MODULE 6

PROCESS SPECIFICATION

Contents

PROCESS SPECIFICATION

MOTIVATION

Before designing a system an analyst must clearly understand the logic to be followed by each process block in a DFD. An analyst's understanding must be crosschecked with the user of the information system. A notation is thus needed to specify process block in detail, which can be understood by a user. Notation used must be appropriate for the type of the application to be modelled. Different notations are needed to represent repetition structures, complex decision situation and situations where sequencing of testing of conditions is important. For complex logical procedures a notation is needed which can also be used to detect logical errors in the specifications. This is called Decision Table. A tabular structure for representing logic can be used as a communication tool and can be automatically converted to a program.

LEARNING GOALS

At the end of this module you will know

1. How to use structured English to precisely specify processes
2. The terminology used in structured English
3. Terminology of decision tables and how it is used to specify complex logic
4. How to detect errors in decision table specifications
5. Terminology and use of decision trees
6. Comparison of structured English, decision tables and decision trees

LEARNING UNIT 1

Structured English specification

PROCESS SPECIFICATION

Once a DFD is obtained the next step is to precisely specify the process. Structured English, Decision tables and Decision Trees are used to describe processes. Decision tables are used when the process is logically complex involving large number of conditions and alternate solutions. Decision trees are used when conditions to be tested must follow a strict time sequence.

STRUCTURED ENGLISH

Structured English is similar to a programming language such as Pascal. It does not have strict syntax rules as in programming languages as the intention is only to give precise description of a process. The structured English description should be understandable to the user.

Example:

```
if customer pays advance
   then  Give 5% Discount
    else
       if purchase amount >=10,000
       then
            if the customer is a regular customer
             then    Give 5% Discount
               else    No Discount
            end if
        else   No Discount
      end if
 end if
```

DECISION TABLE-EXAMPLE

Same structured English procedure given as decision table

CONDITIONS	**RULE1**	**RULE2**	**RULE3**	**RULE4**
Advance payment made	**Y**	**N**	**N**	**N**
Purchase amt >=10,000	**-**	**Y**	**Y**	**N**
Regular Customer?	**-**	**Y**	**N**	**-**
ACTIONS				
Give 5% Discount	**X**	**X**	**-**	**-**
Give No Discount	**-**	**-**	**X**	**X**

DECISION TABLE-EXPLANATION

- Conditions are questions to be asked
- ‘Y’ is yes,’N’ is no & ‘-’ is irrelevant
- A ‘X’ against the action says the action must be taken
- A ‘-’ against the action says the action need not be taken

Rule 2 in decision table DISCOUNT states:
if no advance payment and purchase amount >=10000
and regular customer then give 5% discount

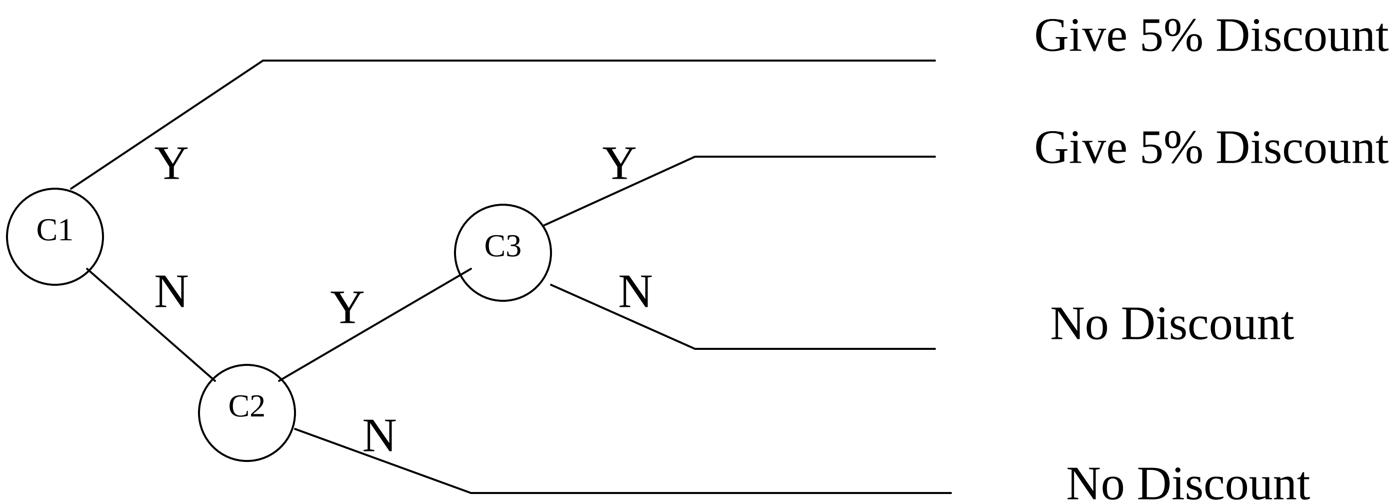
In Structured English, imperative sentences, actions to be performed should be precise and quantified
Good Example: Give discount of 20%
Bad Example: Give substantial discount

The operators and keywords in Structured English are as follows:

Operators -Arithmetic : +, -, /, *
Relational : >, >=, <, <=, =, !=
Logical : and, or, not
Keywords : if, then, else, repeat, until, while, do, case,

until, while, do, case, for, search, retrieve, read, write
Delimiters – {, }, end, end if, end for

The structured English procedure given above is expressed as a Decision tree below



C1: Advance payment made
C2: Purchase amount >=10,000
C3: Regular Customer

Y = Yes
N = No

STRUCTURED ENGLISH-DECISION STRUCTURES

<u>If</u> condition
    <u>then</u>
        { Group of statements }
    <u>else</u>
        { Group of statements }

end if

Example: if(balance in account >= min.balance)
then honor request
else reject request
end if

STRUCTURED ENGLISH-CASE STATEMENT

Case (variable)
Variable = P: { statements for alternative P}
Variable = Q: { statements for alternative Q}
Variable = R: { statements for alternative R}
None of the above: { statements for default case}
end case

Example : Case(product code)
product code =1 : discount= 5%
product code =2 : discount =7%
None of the above : discount=0
end case

STRUCTURED ENGLISH-REPETITION STRUCTURE

for index = initial to final do
{ statements in loop }
end for

Example : Total =0
for subject =1 to subject =5 do

```
        total marks=total marks +marks(subject)
        write roll no,total marks
      end for
```

STRUCTURED ENGLISH-WHILE LOOP

```
while condition do
      { statements in loop }
 end while
```

```
Example : while there are student records left  do
              read student record
                 compute total marks
                 find class
              write total marks, class, roll no
          end while
```

EXAMPLE

```
Update inventory file
for each item accepted record do
   { search inventory file using item code
     if successful
       then { update retrieved inventory record;
              write updated record in inventory file using accepted record}
       else { create new record in inventory file;
               enter accepted record in inventory file}
    end if
end for
```

LEARNING UNIT 2

Decision table based specifications

ADVANTAGES OF DECISION TABLE

Easy to understand by non-computer literate users and managers. Good documentation of rules used in data processing. Simple representation of complex decision rules. Tabular representation allows systematic validation of specification detection of redundancy, incompleteness & inconsistency of rules. There exist algorithms to automatically convert decision tables to equivalent computer programs.

METHOD OF OBTAINING DECISION TABLE FROM WORD STATEMENT OF RULES

EXAMPLE

A bank uses the following rules to classify new accounts
If depositor's age is 21 or above and if the deposit is Rs 100 or more, classify the account type as A If the depositor is under 21 and the deposit is Rs 100 or more, classify it as type B If the depositor is 21 or over and deposit is below Rs 100 classify it as C If the depositor is under 21 and deposit is below Rs 100 do-not open account

Identify Conditions: Age >= 21 Cl
Deposits >= Rs 100: C2

Identify Actions : Classify account as A, B or C
Do not open account

DECISION TABLE FROM WORD STATEMENT

Condition Stub

CODITIONS	Rule 1	Rule 2	Rule 3	Rule 4
C1 : Age >= 21	Y	N	Y	N
C2: Deposit >=100	Y	Y	N	N
ACTIONS				
A1: Classify as A	X	-	-	-
A2: Classify as B	-	X	-	-
A3: Classify as C	-	-	X	-
A4: Do not open Account	-	-	-	X

Action Stub

DECISION TABLE NOTATION EXPLAINED

CONDITION STUB	CONDITION ENTRIES
ACTION STUB	ACTION ENTRIES

• 4 Quadrants-demarcated by two double lines
• CONDITION STUB LISTS ALL CONDITIONS TO BE CHECKED
• ACTION STUB LISTS ALL ACTIONS TO BE CARRIED OUT
• LIMITED ENTRY DECISION TABLE:ENTRIES ARE Y or N or -.Y- YES,N- NO,-IRRELEVANT(DON'T CARE)
• X against action states it is to be carried out.
• - against action states it is to be ignored.
• Entries on a vertical column specifies a rule
•order of listing actions important while order of listing conditions is not important
•actions listed first carried out first
<u>sequential execution of actions</u>
•rules may be listed in any order

<u>INTERPRETING DECISION TABLE-ELSE RULE</u>

C1: Is applicant sponsored?	Y	Y	
C2: Does he have min Qualification?	Y	Y	ELSE
C3: Is fee paid?	Y	N	
A1: Admit letter	X	-	-
A2: Provisional Admit letter	-	X	-

-	-	X

<u>Interpretation</u>
R1: If applicant sponsored and he has minimum qualifications and his fee is paid –Send Admit letter
R2: If applicant sponsored and has minimum qualifications and his fee not paid send provisional admit letter
ELSE: In all other cases send regret letter.The else rule makes a decision table complete

<u>DECISION TABLE FOR SHIPPING RULES</u>

	R1	R2	R3	R4
C1: Qty ordered <= Quantity in stock?	Y	Y	N	N
C2: (Qty in stock-Qty ordered)<=reorder level	N	Y	-	-
C3: Is the partial shipment ok?	-	-	Y	N
A1:Qty shipped=Qty ordered	X	X	-	-
A2:Qty shipped=Qty in stock			X	

EXTENDED ENTRY DECISION TABLE

- Condition Entries not necessarily Y or N
- Action entries not necessarily X or - Extended Entry Decision Tables(EEDT) more concise
- EEDT can always be expanded to LEDT

Example

	R1	R2	R3	R4	R5	R6
C1 : Product code	1	1	1	1	1	2
C2 : Customer code	A	B	A	B	C	-
C3 : Order amount	<=500	<=500	>500	>500	-	-
Discount =	5%	7.5%	7.5%	10%	6%	5%

MIXED ENTRY DECISION TABLE

Can mix up Yes, No answers with codes

	Rl	R2	R3	R4	R5	R6
Cl : Product code = 1?	Y	Y	Y	Y	Y	N
C2: Customer code =	A	B	A	B	C	-
C3: Order amount < 500?	Y	Y	N	N	-	-
Discount =	5%	7.5%	7.5%	10%	6%	5%

Choice of LEDT, EEDT, MEDT depends on ease of communication with user. Softwares are available to translate DTs to programs.DT’s are easy to check.

LINKED DECISION TABLE

Decision table 1

Salary point=6	N	e
Conduct OK?	Y	l
Diligence OK?	Y	s
Efficiency OK?	Y	e
Go to table 2	X	-
No promotion	-	X

Decision table 2

Salary point>2	N	N	N	Y
1 yr as class 1 officer	Y	N	-	-
Departmental test Passed?	Y	-	N	-
Advance to next salary point	X	-	-	-
No promotion	-	X	X	-
Go to Table3	-	-	-	X

Decision table3

Complete departmental Course	Y	else
1 yr since last increment	Y	

1.Observe that one can branch between tables
2. Whenever complex rules are given it is a good idea to break them up into manageable parts

LOGICAL CORRECTNESS OF DECISION TABLE

Consider decision table DTI:

	Rl	R2
Cl: x>60	Y	-
C2:x<40	-	Y
Al	X	-
A2 :	-	X

We can expand decision table by replacing each –by Y & N

DT2:

	R11	R12	R21	R22
Cl: x>60	Y	Y	N	Y
C2:x<40	Y	N	Y	Y
Al	X	X	-	-
A2 :	-	-	X	X

A rule which has no – is an Elementary rule

DT2 is an Elementary Rule Decision Table (ERDT)

From this table we see that the rule YY has two contradictory actions. Thus we need to examine the table further and make sure it is not a serious mistake. Also the rule C1=C2=N is missing which needs further examination

LEARNING UNIT 3

Detecting- Incompleteness, Ambiguity,Contradictions & Redundancy in decision table specification

LOGICAL CORRECTNESS OF DECISION TABLE (CONTD)

A decision table with 1 condition should have 2 elementary rules, each elementary rule must be distinct, each elementary rule must have distinct action, if a decision table with k conditions does not have 2^k rules specified it is said to be incomplete.
For example : DT2 does not have the elementary rule C1:N, C2:N. It is thus incomplete.
If the decision table has the same elementary rule occurring more than once it is said to have multiplicity of specifications
For Example: In DT2 The rule C1:Y,C2:Y occurs twice. Thus it has multiplicity of specification.

If action specified for multiple identical rules are different then it is called ambiguous specifications
DT2 has an ambiguity. Rules R11 and R22 are identical but have different actions. Ambiguity may be apparent or real. It is said to be apparent if the rule leading to the ambiguity is logically impossible
For example,(x>60)=Y and (x<40)=Y cannot occur simultaneously.
Thus in DT2 rules R11 and R22 are apparently ambiguous rules
Apparently ambiguous rules is not an error

If an apparently ambiguous specification is real then it is a <u>contradiction</u>
For example : If C1:(X > 60) = Y and C2:(X > 40) = Y then
X = 70 will satisfy both inequalities.
As two actions are specified for (Cl = Y, C2 = Y) and they are different the rule is really ambiguous and is called <u>Contradictory Specification.</u>

If all 2^k elementary rules are not present in a k condition decision table is said to be <u>incomplete.</u>
DT2 is incomplete as rule C1:N, C2:N is missing
Rule C1=N, C2:=N is logically possible as C1=N is X<=60
and C2=N is X >= 40. A value of X = 50 will make C1=N,C2=N
Thus DT2 has a <u>real</u> incomplete specification
A decision table which has no real ambiguities or real <u>incompleteness</u> is said to be logically correct. Decision table with logical errors should be corrected

<u>USE OF KARNAUGH MAPS</u>

KARNAUGH map abbreviated K-map is a 2 dimensional diagram with one square per elementary rule

The k-map of DT2 is

C2 \ C1	N	Y
N	?	Al
Y	A2	A1,A2

If more than one action is in one square it is an ambiguous rule
If a square is empty it signifies incomplete specification.

<u>USE OF KARNAUGH MAPS</u>

<u>Structured English procedure:</u>

```
If carbon content<0.7
  then if Rockwell hardness>50
           then if tensile strength>30000
                    then steel is grade 10
                    else steel is grade 9
                  end if
            else steel is grade 8
           end if
   else steel is grade 7
 end if
```

DT3:

Decision table-Grading steel

C1:Carbon content <0.7	Y	Y	Y	N	Y	N	N	N
C2:Rockwell hardness>50	Y	Y	N	N	N	Y	Y	N
C3 tensile strength>30000	Y	N	N	N	Y	Y	N	Y
Grade	10	9	8	7	?	?	?	?

KARNAUGH MAPS – GRADING STEEL

C3 \ C1 C2	NN	NY	YY	YN
N	7	?	9	8
Y	?	?	10	?

Observe that the fact that the specification is incomplete is obvious in the Decision table whereas the structured English specification seems complete which is not.

DT4: DECISION TABLE-ARREARS MANAGEMENT

	R1	R2	R3	R4	R5	R6
C1:Payment in current month >min.specified payment	Y	N	N	-	-	-
C2:Payment in current month>0	-	Y	Y	-	N	N
C3:Any payment in last 3 months	-	-	-	N	Y	Y
C4: Actual arrears > 3(min. Specified payment per month)	-	Y	N	Y	N	Y
A1 : Send letter A	X	-	-	-	-	-
A2 : Send letter B	-	X	-	-	-	-
A3 : Send letter C	-	-	X	-	-	-
A4 : Send letter D	-	-	-	X	-	X
A5 : Send letter E	-	-	-	-	X	-

KARNAUGH MAP

C3C4 \ C1C2	NN	NY	YY	YN
NN	?	A3	A1	A1*
NY	A4	A2A4^{+}	A1A4^{+}	A1A4*
YY	A4	A2	A1	A1A4*
YN	A5	A3	A1	A1A5*

C1 : x>m C2:x>0 C3:y>0 C4:z>3m   m>0

C3,C4 independent of C1,C2   C1,C2 dependent

C1: Y C2: Y x>m, x>0 possible

C1: Y C2: N x>m, x<=0 not logically possible

C1: N C2: Y x<=m,x>0 possible

C1: N C2: N x<=m,x<=0 possible

Thus C1,C2,C3 C4:NNNN incomplete specification

BOXES MARKED * NOT LOGICALLY POSSIBLE

Rules C1 C2 C3 C 4 : NYNY and YYNY logical errors

Errors to be corrected after consulting users who formulated the rules

CORRECT DECISION TABLE

If users say that for rules C1C2C3C4:NYNY AND YYNY (marked with + in k-map) the action is A4 and for C1C2C3C4:NNNN also it is A4, the corrected map is

C3C4 \ C1C2	NN	NY	YY	YN
NN	A4	A3	A1	X
NY	A4	A4	A4	X
YY	A4	A2	A1	X
YN	A5	A3	A1	X

IMPOSSIBLE RULES

CORRECTED DECISION TABLE DT4

C1	Y	Y	Y	N	N	N	N	Y	N	N	N	N
C2	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N
C3	N	Y	Y	Y	N	Y	N	N	Y	N	N	Y
C4	N	Y	N	Y	N	N	Y	Y	Y	Y	N	N

Question: Can the number of rules be reduced?
Answer : Yes, by combining rules with the same action

Action A1 can be represented by the Boolean expression:
$C1C2\overline{C3}\,\overline{C4} + C1C2C3\overline{C4} + C1C2C3C4 = C1C2\overline{C3}\,\overline{C4} + C1C2C3\,(C4+\overline{C4})$
$=C1C2\overline{C3}\,\overline{C4}+C1C2C3 = C1C2\overline{C4} + C1C2C3$

LEARNING UNIT 4

Eliminating redundancy in specifications

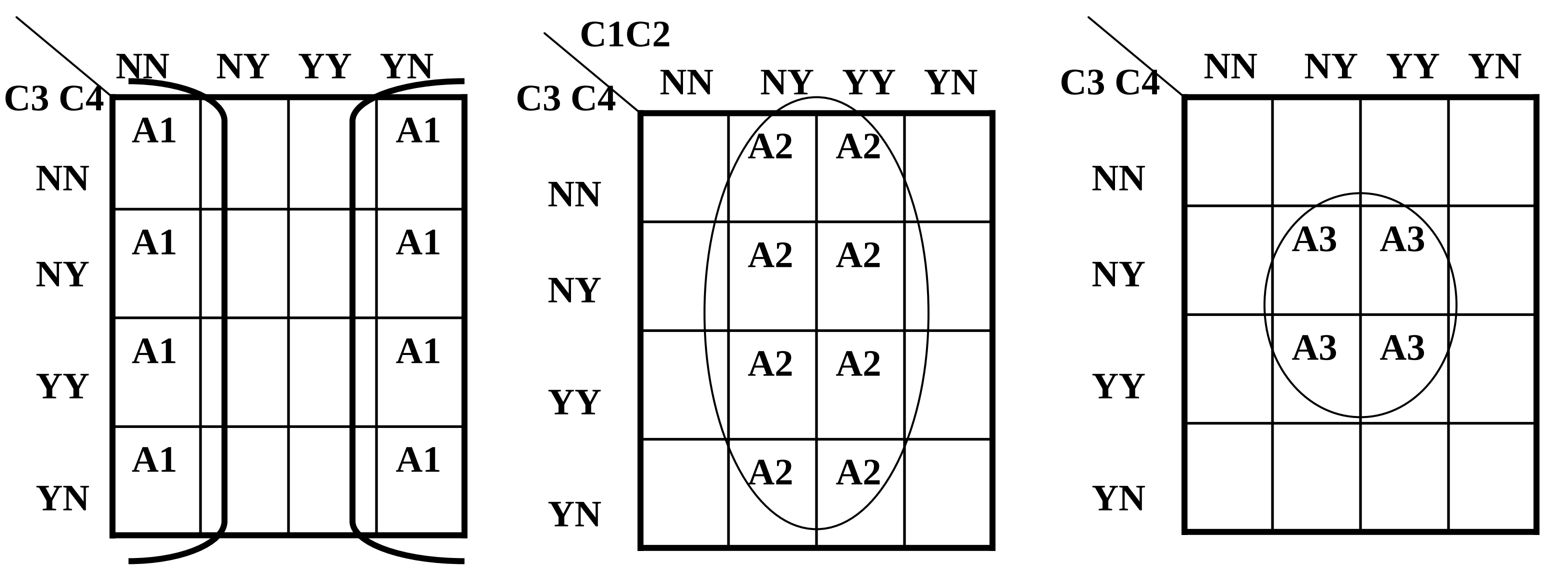
REDUNDANCY ELIMINATION

Redundancy can be eliminated by systematically applying four identities of Boolean Algebra

These identities are

$$A + \overline{A}=1$$

KARNAUGH MAP REDUCTION



$$A3 = C1C2\overline{C3}C4 + C1C2\overline{C3}C4 + \overline{C1}C2C3C4 + C1C2C3C4$$

$$= \overline{C2}\,\overline{C3}C4(\overline{C1}+C1) + C2C3C4(\overline{C1}+C1)$$

$$= C2C4(\overline{C3}+C3) = C2C4$$

REDUCING DECISION TABLES-USE OF K-MAP

C3C4 \ C1C2	NN	NY	YY	YN
NN	A4	A3	A1	X
NY	A4	A4	A4	X
YY	A4	A2	A1	X
YN	A5	A3	A1	X

Boxes marked X correspond to impossible rules.
They can be employed if they are useful in reducing rules

Using k-map reduction rules we get

A1 : $\overline{C1}C4+C1C3$
A2 : $\overline{C1}C2\overline{C3}C4$
A3 : $\overline{C1}C2\overline{C4}$
A4 : $\overline{C3}C4+\overline{C2}C3+C2C4$
A5 : C2C3C4

REDUCING DECISION TABLES

C1: Payment in current month > min specified payment	Y	Y	N	N	-	-	-	-
C2: Payment in current month>0	-	-	Y	Y	-	N	N	N
C3: Any payment in last 3 months	-	Y	Y	-	N	N	-	Y
C4: Actual arrears> 3(minimum specified payment per month)	N	-	Y	N	Y	-	Y	N
A: Send letter A	X	X	-	-	-	-	-	-
B: Send letter B	-	-	X	-	-	-	-	-
C: Send letter C	-	-	-	X	-	-	-	-
D: Send letter D	-	-	-	-	X	X	X	-
E: Send letter E	-	-	-	-	-	-	-	X

EXAMPLE-REDUCTION OF RULES IN WORD STATEMENT

Rules: Insure Driver if following rules are satisfied

1.Drivers annual income > 20000 & is married male
2.Drivers annual income > 20000 & is married and over 30
3.Drivers annual income <= 20000 & she is married female
4.Driver is male over 30
5.Driver is married and age is not relevant
Else do not insure

Conditions:

C1 : Annual income > 20000
C2 : Male
C3 : Married
C4: Age > 30

Action: Insure or do not insure

DECISION TABLE FOR INSURANCE RULES

C1 : Annual income> 20000	Y	Y	N	-	-	E
C2: Male	Y	-	N	Y	-	L
C3: Married	Y	Y	Y	-	Y	S
C4: Age > 30	-	Y	-	Y	N	E
A1:Insure	X	X	X	X	X	-
A2 :Do not insure	-	-	-	-	-	X

C3C \ C1C2	NN	NY	YY	YN
NN				
NY		A1	A1	
YY	A1	A1	A1	A1
YN	A1	A1	A1	A1

A1=C3+C2.C4

REDUCED DECISION TABLE

C2 : Male	-	Y	**ELSE**
C3 : Married	Y	-	
C4 : Age > 30	-	Y	
Al : Insure	X	X	-
A2 : Do not Insure	-	-	X

Reduced rules : Insure if married or male over 30

Observe 6 rules simplified to 2 and 1 condition removed

LEARNING UNIT 5

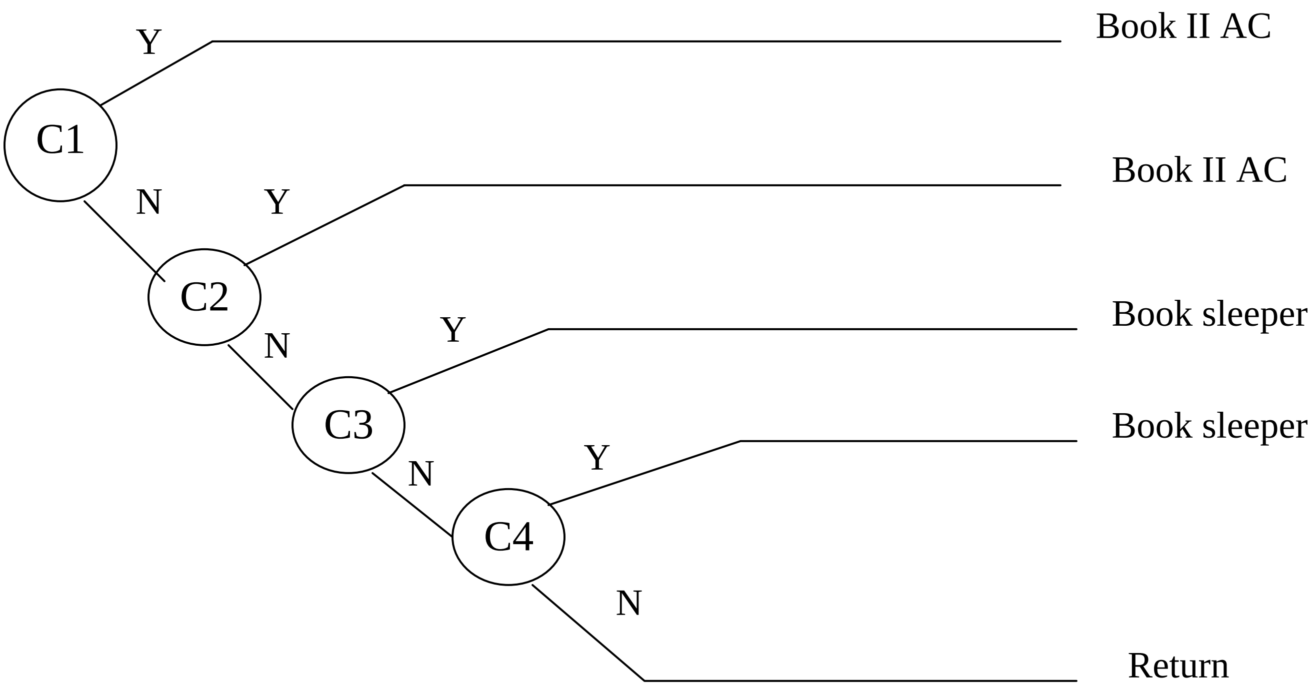
Decision trees for specifications

DECISION TREES

Decision Trees is used when sequence of testing condition is important. It is more procedural compared to Decision tables.

EXAMPLE – DECISION TREE TO BOOK TRAIN TICKET

Book by II AC on 4/8/04 if available else book by II AC on 5/8/04.If both not available book by sleeper on 4/8/04 if available else book on 5/8/04 by sleeper. If none available return.



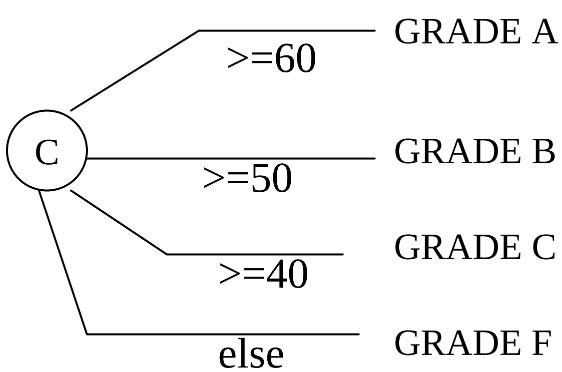
C1: Is II AC ticket available on 4/8/04
C2: Is II AC ticket available on 5/8/04
C3: Is sleeper available on 4/8/04
C4: Is sleeper available on 5/8/04

Observe in the tree sequencing of conditions which is important in this example

CONDITIONS

- Decision trees are drawn left to right
- Circles used for conditions
- Conditions labelled and annotation below tree
- Conditions need not be binary

For example:



▪Sometimes Decision trees are more appropriate to explain to a user how decisions are taken

COMPARISON OF STRUCTURED ENGLISH, DECISION TABLES AND DECISION TREES

CRITERION FOR COMPARISON	STRUCTURED ENGLISH	DECISION TABLES	DECISION TREES
ISOLATING CONDITIONS & ACTIONS	**NOT GOOD**	**BEST**	**GOOD**
SEQUENCING CONDITIONS BY PRIORITY	**GOOD**	**NOT GOOD**	**BEST**
CHECKING FOR COMPLETENESS, CONTRADICTION	**NOT GOOD**	**BEST**	**NOT GOOD**

WHEN TO USE STRUCTURED ENGLISH,DECISION TABLES AND DECISION TREES

Use Structured English if there are many loops and actions are complex

Use Decision tables when there are a large number of conditions to check and logic is complex

Use Decision trees when sequencing of conditions is important and if there are not many conditions to be tested

REFERENCES

1. V.Rajaraman, "Analysis and Design of Information Systems", 2nd Edition, Prentice Hall of India, New Delhi, 2002. Most of the material in this module is based on Chapter 8 and 9 of the above book. The book is perhaps the only one which has extensive discussion on error detection in Decision Tables.

2. K.E. Kendall and J.E.Kendall, "Systems Analysis and Design", 5th Edition, Pearson Education Asia, Delhi, 2003. Has a brief discussion of structured English, Decision Tables and Decision Trees (pages 353 to 369). Website www.prenhall.com/kendall has a lot of support material and case study for students.

3. J.A.Hoffer, J.F.George, J.S.Velacich, "Modern Systems Analysis and Design", Third Edition, Pearson Education Asia, 2002. Chapter 7 (pages 282 to 303) cover the topics in this module. The book has a number of interesting case studies and a good problem set. The web site http://prenhall.com/hoffer has material to assist students who use this text book.

4. E.Yourdon “Modern Structured Analysis”, Prentice Hall of India, 1996. Chapter 11 (pages 203 to 232) describes structured English and Decision Tables. There is a larger set of exercises at the end of the chapter.