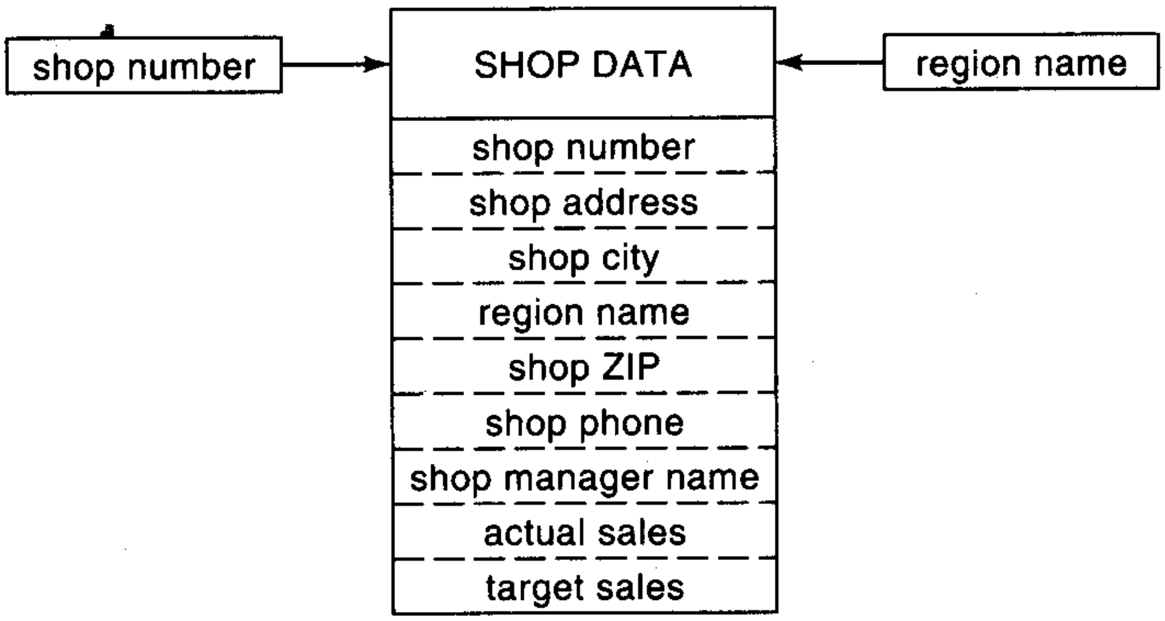

Figure 4.13 Data immediate access diagram (DIAD) for shop DATA

Step 6. Define the Physical Resources

- Now it is known where immediate acces is required and the representation (format) of every file record.

- Consequently, the files can be defined completely as to blocking factor, organization (sequential index, etc), and storage medium

  This is noted down by:

  – SHOP DATA

    Indexed sequential file
    Primary key : *shop number*
    Secondary key: *region number*

- Similarly one has:

  – REGION DATA

    Indexed sequential file
    Primary key : *region name*

- But then it is realized which (wrong) choice of programming language is promised to the client, C, which has no indexed sequential files. So one has to resort to sequential files (tapes) with no immediate access.

Step 7. Determine the Input/Output Specifications

- This consists of determining the *format of the input screens* and *that of the reports*.

- The developers are satisfied with describing the components to be seen on the input screen for the SHOP operations and data, see Figure 4.14, after consulting the rapid prototype:

ADD SHOP SCREEN
Operator must input
shop number
shop address
shop city
region name
shop ZIP
shop name
shop manager name

Product will then ask if user to input target sales. If so, operator must input target sales.

UPDATE SHOP SCREEN
operator must input
shop number

Product will then display
shop number
shop address
shop city
region name
shop ZIP
shop name
shop manager name
target sales for each month
actual sales for each month

Operator must has opportunity to change one or more fields

Figure 4.14 Description of input screens

- This has the following reason: lowering specification costs, because the client can only give the process a go after the (later) planning phase is finished and he knows the total price of the product.

  Once the client gives the contract to SRS the specification of the full screen can happen later. That way the price to be paid for requirements, specifications and planning phases is decreased, in case the client does not agree to accept SRS's offer.

- Similar definitions are set up for the input screens for the operations ADD REGION, UPDATE REGION, DELETE REGION.

Step 8. Perform the Sizing

- Sizing is the process of determining the data needed to decide exactly what hardware is required to run the product.

Step 9. Determine the Hardware Requirements

- Enough *MacIntoshes* and *laser printers* are underused at C–CC to make additional hardware unnecessary.

Testing During the Specification Phase

This has been discussed already on transparencies 51–54.

CASE TOOLS for Structured System Analysis

- A case tool for SSA consists of:
  1. a tool for drawing data flow diagrams (DFDs),
  2. a *data dictionary*, a tool for storing the name and representation (format) of every component of every data item in the product, including data flows and their components, data stores, their components and their internal variables.
  3. These two tools should be integrated, as is done in, e.g. Teamwork^{TM}, Excelerator^{TM}, Analyst/Designer^{TM} etc.
     Often these tools incorporate automatic consistency checkers ensuring consistency between the specification document and the design documents.