A Practical Example of a Decision Table

Company X sells merchandise to wholesale and retail outlets. Wholesale customers receive a two percent discount on all orders. The company also encourages both wholesale and retail customers to pay cash on delivery by offering a two percent discount for this method of payment. Another two percent discount is given on orders of 50 or more units. Each column represents a certain type of order.

Less than 50 units ordered	Y	Y	Y	Y	N	N	N	N
Cash on Delivery	Y	Y	N	N	Y	Y	N	N
Wholesale Outlet	Y	N	Y	N	Y	N	Y	N
Discount Rate 0%				X				
Discount Rate 2%		X	X					X
Discount Rate 4%	X					X	X	
Discount Rate 6%					X			

The Decision Table records the conditions for discounts in the top left quadrant along with the ranges for the conditions in the top right quadrant. The bottom half of the table lists the actions taken, i.e., the discount rates that apply, based on the conditions. Each column represents a certain type of order. For example, column two represents cash on delivery orders of less than 50 units from retailers.

The managers realize that certain loyal customers order from every catalog and that some people on the mailing list never order. These ordering patterns are easy to observe, but deciding which catalogs to send customers who order only from selected catalogs is more difficult. Once these decisions are made, a decision table is constructed for three conditions (C1: customer ordered from Fall catalog; C2: customer ordered from Christmas catalog; and C3: customer ordered from specialty catalog), each having two alternatives (Y or N). Three actions can be taken (A1: send out this year's Christmas catalog; A2: send out the new specialty catalog; and A3: send out both catalogs). The resulting decision table has six rows (three conditions and three actions) and eight columns (two alternatives two alternatives two alternatives).

Conditions and Actions	1	2	3	4	5	6	7	8
Customer ordered from Fall catalog.	Y	Y	Y	Y	N	N	N	N
Customer ordered from Christmas catalog.	Y	Y	N	N	Y	Y	N	N
Customer ordered from specialty catalog.	Y	N	Y	N	Y	N	Y	N
Send out this year's Christmas catalog.		X		X		X		X
Send out specialty catalog.			X				X	
Send out both catalogs.	X				X			

The decision table is now examined to see if it can be reduced.

Q)-A householder is currently considering insuring the contents of his house against theft for one year. He estimates that the contents of his house would cost him £20,000 to replace.

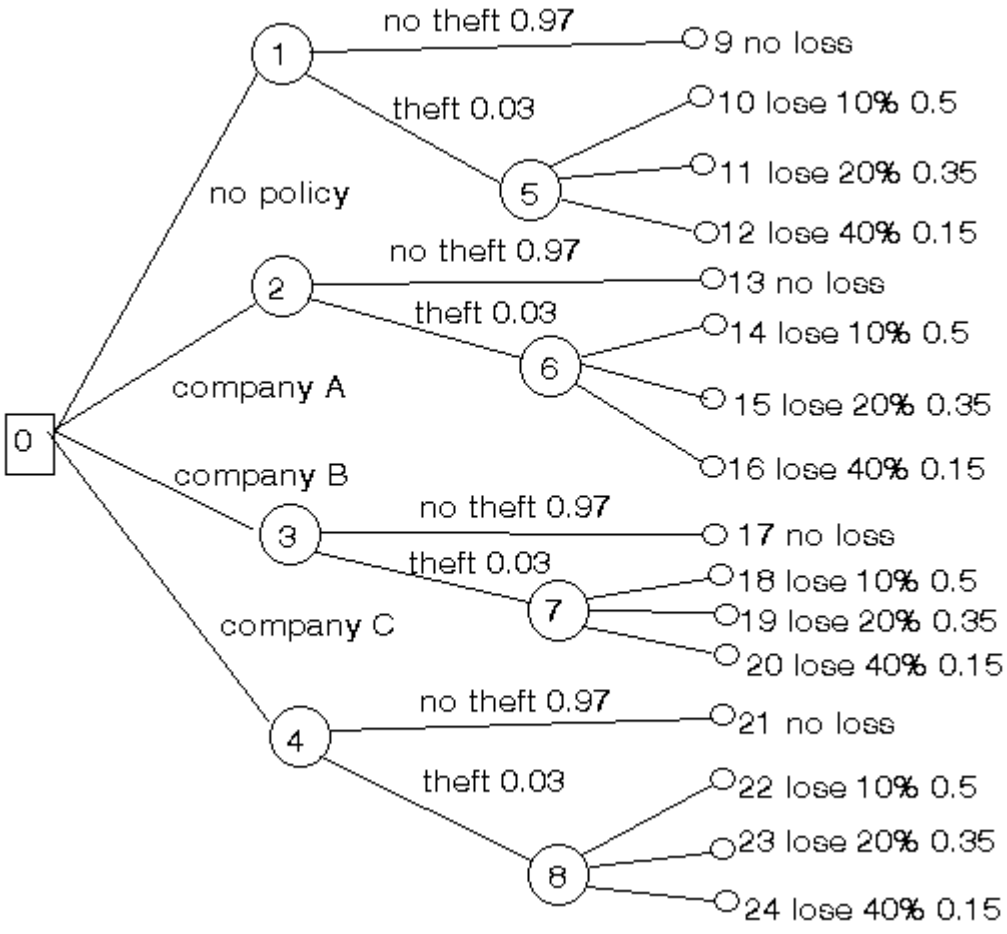
Local crime statistics indicate that there is a probability of 0.03 that his house will be broken into in the coming year. In that event his losses would be 10%, 20%, or 40% of the contents with probabilities 0.5, 0.35 and 0.15 respectively.

An insurance policy from company A costs £150 a year but guarantees to replace any losses due to theft.

An insurance policy from company B is cheaper at £100 a year but the householder has to pay the first £x of any loss himself. An insurance policy from company C is even cheaper at £75 a year but only replaces a fraction (y%) of any loss suffered.

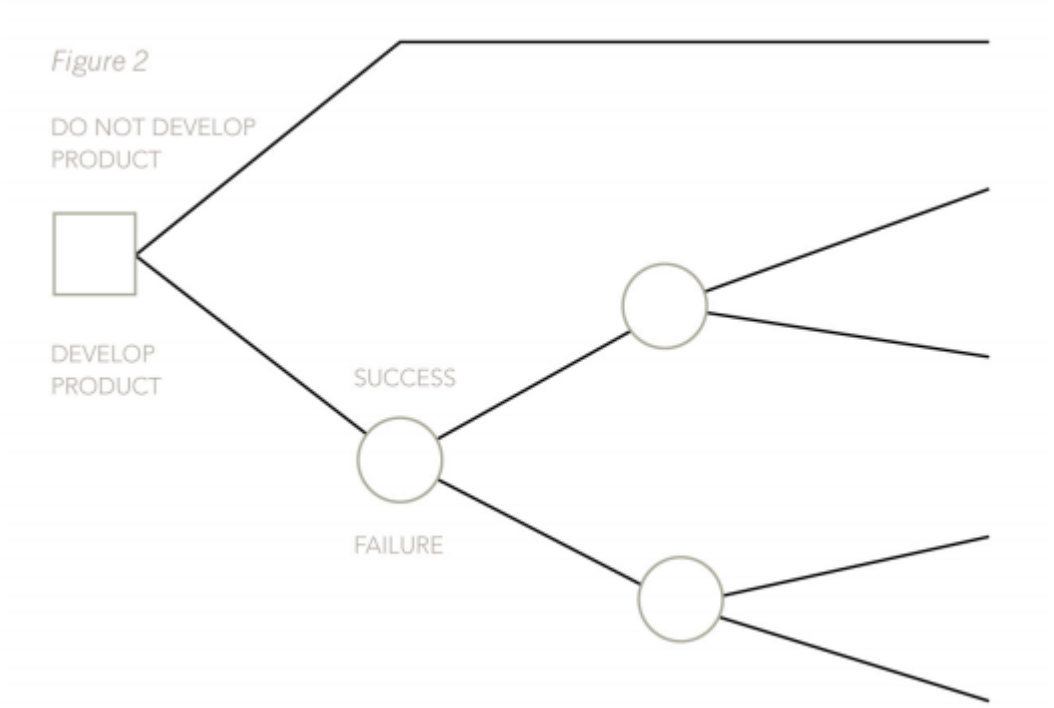
Assume that there can be at most one theft a year.

- Draw the decision tree.
- **Solution**
- The decision tree for the problem is shown below.

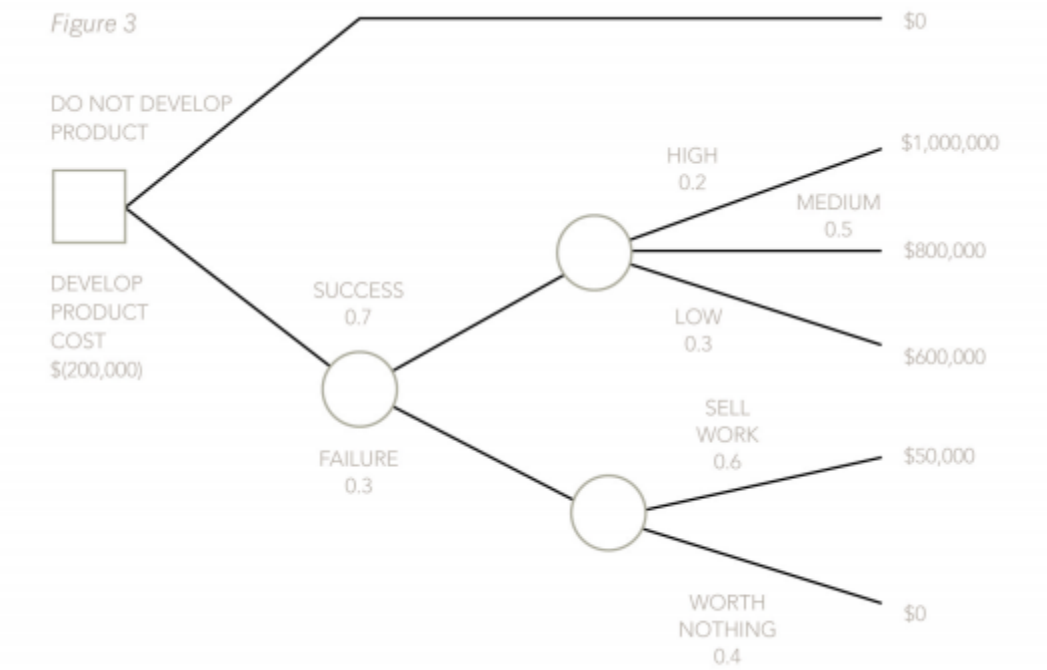


Example :A company is deciding whether to develop and launch a new product. Research and development costs are expected to be $400,000 and there is a 70% chance that the product launch will be successful, and a 30% chance that it will fail. If it is successful, the levels of expected profits and the probability of each occurring have been estimated as follows, depending on whether the product's popularity is high, medium or low: Probability Profits High: 0.2 $500,000 per annum for two years Medium: 0.5 $400,000 per annum for two years Low: 0.3 $300,000 per annum for two years If it is a failure, there is a 0.6 probability that the research and development work can be sold for $50,000 and a 0.4 probability that it will be worth nothing at all.

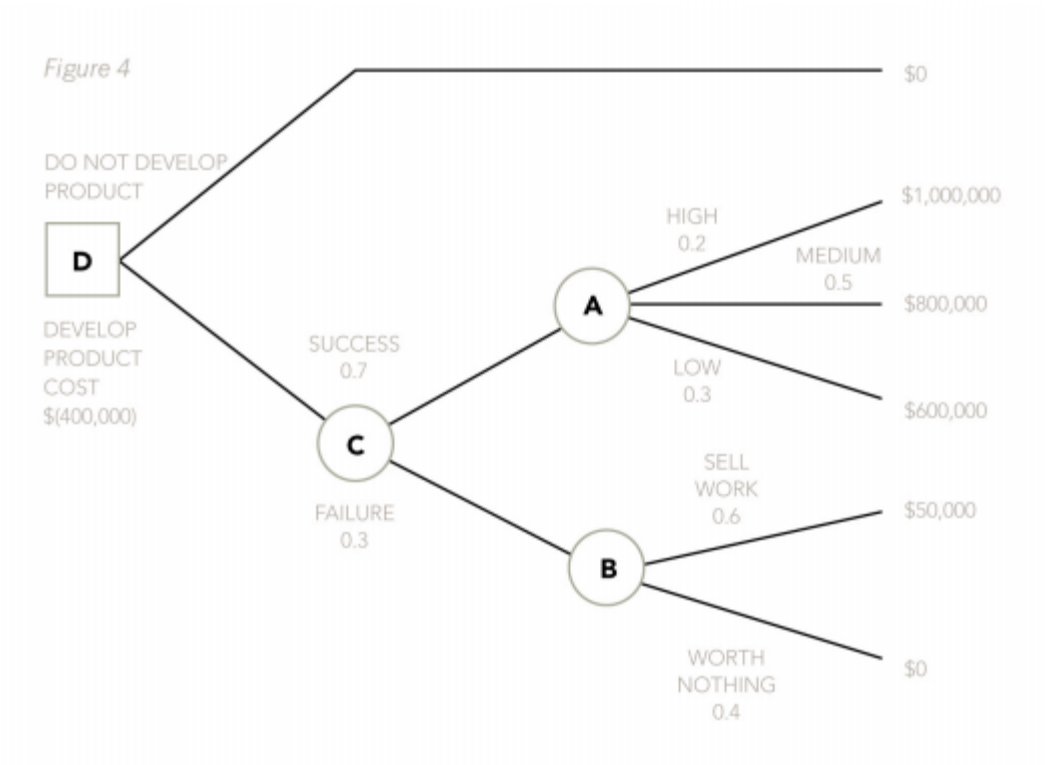
The basic structure of the decision tree must be drawn, as shown below:



Next, the probabilities and the profit figures must be put on, not forgetting that the profits from a successful launch last for two years, so they must be doubled.



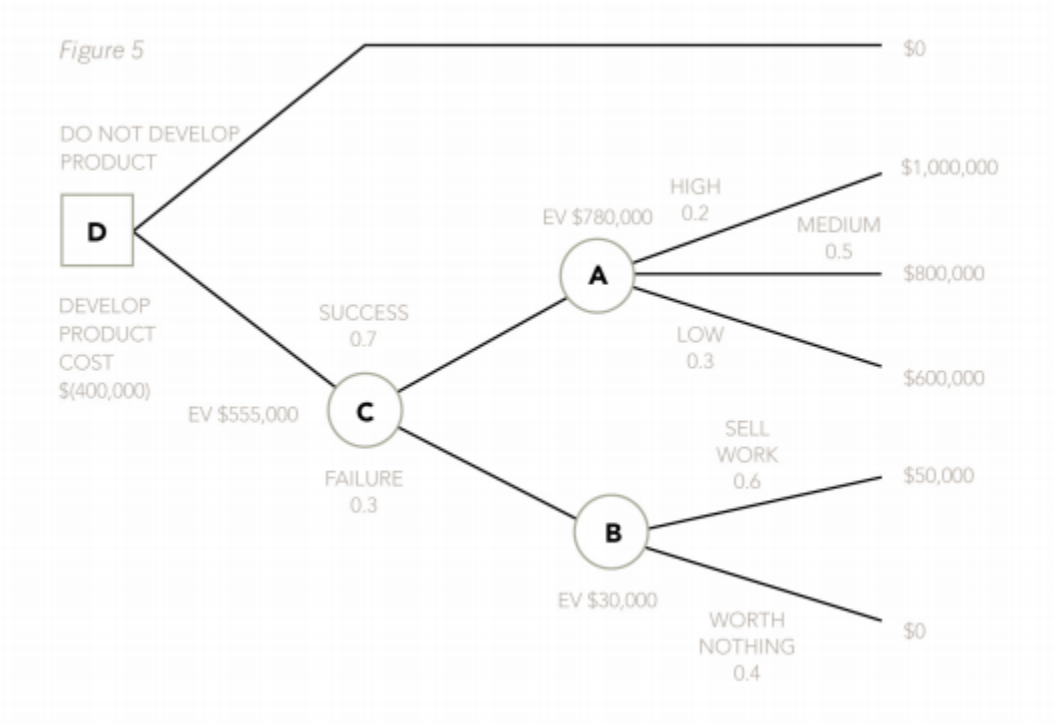
Now, the decision points and outcome points must be labelled, starting from the right-hand side and moving across the page to the left.



Now, calculate the expected values at each of the outcome points, by applying the probabilities to the profit figures. An expected value will be calculated for outcome point A and another one will be calculated for outcome point B. Once these have been calculated, a third expected value will need to be

calculated at outcome point C. This will be done by applying the probabilities for the two branches off C to the two expected values that have already been calculated for A and B. EV at A = (0.2 x $1,000,000) + (0.5 x $800,000) + (0.3 x $600,000) = $780,000. EV at B = (0.6 x $50,000) + (0.4 x $0) = $30,000. EV at C = (0.7 x $780,000) + (0.3 x $30,000) = $555,000.

These expected values can then be put on the tree if there is enough room.



Once this has been done, the decision maker can then move left again to decision point D. At D, the decision maker compares the value of the top branch of the decision tree (which, given there were no outcome points, had a certain outcome and therefore needs no probabilities to be applied to it) to the expected value of the bottom branch. Costs will then need to be deducted. So, at decision point D compare the EV of not developing the product, which is $0, with the EV of developing the product once the costs of $400,000 have been taken off – ie $155,000. Finally, the recommendation can be made to management. Develop the product because the expected value of the profits is $155,000.

Example scenario: "A marketing company wishes to construct a decision table to decide how to treat clients according to three characteristics: Gender, City Dweller, and age group: A (under 30), B (between 30 and 60), C (over 60). The company has four products (W, X, Y and Z) to test market. Product W will appeal to female city dwellers. Product X will appeal to young females. Product Y will appeal to Male middle aged shoppers who do not live in cities. Product Z will appeal to all but older females."

The process used to create a decision table is the following:

1. Identify conditions and their alternative values.

- There are 3 conditions: gender, city dweller, and age group. Put these into table as 3 rows in upper left side.
- Gender's alternative values are: F and M.
- City dweller's alternative values are: Y and N
- Age group's alternative values are: A, B, and C

2. Compute max. number of rules.
   - Determine the product of number of alternative values for each condition.
   - 2 x 2 x 3 = 12.
   - Fill table on upper right side with one column for each unique combination of these alternative values. Label each column using increasing numbers 1-12 corresponding to the 12 rules. For example, the first column (rule 1) corresponds to F, Y, and A. Rule 2 corresponds to M, Y, and A. Rule 3 corresponds to F, N, and A. Rule 4 corresponds to M, N, and A. Rule 5 corresponds to F, Y, and B. Rule 6 corresponds to M, Y, and B and so on.
3. Identify possible actions
   - Market product W, X, Y, or Z. Put these into table as 4 rows in lower left side.
4. Define each of the actions to take given each rule.
   - For example, for rule 1 where it is F, Y, and A; we see from the above example scenario that products W, X, and Z will appeal. Therefore, we put an 'X' into the table's intersection of column 1 and the rows that correspond to the actions: market product W, market product X, and market product Z.

1	2	3	4	5	6	7	8	9	10	11	12	
Gender	F	M	F	M	F	M	F	M	F	M	F	M
City	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Age	A	A	A	A	B	B	B	B	C	C	C	C
MarketW	X				X				X			
MarketX	X		X									
MarketY								X				
MarketZ	X	X	X	X	X	X	X	X		X		X

5. Verify that the actions given to each rule are correct.
6. Simplify the table.
   - Determine if there are rules (columns) that represent impossible situations. If so, remove those columns. There are no impossible situations in this example.
   - Determine if there are rules (columns) that have the same actions. If so, determine if these are rules that are identical except for one condition and for that one condition, all possible values of this condition are present in the rules in these columns. In the example scenario, columns 2, 4, 6, 7, 10, and 12 have the same action. Of these columns: 2, 6, and 10 are identical except for one condition: age group. The gender is M and they are city dwellers. The age group is A for rule 2, B for rule 6, and C for rule 10. Therefore, all possible values of condition 'age group' are present. For rules 2, 6, and 10; the age group is a "don't care". These 3 columns can be collapsed into one column and a hyphen is put into the age group

location to signify that we don't care what the value of the age group is, we will treat all male city dwellers the same: market product Z.

	1	2	3	4	5	6	7	8	9	10
Gender	F	M	F	M	F	M	F	M	F	M
City	Y	Y	N	N	Y	N	N	Y	N	N
Age	A		A	A	B	B	B	C	C	C
MarketW	X				X			X		
MarketX	X		X							
MarketY							X			
MarketZ	X	X	X	X	X	X	X			X

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.

Structured English

Structured English uses plain English words in structured programming paradigm. It is not the ultimate code but a kind of description what is required to code and how to code it. The following are some tokens of structured programming.

```
IF-THEN-ELSE,
DO-WHILE-UNTIL
```

Analyst uses the same variable and data name, which are stored in Data Dictionary, making it much simpler to write and understand the code.

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,
for, search, retrieve, read, write
Delimiters – {, }, end, end if, end for.

STRUCTURED ENGLISH-DECISION STRUCTURES

If condition
then
{ Group of statements }
else
{ Group of statements }
end if

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

Example:

```
BEGIN IF
        IF Customer Age > 65
        THEN Billing Rate = Senior Citizen Rate
        ELSE Billing Rate = Standard Rate
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

Example:

```
SELECT CASE CASE 1 (State = "AZ")
Sales Tax = 0.075
CASE 2 (State = "CA")
Sales Tax = 0.05
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:

DO
ACCEPT Inventory-record-id
READ Inventory-record for Inventory-record-id
PRINT Quantity-in-stock and Inventory-item-name
UNTIL End-of-file

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

Structured English Tips:

Capitalize keywords such as GET, IF, THEN, ELSE, END IF, CASE, DO WHILE, etc.

- Express *actions* as concise, simple commands:
  - RECEIVE Paycheck Request from Employee
  - GET the employee's Employee Record from Employee File
  - SEND the employee's Paycheck to Employee
  - STORE the employee's updated Employee Record to Employee File
  - APPEND the employee's ID number and salary to Manager Salary List

- Express CASE or IF *conditions* as simple logical expressions that anyone can understand (e.g., use of "=", "<", etc is fine)
  - IF Age <= 21 THEN …
  - CASE 1 (Gender = Male)
  - CASE 1 (Weight < 100 pounds)

- For situations in which the IF … THEN logic will include a series of conditions, use ELSE and ELSEIF keywords (see example below):

```
BEGIN IF
    IF Customer Age > 65
        THEN Billing Rate = Senior Citizen Rate
    ELSEIF Customer Age < 12
        THEN Billing Rate = Child Rate
   ELSE Billing Rate = Standard Rate
END-IF
```

- If process steps will be repeated, then use a repetition loop (e.g., DO WHILE … END DO WHILE):

```
DO WHILE there are Orders to process
      ….. [Insert statements here to describe processing] …..
END DO WHILE
```

- Try to avoid complex statements, if possible (see example below):
  - The following conditional statement is readable, but some people may need to think twice to interpret the portion of the statement that is to the right of the "<=":

    IF Start Date <= Todays Date – 30 THEN …

  - To simplify this statement, define a new term ("Days Lapsed") near top of the structured English and then use the term later to simplify the conditional statement:

    Days Lapsed = Todays Date – Start Date
    ……
    IF Days Lapsed >= 30 THEN …

- When appropriate, use indentation to enhance readability!
  - Examples of appropriate applications of indentation: Indent blocks of logic associated with IF statements, CASE statements, DO WHILE statements

- File Names
  - Separate words with hyphens
  - Use Title Case, e.g. Invoice-Record
- Common verbs

- READ, ACCEPT, GET, WRITE, PRINT, SORT, MOVE, MERGE, ADD, SUBTRACT, MULTIPLY, DIVIDE
- Common nouns:
  - Variable names, attributes, data flow inputs and outputs
- Focus on how the process converts the inputs to outputs

It uses there basic types of statements to describe a process –

i. Sequence structure –

It is a single step or action included in a process. It does not depend on the existence of any condition.

For Ex:

To buy a computer science book we follow the steps –

a. Pick out a desirable book.
b. Take it to the sells counter.
c. Pay cash for the book.
d. Collect cash receipt.
e. Collect the book and leave the store.

It is used when two or more actions can be taken depending on the value of a specific condition. We can describe it by using following example –

If

Book found on the store

Then

Take the book to sales counter
Pay cash
Take receipt & book
Leave store

Else

Do not take book
Leave store.

ii. Iterative structure –

In various operating condition it is common to find that certain activities are repeatedly executes while certain conditions exist. It permits analysts to describe these conditions.

Until the category of student is SC

Do give the discount to students on fee 50%
Take next student

End Until

Example

We take the same example of Customer Authentication in the online shopping environment. This procedure to authenticate customer can be written in Structured English as:

```
Enter Customer_Name
SEEK Customer_Name in Customer_Name_DB file
IF Customer_Name found THEN
   Call procedure USER_PASSWORD_AUTHENTICATE()
ELSE
   PRINT error message
   Call procedure NEW_CUSTOMER_REQUEST()
ENDIF
```

The code written in Structured English is more like day-to-day spoken English. It can not be implemented directly as a code of software. Structured English is independent of programming language.

Example

We process all our claims in this manner. First, we determine whether the claimant has ever sent in a claim before; if not, we set up a new record. The claim totals for the year are then updated. Next, we determine if a claimant has policy A or policy B, which differ in deductibles and copayments (the percentage of the claim claimants pay themselves). For both policies, we check to see if the deductible has been met ($100 for policy A and $50 for policy B). If the deductible has not been met, we apply the claim to the deductible. Another step adjusts for the copayment; we subtract the percentage the claimant pays (40 percent for policy A and 60 percent for policy B) from the claim. Then we issue a cheque if there is money coming to the claimant, print a summary of the transaction, and update our accounts. We do this until all claims for that day are processed.

```
DO WHILE there are claims remaining
        IF claimant has not sent in a claim
                THEN set up new claimant record
        ELSE continue
Add claim to YTD Claim
        IF claimant has policy–plan A
                THEN IF deductible of $100.00 has not been met
                        THEN subtract deductible–not–met from claim
                        Update deductible
                ELSE continue
                ENDIF
                Subtract copayment of 40% of claim from claim
        ELSE IF claimant has policy–plan B.
```

```
                    THEN IF deductible of $50.00 has not been met
                            THEN subtract deductible–not–met from claim
                            Update deductible
                    ELSE continue
                    ENDIF
                    Subtract copayment of 60% of claim from claim
            ELSE continue
ELSE write plan–error–message
            ENDIF
ENDIF
IF claim is greater than zero
            THEN print check
ENDIF
Print summary for claimant
Update accounts
ENDDO
```

ILLUSTRATION 1:

Let us assume the following discount policy:-

Bookstores get a trade discount of 35% for order from libraries and individuals; 15% allowed on orders of 6-19 copies per book title; 20% on orders for 20-49 copies per book title; 25% on orders for 50 copies or more per book title. Draw a decision table.

SOLUTION:

A policy statement like this can be time consuming to describe and confusing to implement. The system analyst needs to portray the logic of the policy as follow:

	Type of customer	Size of order	Discount
Discount Policy	Book stores	6 or more	35%
		less than 6	NIL
	Libraries and individuals	50 or more	25%
		20-49	20%
		6-19	15%
		less than 6	NIL

The decision table may be represented as follows:

Decision table

	Customer is bookstore	Y	Y	N	N	N	N
	Ordersize 6 copies or more	Y	N	N	N	N	N
	Customer librarian or individual		Y	Y	Y	Y	
IF	Ordersize 50 copies or more			Y	N	N	N
(Condition)	Ordersize 20-49 copies				Y	N	N
	Ordersize 6-19 copies					Y	N
	Allow 35% discount	X					
THEN	Allow 25% discount			X			
(action)	Allow 20% discount				X		
	Allow 15% discount					X	
	No discount		X				X

Example: **Payroll System**

Conditions/ Courses of Action	Rules			
	1	2	3	4
Employee type	S	H	H	H
Hours worked	–	<40	40	>40
Pay base salary	X			
Calculate hourly wage		X	X	X
Calculate overtime				X
Produce Absence Report		X		

Structured English Representation

BEGIN

BEGIN IF

IF Employee-Type is Salary

```
        THEN PAY base salary
    END IF
    BEGIN IF
        IF Employee-Type is Hourly
        AND Hours-Worked is <40
        THEN CALCULATE hourly wage AND PRODUCE       Absence Report
    END IF
    BEGIN IF
        IF Employee-Type is Hourly
        AND Hours-Worked is 40
        THEN CALCULATE hourly wage
    END IF
    BEGIN IF
        IF Employee-Type is Hourly
        AND Hours-Worked is >40
        THEN CALCULATE hourly wage AND CALCULATE overtime
    END IF
END
```

Pseudocode

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

Examples:

1.. If student's grade is greater than or equal to 60

```
    Print "passed"
else
    Print "failed"
```

2. Set total to zero

Set grade counter to one

While grade counter is less than or equal to ten

    Input the next grade
    Add the grade into the total

Set the class average to the total divided by ten

Print the class average.

3.

```
Initialize total to zero
Initialize counter to zero
Input the first grade
while the user has not as yet entered the sentinel
    add this grade into the running total
    add one to the grade counter
    input the next grade (possibly the sentinel)
if the counter is not equal to zero
    set the average to the total divided by the counter
    print the average
else
    print 'no grades were entered'
```

4.

```
initialize passes to zero
initialize failures to zero
initialize student to one
while student counter is less than or equal to ten
    input the next exam result
    if the student passed
        add one to passes
    else
        add one to failures
add one to student counter
print the number of passes
print the number of failures
if eight or more students passed
```

print "raise tuition"

Example: Pseudocode for making a cup of tea:

```
PROGRAM MakeACupOfTea:
 Organise everything together;
 Plug in kettle;
 Put teabag in cup;
 WHILE (Kettle is not full)
   DO keep filling kettle;
 ENDWHILE;
 Wait for kettle to boil;
 Add water to cup;
 Remove teabag with spoon/fork;
 Add milk;
 IF (sugar is required)
   THEN add sugar;
   ELSE do nothing;
 ENDIF;
 Serve;
END.
```