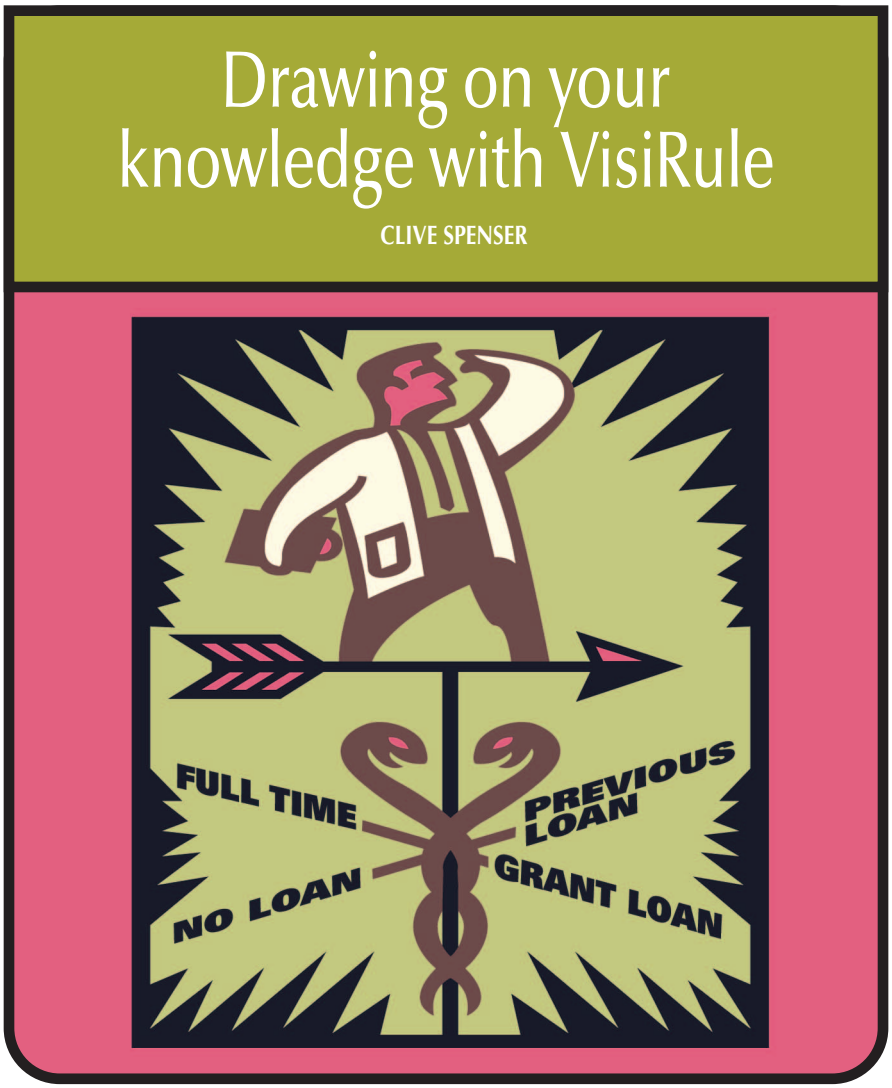
Drawing on your knowledge with VisiRule

CLIVE SPENSER



RULE-BASED SYSTEMS HAVE BEEN around for a long time without breaking through to mainstream computing. Rule-based systems represent knowledge in terms of a bunch of if-then rules, a bunch of facts, and some interpreter controlling the application of the rules, given the facts. One reason for their lack of success is that they have been largely text-based systems that often require specialist rule authors. Decision trees are useful for capturing structured decision-making processes. This approach is useful for troubleshooting and configuration applications. The knowledge for these applications is often structured into a set of steps and decision points. Binary decision trees have been used to build predictive models for target variables and can be automatically constructed from data sets. However, automatic construction is data dependent (i.e., the order of the splits is dependent on the order in which the data is analyzed), and the tree does not allow loop back nor convergence.

VISIRULE

VisiRule is a graphical tool developed by Logic Programming Associates, Ltd. and first released in 2005 that lets you draw decision charts and execute them in situ. The main constructs in VisiRule are nodes that represent questions and/or computable functions and expressions that guard the various paths through the network. You can divide your problem domain into multiple files each containing one or more charts. Charts can have continuation nodes to support modularity and scalability. VisiRule generates code in the form of Flex rules (LPA's expert system product) that can be executed, inspected, and exported for embedding in external applications. Charts can be exported as Windows Metafiles (WMF) for usage within other common desktop applications such as Word.

VisiRule is not designed to automatically construct visual models or executable code from data, rather it is a tool that allows experts to build decision models using a graphical paradigm, like MindMap, but one that can be annotated using code and or Boolean logic and then executed and exported to other programs and processes.

People have argued that decision trees and decision tables do offer a more practical way of representing knowledge than text-based rules where the amount of text can obscure the structure and interrelationships inherent in the rules. Indeed, some business rule vendors now offer limited support for these within their systems.

No doubt, decision tables can prove very compact and familiar structures to work with (especially for those of us with an Excel inclination), but in this article, we examine how decision charts offer a richer and more appropriate format. Only by using a graphical paradigm can we hope to retain and re-use the structuring information required to understand, navigate, and maintain complex rule bases.

COMPANY LOAN EXAMPLE

Let's consider the process of whether or not to grant an employee a company loan. We can start with a

Table 1. Simple set of rules.

If Full time = yes	and Over4yrs = yes	then answer = Grant Loan
If Full time = yes	and Over4yrs = no	then answer = Unclear
If Full time = no	and Over4yrs = yes	then answer = Unclear
If Full time = no	and Over4yrs = no	then answer = No Loan

Table 2. Initial decision table.

Full time	yes	yes	no	no
Over4yrs	yes	no	yes	no
	Grant Loan	Unclear	Unclear	No Loan

0278-6648/07/$25.00 © 2007 IEEE

simple set of rules as shown in Table 1. We can construct a simple decision table, as shown in Table 2 where each question is a row, and each column holds the various answers. You can read down any specific column to find the solution for that combination of answers. You can write the tables transposed so that the rows contain the answers for a given case with the outcome contained in the final column. Decision trees like decision tables can be drawn either way.

We have two questions, each of which has two possible answers, therefore four possible combinations. The questions, as they stand, are independent and can be asked in either order.

Using VisiRule, we can draw a simple chart, Fig. 1, where each expression box evaluates a compound logic expression referencing the two previously asked questions.

We could, however, structure the expression boxes as in Fig. 2. The logic in the boxes is simpler, but we have more boxes; i.e., we have pushed some of the complexity in the logical expressions out into the structure of the diagram. This seems a fair trade.

Now, let's modify this example to ask different questions, depending on the first answer (as is often the case in expert systems). This imposes an ordering of the questions as shown in Table 3.

Now, we have three questions, although only two are ever asked (i.e., you never ask both the three and five years questions).

We can add another row to the decision table, Table 4. A hyphen indicates questions that are not appropriate and will not be asked in that column.

Figure 1 is no longer appropriate, as we would have to ask all three questions up front and then test the logic. However, by introducing the secondary questions after the initial branching, we can convert the second chart in Fig. 2 into a third chart as in Fig. 3.

Now, the answer to the first question determines which branch you go down and the next question will be asked. We can see the branching and the relative structuring of the questions in the chart.

Now, let's merge the two Unclear solutions into a single track, call it Check_previous, and then continue it. This frequently occurs; two positive responses produce an approval, two negatives, a rejection, but mixed answers need further exploration.

We can extend our rules as follows in Table 5 and then define some continuation rules for the intermediate conclusion, Check_previous as shown in Table 6.

Again, we can extend the table to include the additional question to get Table 7.

Note there is some duplication in the table now related to Previous_loan. This is usually bad, as it makes maintenance and updates difficult. To avoid that duplication, we need to use an intermediate conclusion with an associated nested table. But then we would have two representations to think about, first, the decision tables and second, the relationship between tables.

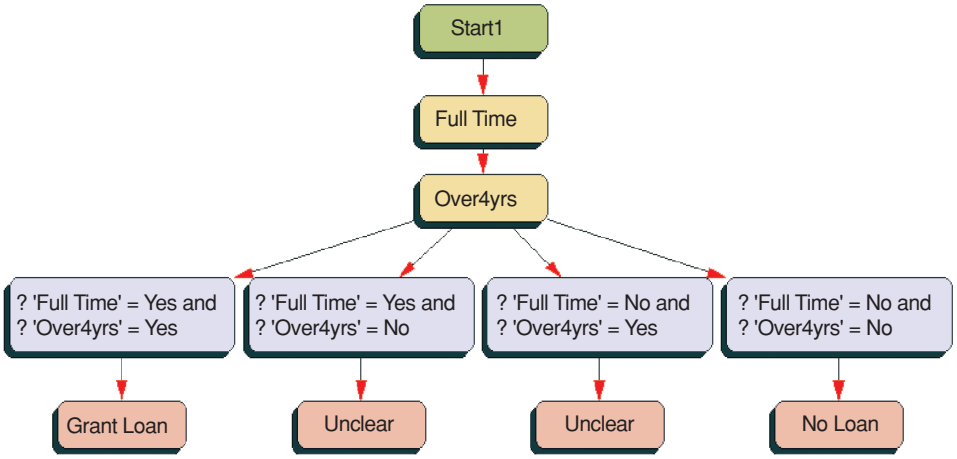


Fig. 1 Initial chart with compound logic expressions

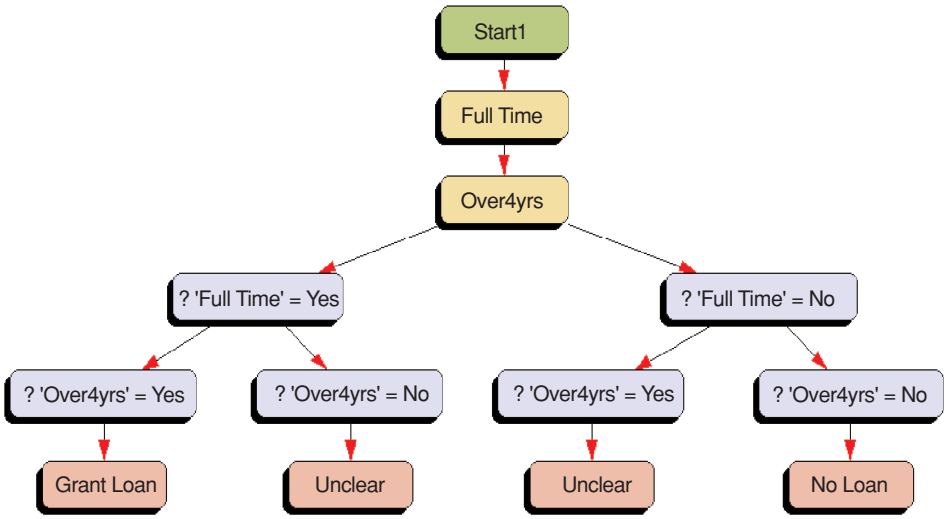


Fig. 2 Initial chart using structured expression boxes with simple logic

Table 3. Modified rules with split questions.

If Full time = yes	and Over3yrs = yes	then answer = Grant Loan
If Full time = yes	and Over3yrs = no	then answer = Unclear
If Full time = no	and Over5yrs = yes	then answer = Unclear
If Full time = no	and Over5yrs = no	then answer = No Loan

Table 4. Extended decision table with split questions.

Full time	yes	yes	no	no
Over3yrs	yes	no	—	—
Over5yrs	—	—	yes	no
	Grant Loan	Unclear	Unclear	No Loan

In our chart, we can simply join the branches represented by the two Unclear nodes in the third chart in Fig. 3 and link that merged node into our new question, Previous_loan, as in Fig. 4.

Again, we can see the merging of the branches and subsequent branching graphically.

Finally, let's resolve the remaining Unclear solution. We continue the "yes" branch to Previous_loan and leave the "no" branch to still give, No Loan. Replacing Unclear with Previous_loan results in a slight change to the rules previously shown in Table 6 to give a new set of rules as shown in Table 8.

The continuation rules for the intermediate conclusion, Check_repaid, are simple as shown in Table 9.

Table 5. Merged rules referencing previous loan.

If Full time = yes	and Over3yrs = yes	then answer = Grant Loan
If Full time = yes	and Over3yrs = no	then answer = Check_previous
If Full time = no	and Over5yrs = yes	then answer = Check_previous
If Full time = no	and Over5yrs = no	then answer = No Loan

Table 6. New rules relating to previous loan.

If Check_previous and Previous_loan = yes	then answer = Unclear
If Check_previous and Previous_loan = no	then answer = No Loan

Table 7. Question about previous loan added to decision table.

Full time	yes	yes	yes	no	no	no
Over3yrs	yes	no	no	—	—	—
Over5yrs	—	—	—	yes	yes	no
Previous_loan	—	yes	no	yes	no	—
	Grant Loan	Unclear	No Loan	Unclear	No Loan	No Loan

Table 8. Modified rules relating to previous loan.

If Check_previous	and Previous_loan = yes	then answer = Check_repaid
If Check_previous	and Previous_loan = no	then answer = No Loan

Table 9. New rules relating to loan repayment.

If Check_repaid	and Repaid_on_time = yes	then answer = Grant Loan
If Check_repaid	and Repaid_on_time = no	then answer = No Loan

Table 10. Question about loan repayment added to decision table.

Full time	yes	yes	yes	yes	no	no	no	no
Over3yrs	yes	no	no	no	—	—	—	—
Over5yrs	—	—	—	—	yes	yes	yes	no
Previous_loan	—	yes	yes	no	yes	yes	no	—
Repaid_on_time	–	yes	no	—	yes	no	—	—
	Grant Loan	Grant Loan	No Loan	No Loan	Grant Loan	No Loan	No Loan	No Loan

Now, we have only two outcomes and five binary questions; users will be asked two, three, or four depending on their responses

The corresponding decision table with the new question, Repaid_on_time, is shown in Table 10.

The bottom halves of columns 2-3-4 are the same as the bottom halves of columns 5-6-7. This duplication is not good, but to avoid it, we would need to use nested tables. This corresponds to using intermediate conclusions and subsidiary rules and again introduces another layer of structuring.

By linking the "no" answer to Previous_loan back to the box marked "No Loan" and continuing on with the "yes" branch through to Repaid_on_time, we arrive at a fifth chart as shown in Fig. 5.

The intermediate conclusion, Check_previous, corresponds to the node marked "true" in the chart and would also be the name of the nested table were we to write one.

Structuring

Simple, unstructured tables and rules store answers and conclusions and expand linearly in size as the number of questions grow. To avoid duplication of parts of rules or tables, we need to introduce intermediate conclusions. This, in turn, leads to structured rules or nested tables.

It is this imposed structuring that makes large text-based rule bases hard to understand and maintain. Nested decision tables offer a slight improvement, but they introduce a secondary representation and require additional management facilities.

Some meta-level structuring or layering is required to support large models, but VisiRule uses it to support modularity, rather than to avoid duplicated data. Charts can be viewed as modules and reused within different contexts.

When each combination of answers leads to a unique outcome, the decision table is very appropriate. When there are many questions and answers but only a few outcomes, i.e., a high degree of convergence, the charts are far more accessible.

Sparseness

While tables start as quite compact, each additional question requires an extra row even if the question is only ever asked in one place. This can lead to tables getting quite sparse.

These issues become more obvious once we move away from simple binary questions toward single selections from large lists of items and multiple selections of even quite small lists (like "which of the following foods do you like ?").

Questions and additional information

In VisiRule, we can display not only the name of the question but also both the question prompt and/or any explanation associated with that question. In some industries, the explanation is as important as the question. This contributes a far more rounded and holistic view of the decision-making process we are modeling.

VisiRule provides statement boxes backed up by Prolog-based predicate and Boolean logic to represent functions that can be evaluated. These can be used to either look up or compute answers to questions and/or combine previous answers to produce true or false responses. VisiRule also offers general purpose code boxes to facilitate customization and potentially complex executable procedures.

In some ways, text-based rules give a flat, one-dimensional view of knowledge, decision tables give a 1 1/2-dimensional view and charts a truly two-dimensional view.

Scalability

We can use multiple linked charts within single files to achieve a high degree of modularity. In addition, we can distribute the logic across multiple files to aid the shared development process and enable the concept of common logical subcharts. Individual charts can be developed and tested in isolation and integrated within a larger chart downstream in a manner not dissimilar to components.

Redundancy and duplication

The ability to draw cross-over lines means that we can explicitly represent convergent paths, rather than rely on the human eye to notice that two columns in a decision table point to the result or have the same continuation table. The ability to merge convergent paths both reduces the amount of duplicated data and overcomes the need for continuation trees or tables.

Consider the following example from *Ruling the Business: About Business Rules, Decision Tables and Intelligent Agents* with rules used by a personnel department to establish holiday entitlement which states:

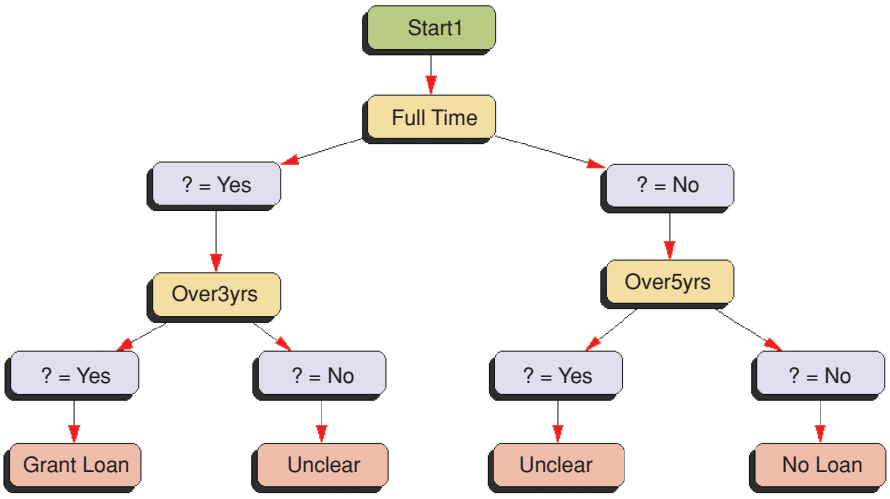


Fig. 3 Extended chart with split questions

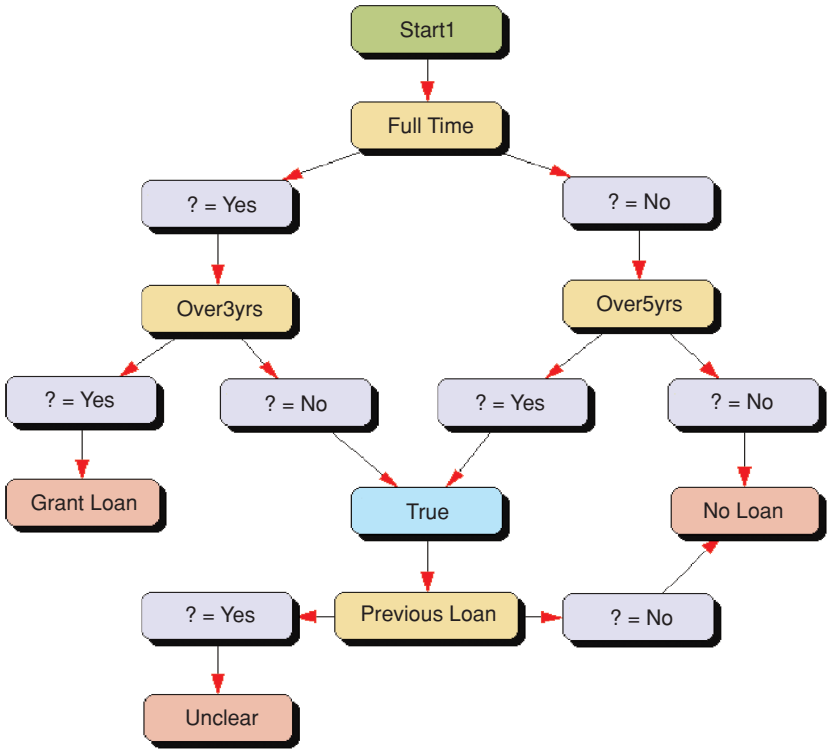


Fig. 4 Merged paths lead to previous loan

"The number of holidays depends on age and years of service. Every employee receives at least 22 days.

"Additional days are provided according to the following criteria: Only employees who are younger than 18 or at least 60 years old, or employees with at least 30 years of service will receive five extra days.

"If the employee has at least 15 but less than 30 years of service, two extra days are given. These two days are also provided for employees who are 45 or older. The two extra days cannot be combined with the five extra days.

"Employees with at least 30 years of service and also employees aged 60 or more, receive three extra days, on top of the possible additional days already supplied."

The initial decision table looks something like Fig. 6. But this can be contracted to Fig. 7 (as explained by Vanthienen).

Notice that there are two questions that expect a number as input, and the branching is done according to "bands" of values ranges.

In VisiRule, we can produce something like the sixth chart, which appears in Fig. 8.

As is often the case in knowledge-based systems, while there are many conditions and tests, there are not that many unique outcomes.

We can use merging, in the case of 22+5+3 being arrived at via multiple routes, and use complex logic (mixture of OR and AND) to get to the 22 + 2 solution.

On the assumption that we are using rows to hold the questions in the decision tables, then OR corresponds to multiple columns in a table with the same outcome (i.e., there are multiple ways of achieving this).

Finally, we consider a lawn diagnostic problem, which is shown in a seventh chart in Fig. 9.

The decision table for this would contain nine rows (nine questions), ten columns, a longest path of seven question/answers, a shortest path of just two questions/answers even though it only has four distinct end nodes (outcomes). The chart is compact, fits onto a single sheet of paper, can be projected onto a wall for group discussion purposes, and yet is still opaque.

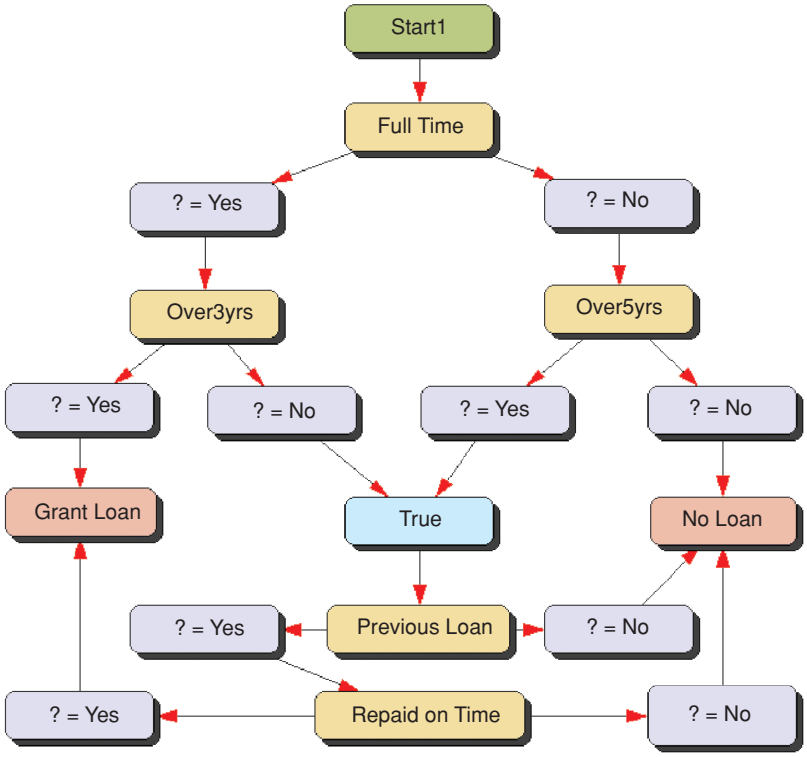


Fig. 5 Merged paths for repaid loans

Import and Export

Currently, VisiRule requires the chart to be drawn manually, though it does provide many editing and layout tools. The chart is represented internally as a collection of Prolog data objects that can be used to generate either executable Prolog or Flex code. Flex is a hybrid expert system toolkit that LPA sells. VisiRule can export an XML representation of its charts that can then be used in conjunction with other rule-based systems and applications.There are also plans to support the importation of such data along with the provision of automatic layout tools.

Benefits

Decision charts are more compact and intuitive than decision tables or rules. They are easier to understand and discuss especially within large groups of people. This makes them less prone to error and easier to manage and maintain.

They are compact and occupy a small amount of space in memory and on disk.

There is no performance degradation as it is possible to generate efficient code from a decision chart. Current benchmarks show execution of large charts on a par with large decision tables. It is too early to have empirical evidence to confirm this.

CONCLUSIONS

We believe that decision charts provide an excellent medium for storing and communicating the knowledge associated with making decisions. They help overcome many of the problems associated with text-based rules and decision tables. Programming-oriented

1. Age	<18			18-<45			45-<60			>=60		
2. Service	<15	15-<30	>=30	<15	15-<30	>=30	<15	15-<30	>=30	<15	15-<30	>=30
1. Assign 22 Days	.	.	.	.	.	.	.	.	.	.	.	.
2. 5 Extra Days	.	.	.	.	.	.	.	.	.	.	.	.
3. 2 Extra Days	.	.	.	.	.	.	.	.	.	.	.	.
4. 3 Extra Days	.	.	.	.	.	.	.	.	.	.	.	.
	1	2	3	4	5	6	7	8	9	10	11	12

Fig. 6 A first view of the decision table

engineers are not required to construct and maintain them. They can be accessed and used by a wide range of people directly involved with the business processes.

VisiRule provides a drawing environment that lets you draw decision charts, which can be immediately executed and verified, or exported as graphical objects or as program text for embedding within larger computer processes and/or applications.

READ MORE ABOUT IT

• J. Vanthienen, "Ruling the business: About business rules, decision tables and intelligent agents," in *New Directions in Software Engineering*, J. Vandenbulcke and M. Snoeck, Eds. Leuven, Belgium: Leuven Univ. Press, 2001, pp. 103–120, 160.

ABOUT THE AUTHOR

Clive Spenser completed his M.Sc. at Imperial College, London, in the area of advanced information systems. He has worked for LPA for more than 20 years as a researcher, project engineer, and in product development, sales, and marketing departments. He also advises research centers on the commercial exploitation of innovative research.

1. Age	<18	18-<45			45-<60		>=60
2. Service	-	<15	15-<30	>=30	<15 or 15-<30	>=30	-
1. Assign 22 Days	X	X	X	X	X	X	X
2. 5 Extra Days	X	.	.	X	.	X	X
3. 2 Extra Days	.	.	X	.	X	.	.
4. 3 Extra Days	.	.	.	X	.	X	X
	1	2	3	4	5	6	7

Fig. 7 The final decision table

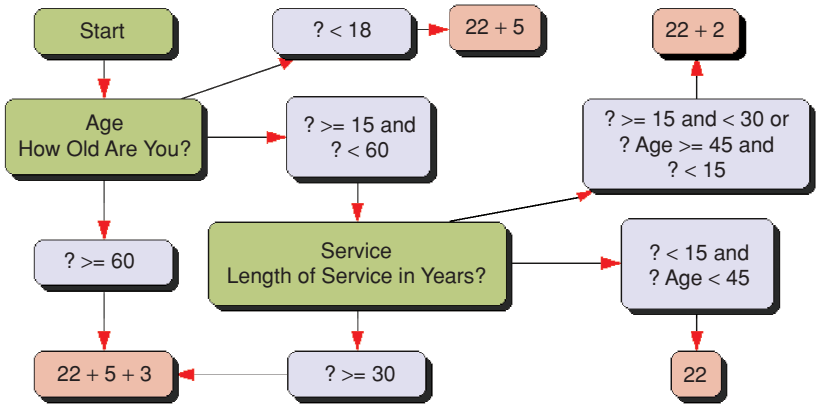


Fig. 8 Chart for holiday entitlement

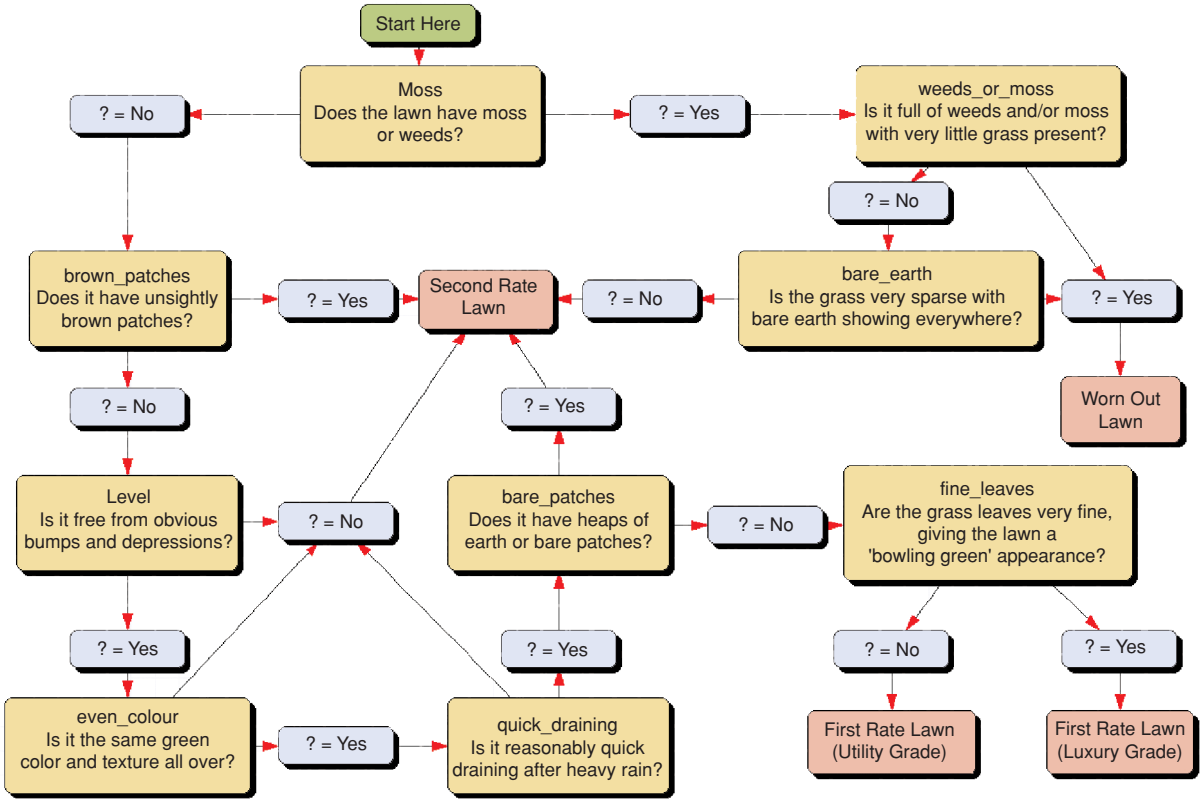


Fig. 9 Chart for diagnosing lawn problems