

System Analysis and Design
Prof. V. Rajaraman
Department of Super Computer Education and Research
Indian Institute of Science, Bangalore

Lecture - 17

We were talking about the specification of processors and we said that, there are three different methods of specifying processes. One is called structure English it is very similar to programming languages, which we have learnt already. The other two are what is called decision tables. The third is called decision trees. And, decision tables are useful for complex decision processes where, the number of conditions to be tested is quite large.

And, it also gives a non-procedural method of specifying the business rules. And, if a sequence of taking actions are sequence of taking, testing the conditions is important in any given problem. Then, you go to something called decision trees. So, I will look at all these trees. And today I will start with ((Refer Time: 02:14)) structured English for given word statement. I am starting with a very simple example, but normally in practice, your statements will be very, very long.

(Refer Slide Time: 02:29)

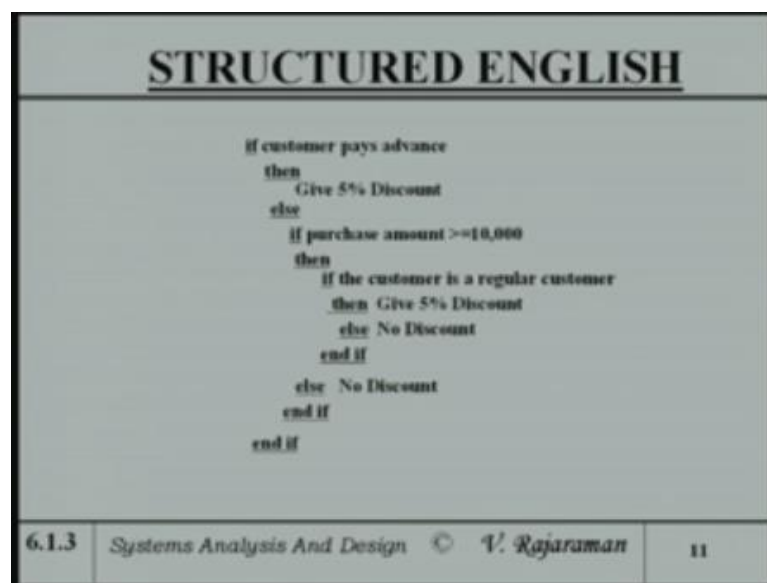
<u>STRUCTURED ENGLISH</u>		
<u>Customer Discount Policy</u>		
Give a discount of 5% if the customer pays advance or if the purchase is for Rs 10000 or more and the customer is a regular customer.		
6.1.2	Systems Analysis And Design © V. Rajaraman	10.1

When I say, customer discount policy what it really says is that, it is a business rule, which is used by the company to take a decision on when to give discounts. And, what condition should be fulfilled in order to give discounts. So, the business rule says give a

discount of 5 percent, if the customer pays advance or if the purchase is for 10,000 rupees or more and the customer is a regular customer.

So actually, if you look at this there is a, the action is giving a discount. Whereas, the conditions to be tested are customer paying advance or purchase order is greater than 10,000. Greater than equal to 10,000 and customer is a regular customer. ((Refer Time: 03:25)) there are conditions, which are three different conditions are involved in this. And an action is either giving a discount or giving no discount.

(Refer Slide Time: 03:36)



So, if I want to put it in structured English. I will say, if customer pays advance then give 5 percent advance discount. If, purchase amount is greater than or equal to 10,000 rupees. Then, if the customer is a regular customer then give 5 percent, else no discount end if. And then, the as you know else, if and else to be bracketed. And, the next else is corresponding to the latest if mainly the, that is if purchase amount is greater than equal to 10,000.

And then the end if, final classes ((Refer Time: 04:21)) is customer pays advance. That is for the whole classes. You can see here, the similarity between this and the way in which you will write a programming program, in a programming language. Because, this can be easily converted into a what is now known as procedural language. Procedural languages are the ones, which people normally use that is C language or C plus plus and so on or Java or what have you.

So, all procedural languages give a step by step procedure. Except the ((Refer Time: 04:59)) the English statement. Because, primarily as I said the English specification. Structure English specification is something which could be understood, both by the user and by the developer of the program. So, it is something of a familiar intermediate part. In fact, many people suggest, that during the development of a large project. It is important to break it up into smaller parts. And then at each level generate the testing to be done.

So, that testing is done simultaneously with the program development. And, so that the testing is not left at the end. Then, you ((Refer Time: 05:50)) find arrays which is not expected. So, this is called test driven programming and so on. So, but then this is only a tool to write a program later on. But, at that time the ((Refer Time: 06:06)) English statement it is easier to get the test data generated. Or test what you want to test. And in fact, get these tests cleared with the user.

((Refer Slide Time: 06:22))

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
6.1.4	Systems Analysis And Design © V. Rajaraman			12

Saying that these are the ones I am going to test. So, same thing can be expressed decision table. As you can see, it is a non-procedural thing. In other words, it is does not say in detail, what are done in sequence. And, what actions are taken in sequence ((Refer Time: 06:42)) in other words in a structured English ((Refer Time: 06:43)) more or less like, it is called procedural. Because, ((Refer Time: 06:44)) step by step you go and at the

end or during the steps you talk about a certain action to be taken, in this case, during either discount or no discount.

Whereas, in this case it is a fabulous structure. And at a glance you can say for instance the advance payment is made, there are number of business rules. Rule one says, is advance payment is made the answer is yes, then give five percent discount. The other purchase amount and regular customer are irrelevant. So, that is why put dashes. Wherever, condition testing result by testing a condition is irrelevant, you put a dash. If the conditions got to be true, you put a yes. And, if the conditions got to be false then you put a no.

Rule number 2 says, if advance payment is made ((Refer Time: 07:43)) no means, No advance payment is made. But, the purchase amount is a greater than or equal to 10,000. And, there is a regular customer. The rule says that, in that case you give a discount of 5 percent. And, rule 3 again says that, no advance is paid and he is not regular customer. And, the purchase amount is not is greater than equal to 10,000, even then you do not give a discount.

And, in the last case namely no advance is paid no purchase amount smaller than ten thousand. Then you also give no discount. So, this is essentially, conversion of the word statement. Or, a statement of business rules given by the customer to an equivalent table. And you can see, the advantages of the table is that any user, can easily visualize these rules. It does not have no any kind of programming language. And, does not get cluttered up with the whole lot of if then else statements and then bracketed if then else and so on.

So, when the number of conditions increase the bracketed, if then else become little confusing. And, that is the reason why a tabular structure is somewhat more you know, easy to understand for the user.

(Refer Slide Time: 09:25)

DECISION TABLE-EXPLANATION			
<ul style="list-style-type: none">▪ 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			
6.1.5	Systems Analysis And Design	© V. Rajaraman	13

So, conditions are questions to be asked, yes is yes, N is no, dash is irrelevant. And, X against an action, if I put a- X against an action ((Refer Time: 09:34)) that is means the action is to be taken. Give 5 percent discount, the first two rules I put a X against the give 5 percent discount. And, last two rules, I say give no discount. So you use actually a cross against actions, which are to be taken under dash against actions. My wish need not be taken or which are not really taken.

(Refer Slide Time: 10:04)

DECISION TABLE-EXPLANATION			
<p>Rule 2 in decision table DISCOUNT states:</p> <p><u>if</u> no advance payment <u>and</u> purchase amount ≥ 10000 <u>and</u> regular customer <u>then</u> give 5% discount</p>			
6.1.5	Systems Analysis And Design	© V. Rajaraman	14

So, the rule 2 as I said I already explained that of the decision table discount states, if no advance paid payment. And, purchase amount is greater than equal to 10,000 and regular customer. Then give 5 percent discount.

(Refer Slide Time: 10:20)

STRUCTURED ENGLISH

- Imperative sentences- Actions to be performed should be precise and quantified

Good Example: Give discount of 20%

Bad Example: Give substantial discount

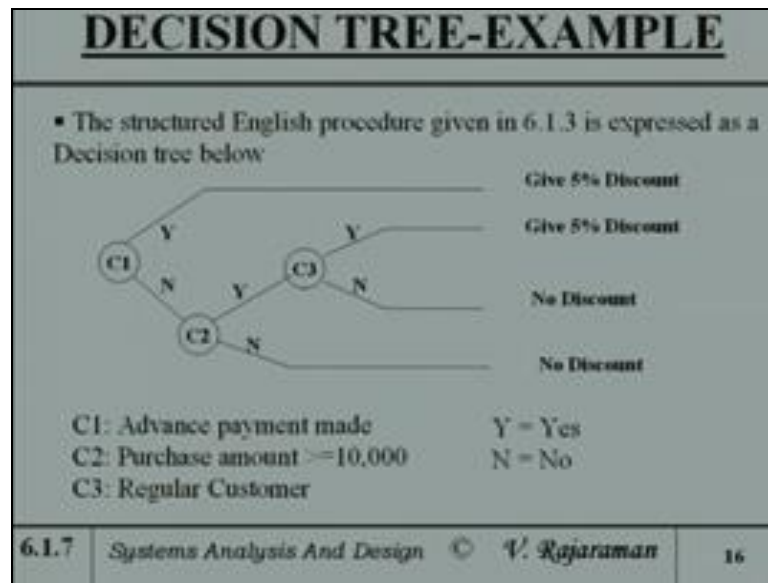
6.1.6 Systems Analysis And Design © V. S.

The slide features a small video inset in the bottom right corner showing a man in a white shirt speaking. The footer contains the text '6.1.6 Systems Analysis And Design © V. S.'.

Actions are imperative sentences. see the you know the imperative sentences are actions to be performed should be precise and quantified. In whether it is structured English or decision tables any system ((Refer Time: 10:38)) the actions which the program is ((Refer Time: 10:41)) had to be quantified. You cannot for instance say ((Refer Time: 10:47)) substantial discount. Because one does not know, what substantial really means whereas give 20 percent discount as a clear meaning.

So, in all cases particularly in programming, you have to give precise actions to be taken. Very often, when people gave business rules they use such terms as giving substantial discount is given. But; obviously, you cannot program it. What is substantial you see is it 10, is it 20, it is 30. So, the whole point is that the ambiguity, which is inherent in a English statement such has English words such as substantial is removed in the case of a, you know action which is specified.

(Refer Slide Time: 11:41)



The same structured English procedure can be expressed as a decision tree. It is somewhat like a flowchart. In the sense that, you say advance payment is made, then give 5 percent discount. And you can go down, if advance payment is not made, you check the condition two. Which is purchase amount is greater than equal to 10,000. If the answer is not automatically you give no discount. And then, you test the next condition, which is to ask the question is a regular customer is a regular customer and he also advance is not paid but the purchase amount is greater than equal to 10,000.

Then give 5 percent discount. So, this the otherwise no discount. So, the point is here also it is more procedural in the sense that, it gives the sequence in this conditions are tested, because the final action. Some extent some people are comfortable with this specification. Whereas, many people are ((Refer Time: 12:53)) more comfortable with a tabular structure, because everybody is used to tabular structure in real life.

(Refer Slide Time: 13:05)

<u>STRUCTURED ENGLISH</u>			
▪ Operators -Arithmetic : +, -, /, *			
Relational : >, >=, <, <=, =, !=			
Logical : and, or, not			
Keywords : if, then, else, repeat,			
until, while, do, case,			
for, search, retrieve, read,			
▪ Delimiters – {, }, end, end if, end for			
6.1.6	Systems Analysis And Design	© V. Rajaraman	17

Now, we will describe structured English in ((Refer Time: 13:10)) detail. And, in structured English as I said, it is not a programming language. But still certain types of notations are to be used in the structured English. So, that there is no ambiguity. And also, later on it is going to be used by a programmer to convert that structured English into an equivalent program. So, if you use there is clarity in the notation, which we use then the clarity in the notation ((Refer Time: 13:51)) very, very important for converting without any kind of a misinterpretation.

This strict syntax is not necessary as I said, unlike programming language. But, strict syntax is necessary only to kind of make it so that, you do not make any logical errors, while developing it. So, this is the whole idea of having specific definitions. There are arithmetic operators like, plus for add, minus for subtract slash for divide and star for multiply. And relational operators usual greater than less than equal to, less than, less than equal to, not equal to and logical connectors.

And or not which are logical connectors and there are keywords. If, then, else, repeat, until, while, do, case, for. These are all these, the if then else repeat while and so on are all there in those programming languages. The only extra things I have really put is search and retrieve. Which are really higher level operations and they have to be converted to lower level structures in a programming language. But, in the structured English, it is easier to understand you say search and retrieve.

Because, read is there in every language. Read and write also will be there or print or whatever. And delimiters, delimiters are the ones which brackets which are used. In fact, end if and end for are also delimiters. That means, they effectively give the bracketing information. For if, if as got to be bracketed with end if. So, that is there to be able to give you an idea. that you have not made an error in nesting of the loops. Nesting of the loops really becomes a very often. It is a source of many errors.

And so the nesting as to be done with a ((Refer Time: 16:26)) amount of care and for that reason, we provide the so called delimiters. Like ((Refer Time: 16:36)) and so on.

(Refer Slide Time: 16:39)

<u>STRUCTURED ENGLISH-</u> <u>DECISION STRUCTURES</u>	
<pre>if condition then { Group of statements } else { Group of statements } end if</pre>	
Example: if(balance in account \geq min.balance) then honor request else reject request end if	
6.1.8	Systems Analysis And Design © V. Rajaraman 18

Then there are conditional statements. Like if, if condition then carry out a group of statements. Else another group of statements and end if. We can say that if and end if are bracketed. So, if balance in account is greater than or equal to minimum balance ((Refer Time: 16:57)) an example you can see that. If balance in account is greater than, equal to minimum balance. Then, honor request else release reject request. In other words, it is very simple statement.

In for instance when you go to take some money from the bank, they look at your balance in the account. If it is greater than equal to minimum balance required for that account. Then you give it. give the money otherwise you do not. That effect is what this is.

(Refer Slide Time: 17:31)

<u>STRUCTURED ENGLISH-CASE STATEMENT</u>		
<u>Case (variable)</u> 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} <u>end case</u>		
Example : <u>Case</u> (product code) product code =1 : discount= 5% product code =2 : discount =7% None of the above : discount=0 end case		
6.1.9	Systems Analysis And Design © V. Rajaraman	19

The case is number of alternatives which are there. And case as you know, in any language you have got case statements. Whereas, the number of different variable equal to P the alternative P and so on. Instead of having this variable equal to P, you can also have some kind of a testing. But, by and large this is a because testing is already there elsewhere.

The equality testing is something which is very often done. And so the case also is this, if you check these kinds of things are very useful. If your codes, you have given codes for certain variables. And if the codes match, then you do some action. For instance, you may have given a code of 1 for a person above 25 years of age.

A code 2 for a person who is between 25 and 45, a code 3 for somebody between 45 and 65 and so on. So, when you check the having checked. For instance, either you check it in the program or when they do it manually, they just give the code numbers. Say 1, 2, 3, 4, and you say whether the code is one base some action. The code is to do some other action.

So, very often similarly for codification is used not only for age, it is not a very precise example. Whether, another good example of codification is product codes. We have certain products all products have certain universal codes given. So, for each product code, you may take some different action. So, you can compare project code against the

based on the truth or false root of the variable being equal to product code, you take some action.

Example, say if product code is equal to 1 ((Refer Time: 19:48)) discount of 5 percent. If product code is equal to 2 give a discount of 7 percent. So, different products different discounts. In fact, the number can become very large in any business situation. And so the number of the statements ((Refer Time: 20:07)) within a case may be quite large.

(Refer Slide Time: 20:13)

<u>STRUCTURED ENGLISH- REPETITION STRUCTURE</u>		
<pre><u>for</u> index = initial to final <u>do</u> { statements in loop } <u>end for</u></pre>		
<p>Example: Total = 0</p> <pre>for subject = 1 to subject = 5 <u>do</u> total marks = total marks + marks(subject) write roll no, total marks <u>end for</u></pre>		
6.1.10	Systems Analysis And Design © V. Rajaraman	20

And it is a very easy way of seeing what you do? And there is a repetition structure. Repetition structures, which is their many programming language. For index you for initial to final do statements in the loop. Example, start with total equal to 0, for subject 1 to subject 5 do total marks equal to total marks plus marks in the subject, write roll number, total marks end for.

The point is here you are going through, for you are finding out the sum of the marks in 5 subjects. And then, finding out the total marks. So, you start with total equal to 0 and that the end of it ((Refer Time: 20:59)) total marks available. Because, layed subsequently it will be used, it is only very small part of the of the program.

(Refer Slide Time: 21:07)

<u>STRUCTURED ENGLISH- WHILE LOOP</u>		
<pre>while condition do { statements in loop } end while</pre>		
<p>Example : while there are student records left to do read student record compute total marks find class write total marks, class, roll no end while</p>		
6.1.11	Systems Analysis And Design © V. Rajaraman	21

While is somewhat similar it is also repetitive structure. While condition, do statements in the loop, end while. While there are students records left do read student record compute total marks find class, write total marks, class, roll number, end while. So, the here in this case again I take an example, where number of records of students are taken one after the other read. And then, some action is taken and the result is also printed in this case.

(Refer Slide Time: 21:45)

<u>EXAMPLE</u>		
<pre>Update inventory file</pre> <pre>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</pre>		
6.1.12	Systems Analysis And Design © V. Rajaraman	22

There is another example, for each item accepted record you know, when a items come under the store. Then, you as the very first earlier examples I talked about a situation, where products are received by a company. And sent for inspection and after inspection if they are accepted. Then, it is goes to the stores and stores does the updating of the database.

So, in this case for each item accepted which the acceptance note goes from the inspection office to the inventory control there is stores office, search inventory file. So, the action to be taken by the store is search inventory file using item code. So, the search is a higher level construct are using here and if successful.

Then, update retrieve inventory record write updated record back in the inventory file using the accepted record. There is a number of amount, amount came is came and so on is added to existing inventory record and stored back. Else, create new record in inventory, enter the accepted record in the inventory file.

In other words it is not already existing in the inventory, it is a new item which is come in. So, then what you do is, you enter it is co ordinate whatever identify the information is there for that. And then, do the rest of it. In other words put it in the file. So, this is a again repetition, because for each item accepted.

So, one by one when the items are accepted, then you do lot of this. And may be at the end of the day, you have a an updated file, which is up to date status of inventory in the store.

(Refer Slide Time: 24:02)

DECISION TABLE-MOTIVATION		
<ul style="list-style-type: none">▪ A procedural language tells how data is processed▪ Structured English is procedural▪ Most managers and users are not concerned how data is processed-they want to know what rules are used to process data.		
6.2.1	Systems Analysis And Design © V. Rajaraman	23

A procedural language tells how data is processed. So, that is exactly what I was telling at the beginning, it is a step by step set of procedures or language, which talks has a detail about what is to be done, when it is to be done and so on. And what sequence it is to be done. Whereas this is the ((Refer Time: 24:34)) advanced procedures. Most managers and users are not concerned how data is processed. They want to know, what rules are used to process data.

So, that is important thing as for as the manager is concerned. Are you using the right rules, are the rules interpreted correctly. The rules are not interpreted correctly, then at the back stage itself. Where, you are getting the requirements specification. They can change the rules or have you not understood the rules. As given by the user due to some communication gap. You should at that stage itself with the users concurrence, change the business rule.

So, most bus- managers are concerned only about, what rules are doing process, rules are being followed. They are concerned about, how you are going to do it. Because, that is the least of any managers concern.

(Refer Slide Time: 25:36)

DECISION TABLE-MOTIVATION			
<ul style="list-style-type: none">▪ Specification of what a system does is non-procedural.▪ Decision Tables are non-procedural specification of rules used in processing data			
6.2.1	Systems Analysis And Design	© V. Rajaraman	24

So, specification of what a system does is nonprocedural. See, in the case of decision tables just specification of what a system does. Decision tables are nonprocedural specification of rules used in processing data.

(Refer Slide Time: 25:56)

ADVANTAGES OF DECISION TABLE			
<ul style="list-style-type: none">• Easy to understand by non-computer literate users and managers• Good documentation of rules used in data processing.• Simple representation of complex decision rules• Allows systematic creation of test data			
6.2.2	Systems Analysis And Design	© V. Rajaraman	25

The advantage of decision tables are easy to understand for non-computer literate users and managers. In other words the managers of course, understand little bit of what a language means and so on. But, there are no about details of any programming language or even structured English. Even structured English ((Refer Time: 26:18)) strange to

them. So whereas, everybody understand the table. Good documentation rules used in data processing. Actually, documentation is as important as program.

In other words, the ones good documentation is there. Program writing becomes straight forward. And simple representation of complex decision rules. Otherwise, representation of a complex set of decision rules. Becomes at a glance, you can see what is going on. And, allows also systematic creation of test data, which we will see there is a ((Refer Time: 26:59)) set while you are developing programs. You should be able to also create simultaneously test data to check the program, never leave testing to the end. You do incremental testing as and when you go.

(Refer Slide Time: 27:15)

<u>ADVANTAGES OF DECISION TABLE</u>			
<ul style="list-style-type: none">•Tabular representation allows systematic validation of specification detection of redundancy,incompleteness & inconsistency of rules•Algorithms exist to automatically convert decision tables to equivalent computer programs.			
6.2.2	Systems Analysis And Design	© V. Rajaraman	26

Tabular representation allows systematic validation of specifications. Detection of redundancy incompleteness and inconsistency of rules. So, these are very important as I said, many are very often rules are rules grow over time. And because, your grow over time, there is always a contraction between something which I said earlier or something which is said later. And this contradictions are to be avoided. Also some cases will rules are not stated, that is called incompleteness.

In the case, of decision tables algorithms exist to automatically convert decision tables to equivalent computer programs. And also algorithms exist to check for the ambiguity in tables.

(Refer Slide Time: 28:14)

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 \rightarrow 21 C1 Deposits \rightarrow Rs 100: C2			
Identify Actions: Classify account as A, B or C Do not open account			
6.2.3	Systems Analysis And Design	© V. Rajaraman	27

I will give an example of how you go from a statement, word statement to a equivalent table. The business rule says that a bank uses the following rules to classify new accounts. A depositor's age is 21 or above and if the deposit is rupees 100 or more. Classify the account as type A. If the depositor under 21 and is deposit is over 100 rupees classify it as type B. If the deposit is 21 or above over and deposit is below 100 classify it as C ((Refer Time: 29:03)) depositor is under 21, and deposit is below 100 do not open account.

In other words if 21 or over and deposit is below 100, classify it as C otherwise do not open. Because ((Refer Time: 29:20)) below 21 and deposit is below 100 do not open account. ((Refer Time: 29:24)) The numbers arbitrary, the age 21 the deposit 100 and so on are this arbitrary, just I pick it out to the hand. And, you can actually you any given bank will have similar rules which will not be related to age and may not be related to the most operand will be ((Refer Time: 29:47)) related to the amount of deposit.

Now, the first thing you have to do when a statement like this is given is to identify the conditions. When you go through this statement you say, you see that, if the depositor age is 21 or above. So, immediately I detect a condition, age 21 and above greater than equal to 21, let us come to C1. And the second condition is that, if the deposit 100 rupees or more. So, condition two is deposit greater than equal to 100. And there is no other condition. There are only age and deposit amount.

And the actions we have taken are classification of accounts, if they are open and a denial if it is not opened. So, we are actions are classify account as A, B or C or do not ((Refer Time: 30:43)) open account. So, ((Refer Time: 30:45)) identify conditions and the actions to be taken. Next step is to develop the table, based on the conditions and actions.

(Refer Slide Time: 30:56)

DECISION TABLE FROM WORD STATEMENT				
<u>Condition Sub</u>				
<u>CONDITIONS</u>	Rule 1	Rule 2	Rule 3	Rule 4
C1 : Age ≥ 21	Y	N	Y	N
C2: Deposit ≥ 100	Y	Y	N	N
<u>ACTIONS</u>				
A1: Classify as A	X	-	-	-
A2: Classify as B	-	X	-	-
A3: Classify as C	-	-	X	-
A4: Do not open Account	-	-	-	X
<u>Action Sub</u>				
6.2.4	Systems Analysis And Design	©	V. Rajaraman	28

Like, the conditions are in this case C 1 age greater than equal to 21, condition 2 deposit is greater than equal to 100. And the actions are classify as A, classify as B, classify as C and do not open account. There are four different actions the rule you can go through the rule one by one very easy. If age is greater than 21 and deposit is greater than equal to 100 classify as A.

If age is less than 21 deposits is over 100 classify as B. If the age is above 21 deposits is greater than is less than 100, then classify it as C. If the age is below 21 and deposit is also less than 100 do not open an account. In fact, I am giving every possible rule we are two conditions. And each condition can have a yes or no answer.

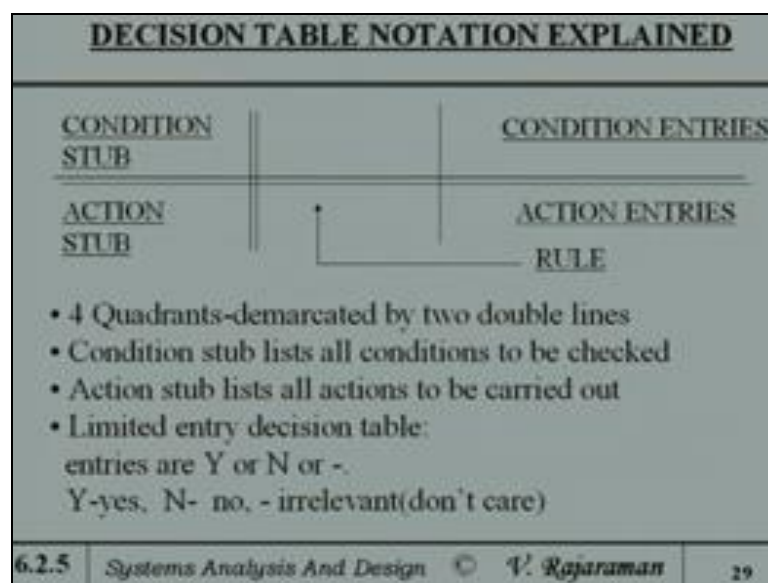
There will be total of 2 squared rules. That is 2 to the 2 squared is; that means, 5 rules. Because, I exhausted the combinations possible, yes, yes, no, yes, yes, no, no, no. And each one I take particular action. Now, there is some terminology used in decision table as a terminologist primarily as a definitions of that everybody understands, the structure of a table. And what it really means.

Because, there is necessity for some kind of a notation, just like structured English there is a need for a notation. In the case of decision tables also, there is a need for a certain notation. And that notation, in this case the conditions are all put together, in the left corner of the 4, it is a 4 quadrants, the table has got 4 quadrants as a left top quadrant. And where the list conditions are listed, this is called the conditions stub.

And the one delayed and the left hand side of the double line is called action stub, there is actions to be taken. So, the conditions stub is states the conditions in to be checked at. The actions stub list of actions to be taken and the entries, the condition entries which is the there on the top right hand quadrant are the answers to those questions, which are asked.

And below that are is called ((Refer Time: 34:03)) called action entries. So, that is the condition step, action step, condition entries and action entries.

(Refer Slide Time: 34:11)



So, I am summarize that here. So, there are four parts and if four parts are divided by 2 double lines, just to 4 quadrants ((Refer Time: 34:22)) two double lines, condition stub lists all conditions to be checked. Action lists all actions to be carried out and a table is called limited entry decision table.


If the answers to conditions are either yes no or dash. Dash is irrelevant; that means, whether the answer is yes or no does not matter. So, that irrelevant is shown by a dash.

(Refer Slide Time: 34:53)

<u>DECISION TABLE NOTATION</u>			
<u>EXPLAINED</u>			
<ul style="list-style-type: none">• 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			
6.2.5	Systems Analysis And Design	© V. Rajaraman	31

X against an action, states it is to be carried out. A dash against an action, states it is to be ignored not to be it carried out. Entries in a vertical column specifies a rule. In other words as I read it out ((Refer Time: 35:08)) each vertical column here rule 1, rule A 2 and so on. Each one specifies a rule, where you test the condition and take appropriate action.

(Refer Slide Time: 35:21)

<u>DECISION TABLE NOTATION -</u>			
<u>CONTD</u>			
<ul style="list-style-type: none">• Order of listing conditions irrelevant Conditions may be checked in any order• Order of listing actions important			
6.2.6	Systems Analysis And Design	© V. S	

Very important point about decision tables which makes it nonprocedural. And which also makes it somewhat simpler to understand and later on to process, like computer to detect

ambiguity and so on is that, the order of the conditions is irrelevant. That is conditions can be tested in any order

Now I can even if I write C 1, C 2, C 3 I can reorder it a C 3, C 2, C 1 or any order. And does not really matter ((Refer Time: 36:04)) simple reason that each rule will depend on the answers to all the conditions in that step. And what order I check, I should be irrelevant in the case of tables. In other words the definition of table says, that order of testing conditions irrelevant.

However, order of listing actions is important. That means, if an action is listed first ((Refer Time: 36:32)) first. An action is ((Refer Time: 36:34)) number 2 is got to be second next interval. If there is sequencing of actions whereas, there is no sequencing of conditions.

(Refer Slide Time: 36:45)

DECISION TABLE NOTATION - CONTD

- Actions listed first carried out first
Sequential execution of actions
- Rules may be listed in any order

6.2.6 Systems Analysis And Design © V. S.

Sequential execution of actions and rules may be listed in any order. Because, conditions can also be listed in any order, rules can be list. Here, I can change around the order of the rules. Because, ultimately rules where it occurs is irrelevant as far as the jar is concerned.

(Refer Slide Time: 37:12)

INTERPRETING DECISION TABLE-ELSE RULE			
	R1	R2	ELSE
C1: Is applicant sponsored	Y	Y	
C2: Does he have min qualification	Y	Y	
C3: Is fee paid?	Y	N	
A1: Admit letter	X	-	-
A2: Provisional Admit letter	-	X	-
A3: Regret letter	-	-	X
Interpretation			
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 cases send regret letter. The else rule makes a decision table complete			
6.2.7	Systems Analysis And Design	© V. Rajaraman	34

One more type of a decision table because it is something called as else rule. In other words else rule is cover all, in other words in all cases which are not specified strictly in a set of rules. All that are clubbed together and you take one column action, it is like the else, if then else thing. Except, that in this case else may encompass many possibilities.

So, for instance a business rule is says that, if application is sponsored and his dues have been paid. These are rules for admission of students to a particular short term course. So, the applicant is sponsored by the organization to attend the short term course. And if has minimum qualifications to profit in that course. And if his fees is paid, then send ((Refer Time: 38:19)) right away.

If the applicant is sponsored he has minimum qualifications. But, the ((Refer Time: 38:31)) fees not paid. So, the you have already filtered the application and found out that he is eligible. But, his fees not paid. So, send a provisional admit letter, which may state that kindly pay the fee by submit date for your admission to be valid. Otherwise, you will not be admitted.

Else, in all other cases send a ((Refer Time: 38:57)) letter. Because, if the application is not sponsored automatically proceed. And if the he does not have any qualification, then also it goes automatically. And you can say, because yes and yes are covered, any one of the first conditions is known, the last condition is irrelevant.

So, any case in all these cases you effectively say that send a regret letter. The else rule actually makes a decision table complete, what I mean by decision table complete is that all possible conditions, all possible rules which can occur are covered. So, in other words it is even if you miss out something, it is an exception action

And sometimes it is could be quite dangerous, to an take an exception action I have to be careful in using an else. And clubbing a lot of things together the else. It is better to always to look at the complete part.

(Refer Slide Time: 40:20)

DECISION TABLE FOR SHIPPING RULES				
	R1	R2	R3	R4
C1: Qty ordered \leq Quantity in stock?	Y	Y	N	N
C2: (Qty in stock - Qty ordered) \geq 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	-
A3: Qty shipped = 0	-	-	-	X
A4: Qty in stock = 0	-	-	X	-
A5: Back order = qty ordered - qty shipped	-	-	X	X
A6: Initiate reorder procedure	-	X	X	X
A7: Qty in stock \leftarrow Qty in stock - Qty shipped	X	X	-	-

This another table here, which the point the reason why I have put this table is here the action, testing the sequence of testing actions becomes relevant. This is a set of rules for to be given to a supplier. And when the order is to be initiated, if quantity ordered is less than equal to quantity in stock.

And quantity in stock minus quantity ordered is less than equal to reorder level. That means, it is above it. Then, required you know the quantity shipped is equal to quantity ordered same. Because, you have this rule and then, we update the quantity in stock as quantity in stock minus quantity shipped.

So, the point really in this case is the fact that, you first do the quantity shipped is quantity ordered. And then, you do the... This is a set of rules for a vendor to follow,

when the order comes and to him and he has to supply. So, when quantity ordered is less than. That is he has got enough stock in his inventory and it is above the reorder level.

Then, you can immediately ship the whatever is order. And of course, update that is whatever is shipped is reduced. And the next rule says, the quantity ordered is less than equal to quantity in stock. That means, he has got enough quantity to ship and quantity ordered is minus quantity, the quantity in stock minus quantity ordered is less than equal to reorder level yes, if it is below reorder level.

That means, the vendor in order to has to kind of bring up the stock to a certain level. Because, is a any inventory control of a vendor, whenever he supplies items I to get an example of a medical shop for instance. Medical shop when a customer comes and buys some medicines. He notes down, by that there is enough left in the shelf. If it is not enough left in the shelf in terms of what is consider as a reorder level. He will order that he will reorder that amount otherwise he will not do it.

So, in this case, it is important to kind of the rule says that, you ship the quantity. And then initiate the reorder procedure right away. That means, you are calling another program, you might say. So, which will initiate the reorder procedure and then you update the quantity in stock by removing whatever was supplied. The third rule says, the quantity added is less than and equal to quantity in stock. In other words, he has ordered an amount, which is larger than what you have in stock. That means, you cannot supply the entire amount.

You can ask the customer, when he whether he will take a partial order. In other words suppose he ordered 1000 and you have got only 800. You ask whether he will take 800 or he will take 1000 and nothing else. In some cases he may not. So, if partial segment is not is ok, then you ship the quantity. Your quantity in stock automatically becomes 0, because the quantity ordered is greater than quantity in stock. And you have to kind of have a back order file.

In other words, you have a supply of two hundred which you did not supply later on. So, backorder put on the backorder file quantity added now it is quantity shipped. So, that you ship that amount. And then initiate reorder procedure automatically. We have to do that, because we have no stock. And last rule says, if quantity in stock is ordered is greater than

quantity in stock, you do not have a stock. And the customer will not accept partial order. That means, you do not ship anything. Quantity shipped is 0.

But, you have to reorder, because you know he may ask whether some item or somebody else ask for the item so reorder. And so backorder of course, is the what is to be shipped later on. If you know, he asks for 1000 and you do not have 1000, he is not willing to accept the partial quantity. Then you put the 1000 in the backorder file. And when you got enough stock, you supply 1000 time.

At that time of course, there may be one more business rule saying that, if the delay, if more than three or four days he will not accept. So, that would complicate. In other words the business rule may be more complicated than this. But this gives a sample of the business rules normally used for in this case, when an order comes to a, he has got a order processing system. When, an order comes to a vendor. How does he process that order.

What rules will he follow to process the order and it is a very, very elementary example. When just to so that I can fit in the whole thing in one transparency. Normally, order processing procedures in many companies will be quite complex. They may have many, many conditions to be checked. And I will come to that later on, in terms of what happens, if the number of conditions become too large.

(Refer Slide Time: 47:05)

EXTENDED ENTRY DECISION TABLE						
<ul style="list-style-type: none">• 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%

6.2.9 Systems Analysis And Design © V. Rajaraman 36

We so far we looked at tables, where the entries the condition entries. Are only yes or no or dash. These types of tables are not is not restricted in the terminology ((Refer Time: 47:29)) decision tables. And similarly, actions are put X or dash. And that are, that also not really the only actions you can specify. There is something called extended entry decision table. Where, it is used for descriptive, then the limit entry table. And which also can be converted easily into limit entry table.

Almost automatically by a computer program. In case, the advantages of extended entry decision table is easy for people to understand. See, the number of rules will become less somewhat smaller. And it is easier for a non computer person to understand the rules. But, again it is non-procedural, because it is non-procedural it is a as I said, it is going to again easy for a person to understand. So, in this case, the product code will be 1. And customer code is A. And ordered amount is less than equal to 500.

You see, in this case the rules, the rules are not the, rules are not yes or no. I put the actual values of product codes. In this case the product codes are either 1 or 2. In the case, the product code is A. and customer code is also there are three possible customer codes. And order amount there are two I am using. One is less than equal to 500 or greater than 500. So, this again is a discounting policy ((Refer Time: 49:08)) or particular company.

The way in which I read these rules is that if the product code is 1, and the customer code is A. And the order amount is less than equal to 500 give a discount of 5 percent. If, product code is 1, customer code is B. And order amount is less than equal to 500 give 7 and half percent discount. And third rule says that product code is 1, customer code is A. Order amount is greater than 500 give 7 and half percent discount. You see that, there is a difference between rule 1 and rule 3.

Rule 1 is for less than equal to 500. And rule 3 is for greater than 500. ((Refer Time: 50:04)) these two cases there are two different discount processes. And rule 4 says that the customer code is B. And order is above 500. Then give 10 percent discount. That means, again you see the difference between R 2 and R 4. The first two conditions ((Refer Time: 50:25)) the third condition is different. And so there are the actions are different.

And then the codes may be assigned based on the convenience or based on whatever policy a company follows. A type customer will be different type. B type customer and so on. If the product code is 1 and the customer code is C. Then give 6 percent discount. And the product code is 2 give 5 percent discount everybody. In the case of you know customer code of C does not matter, whether it is 500 or above and so on. So, I put a dash there to say that, that condition is irrelevant.

So, ((Refer Time: 51:13)) only first two conditions mainly product code equal to 1. And customer code equal to C has to be checked. And I ignore the third condition. And I will take the action of giving 6 percent discount. And, if the product code is 2 ((Refer Time: 51:28)) discount of 5 percent is given to everybody. So, this is what this particular table says, I interpreted rule after rule. The reason why I am doing it, I could have done by earlier on.

In other words, I could have started with a set of rules given. And converted the rules into a into a equivalent table, which I did in the previous case. See the very first example. In this case, the reason I am doing this is to show you, how simple it is to explain what the table is to every person and that is a great advantage. See you that is a greatest advantage of a tabular structure.

(Refer Slide Time: 52:20)

MIXED ENTRY DECISION TABLE						
Can mix up Yes, No answers with codes						
	R1	R2	R3	R4	R5	R6
C1: Product code = 1?		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, software available to translate DTs to programs, ease of checking etc.						
6.2.10	Systems Analysis And Design © V. Rajaraman					37

We can also mix up the extend entries with limited entries, that is called a mixed entry decision table. In this case you can see, the product code equal to 1 or 2 are the only t

two alternatives So, I am able to say yes, yes, yes, no. And customer code is that there is a there should be a yes. And the first the rule 1. There should be yes A yes ((Refer Time: 53:01)) the Y unfortunately is missing and the typing. So, it is a printing error. So, it has to got to be Y A Y. Product code is 1, customer code is A and amount is less than 500. Then 5 percent discount.

Let us say same table, but in a different form. We have to put yes and no together. Because, of the yes and no in some cases and codes in some other cases. The reason why, these all work the next entry, this is called mixed entry table. We, use mixed entry table, wherever the condition to be checked as only two alternatives. Like in this case, ((Refer Time: 53:43)) you have only a one or two as two alternatives. So, you can say product code equal to 1 or no. Similarly, order amount is only less than 500 or greater than equal to 500.

So, that is the reason why you can say, use yes or no. So the, this kind of a table is also allowed. In other words, the notation of tables ((Refer Time: 54:04)) limited entry tables. That limited entry tables are the ones, which are very useful. And also is one of the earliest kind of a table, which is to be started. And later on we see, that it is a very useful thing to have from the point of view of actually, automating the process of converting into a program. Also to some extent generating programs, I mean they are generating test data and so on.

But, there is no, you know depending on the situation. Depending upon a particular situation, you could either use limited entry or extended entry. See as I said, the convenience are based on the types of alternatives. And the number, if the types of alternatives are large in number. Then instead of increasing number of conditions, I could actually put codes A B C.

In other words, if I had used the limited table in this case, then you know the ((Refer Time: 55:22)) see, I would have what would I do is customer code equal to A, I had put yes or no. Customer code equal to B yes or no. And customer code equal to C yes or no. ((Refer Time: 55:35)) the same condition. Condition two will be, will kind of get expanded, to three conditions, which are unnecessarily matter of the table. So, in the case of things like codes somewhat like case tables you know.

The case the code occur, it is about it does not clutter up the structured English program.

(Refer Slide Time: 55:59)

LINKED DECISION TABLE					
Decision table 1			Decision table 2		
Salary point=6	N	e	Salary point=2	N	N
Conduct OK?	Y	1	1 yr as class 1 officer	Y	N
Diligence OK?	Y	s	Departmental test Passed?	Y	-
Efficiency OK?	Y	e		N	-
Go to table 2	X	-	Advance to next salary point	X	-
No promotion	-	X	No promotion	-	X
Decision table 3			Go to Table 3	-	-
Complete departmental Course	Y	else		-	-
1 yr since last increment	Y			X	X
Advance to next salary point	X	-		-	-
No promotion	-	X		-	X
			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		
6.2.11 Systems Analysis And Design © V. Rajaraman					
					38

This is you can link decision tables together. The linking means that, the one table can link another table, which can in turn lead to another third table. In other words, there is a sequencing of tables possible. And linking is very important and necessary in very complex problems. For a simple reason that, suppose I had not linked all of this. You see, because the table number 1, there are 4 conditions. Table number 2 there are 3 conditions. And table number 3, there are 2 conditions.

So, if I add 4 plus 4, 4 plus 3 7 plus 2 9. So, 9 conditions, if I put all 9 conditions in one large ((Refer Time: 55:55)) table. There will be 2 to the power 9, number of rules I have to cover. And 2 to the power 9 is how much, it is 512 rules. And nobody will be able to understand it clearly. So, it is very important to also in any table, limit the number of conditions. To be tested in a particular table to number which is again somewhat like magic number 7 plus or minus 2, your 9 ((Refer Time: 57:31)) too much.

Normally, you try to keep the table with not more than six or seven conditions. If it becomes larger than six or seven conditions, then you start trying to linking up, linking up with other tables. Then break it up, break the table into smaller parts. Somewhat, like leveling in D F D. You, have one complex table, you got to kind of break it up into smaller tables. Just like a complex D T complex D F D, you break it up into level parts. Similarly, in a complex decision table.

You kind of make it into link parts. Which are all using the same conditions, but add a final level, where the conditions are limited. In this case, I will interpret the table like this. If there is a set of promotion rules. Which is given to a company in terms of under what conditions, should person be promoted or not. And I would, let you look at this with some care. And interpret yourself and try to get the business rules yourself. And you will understand, how linking take place.

You almost ((Refer Time: 58:58)) self evident the way in which it is done. So, next time I will start with this point and explain this linking issue. And also the conversion of this into a equivalent word statement.