

The Visual Development of Rule-Based Systems

By Charles Langley and Clive Spenser

Introduction

In the late 1980's Knowledge Based Systems (KBS) were seen to be leading edge software technology. Developers thought that the simplest KBS paradigm, Expert Systems, perhaps combined with probabilistic and fuzzy logic extensions would soon revolutionise the way that software was used throughout business and other sectors of the economy.

KBS software was built on rules which encoded the knowledge of experts in any given domain. Computers would then use this encoded knowledge to make decisions on behalf of their human users.

It was not long however, before the bubble of hype surrounding these systems began to burst. Something was wrong, but what was it?

The Knowledge Acquisition Bottleneck

Apart from the limited power of the computers available at the time, the major problem was the difficulty of acquiring implicit knowledge from the minds of experts and then representing it explicitly. This so-called Knowledge Acquisition Bottleneck was believed to be the limiting factor on building systems that could do complex, useful tasks.

By the end of the twentieth century however, university departments were working hard at this problem. Curiously it was often Psychology departments rather than Computer Science departments which had the most impact in this area.

In particular, Ethnography (by then seen as a core part of Cognitive Psychology) was being used to study behaviour in situ with the aim of identifying the cognitive processes underlying that behaviour. Just as Margaret Mead

(an early ethnographer) had lived amongst native tribes in Papua New Guinea in order to study their cognitive behaviour, so Psychology departments were sending researchers (often under cover) into workplace environments to discover how people approached problem-solving activities.

This work was, and continues to be, very successful. Knowledge acquisition is no longer the 'black art' it was deemed to be. Despite this, KBS has continued to be underused. Why might this be?

A Knowledge Representation Bottleneck?

It is my contention that the problem was not primarily with how we obtained knowledge, but with how we represented it. I am not arguing that rules (or Bayesian networks and other knowledge representation methods) are inadequate to the

Order the new Volume 17
CD at www.pcai.com/store

task, but rather that it is the way in which these rules and other representational formalisms are themselves represented that is the limiting factor.

At first a simple rule-base is relatively transparent, especially if properly documented. Certainly such systems were easier to comprehend than procedural code and were subsequently easier to update and amend. As such rule bases became larger and more complex however, a simple syntax error, perhaps only involving one word, could prevent them from operating correctly. The complexity of these rulesets also meant that it was difficult to get an overview of what was intended, thus impeding their maintenance and extension.

A Picture is Worth a Thousand Words

The problem of rulebase comprehensibility, I would argue, is the fact that we have primarily represented knowledge using text based structures rather than visual ones. No matter how close to natural language a knowledge representation language is, you cannot see at a glance what a complex system is trying to do.

Visual Rule Generation

Rule generation via a graphical interface is a hot topic right now, with offerings from a number of companies small and large. This is being driven in part by current interest in so-called 'business rules management' which is arguably a reawakening of the KBS paradigm we mentioned earlier.

London based Logic Programming Associates is an appropriate company to enter this market as it has been producing rule-based software since the mid 1980s. Its latest product, VisiRule, enables rule-based systems to be automatically generated from a flowchart drawn on the screen.

Consider the following business rule (Ross 2003):

Rule: An order must be credit-checked if any of the following is true:

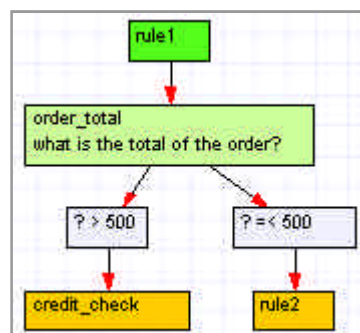


Figure 1

- * The order total is more than \$500
- * The outstanding balance of the customer's account plus the order amount is more than \$600
- * The customer's account is not older than 30 days
- * The customer's account is inactive
- * The customer is out of state

Build more powerful,
scalable applications
with
Allegro CL
and **Lisp**

Webify Seamlessly

Deploy applications over the
internet with AllegroServe and
other web/app servers

Easy Connectivity

Integrate applications with a
variety of Java, XML, database,
CORBA and COM/OLE tools

Get to market *on time* and
on budget with
Allegro CL and Lisp.

Find out why so many
leading companies & top
researchers won't program
with anything else.

Free trial download!

www.franz.com

AllegroServe is a trademark, & Allegro CL is a registered trademark of Franz Inc. Copyright 2002 Franz Inc. All rights reserved.

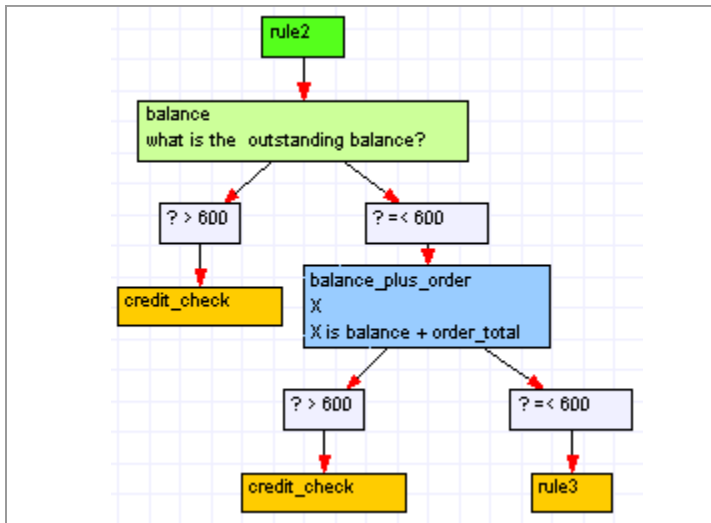


Figure 2

This rule can be decomposed into five separate rules, one for each of the bullet points. This is how the first of these rules can be constructed using VisiRule:

Start Boxes, Questions, Expressions and Continuation Boxes

Figure 1 consists of different coloured boxes connected by arrows. The top box is a Start Box, used to start a chart or sub-chart. The second box is a Question Box with a name of a variable (order_total) and a prompt (what is the total of the order?). This one is a Number Input type Question Box. Other types available are Single Choice, Multiple Choice, Integer Input and Set Input, each having a separate colour.

Question Boxes can also optionally contain an explanation to provide explanatory text if the user presses an Explain button.

The two white boxes in figure 1 are called Expression Boxes. These boxes determine the direction of flow of control in the chart. If the user answers 600 to the question above, control passes to the credit_check Continuation Box. If the user enters 200, control passes to the rule2 Continuation Box.

Continuation Boxes provide modularity since they direct the flow of control to a Start Box of another chart or sub-chart with the same name.

Calculations Using Statement Boxes

The chart shown in figure 2 introduces another type of box, the Statement Box. Here we use one to calculate the total of the

order and the outstanding balance which is then tested by Expression Boxes just as with a Question Box. We now have the first two rules.

The final three rules are produced in the same way (Figure 3).

Finally we build a small chart representing the credit check. This introduces two new types of box. The second box down in figure 4 is a Code Box. Code Boxes can contain any Prolog code. In this case, the Code Box is used to display a message box with advice for the user. More sophisticated multi-line Code Boxes can be used

for date handling, invoking external programs and loading spreadsheets.

The other new box type in figure 4 is the End Box. This terminates the

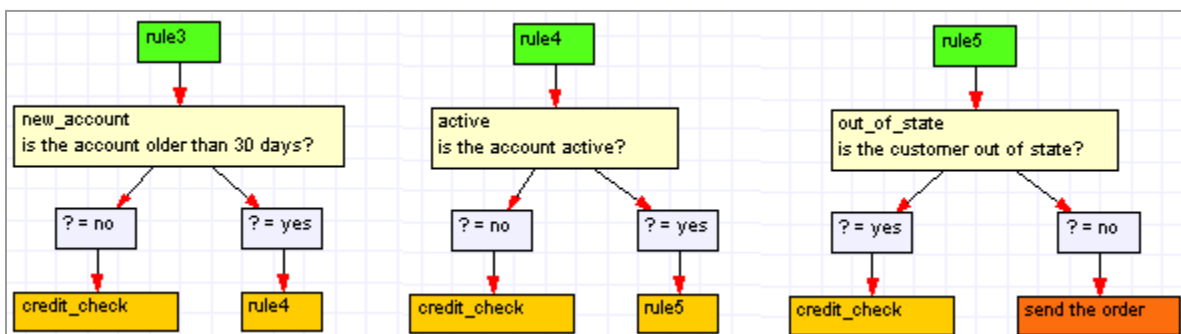


Figure 3

...Thinking Software

LPA's integrated suite of advanced software tools enables you to build intelligent applications both rapidly and safely.

LPA products feature:

- Robust and reliable run-time performance
- Support for DLLs, DDE, OLE, ODBC, TCP/IP, HTML standards
- Graphical tools and debugging aids
- Choice of delivery environment (VB, Java, Web, Delphi)

Modules include:

- LPA Prolog for Windows - leading Prolog compiler system
- Flex - popular hybrid expert system toolkit
- Agent - distributed agent toolkit
- Flint - fuzzy and probabilistic reasoning
- VisiRule - graphical expert systems generator

Logic Programming Associates Ltd

www.lpa.co.uk info@lpa.co.uk

Phone: + 44 (0)20 8871 2016

Fax: + 44 (0)20 8874 0449

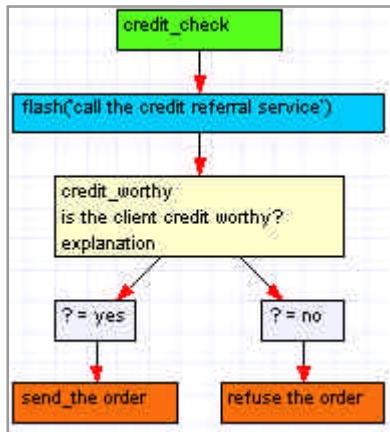


Figure 4

program after first displaying a Windows type message box containing the text "send the order" or "refuse the order".

All in One

Although we chose to represent our five rules as five separate charts, we could have put them all together in one chart as in figure 5. There is no limit (apart from computer memory) to the size of a flowchart, and a zoom control enables you to see as much of it on one screen as you wish.

Simplifying More Complex Rulesets by Using Statement Boxes

So far we have a very simple rule network. Let us make it a little more complex by modifying rule1 to include the notion of different customer types, each of which has its own order limit:

Rule1: An order must be credit-checked if any of the following is true:

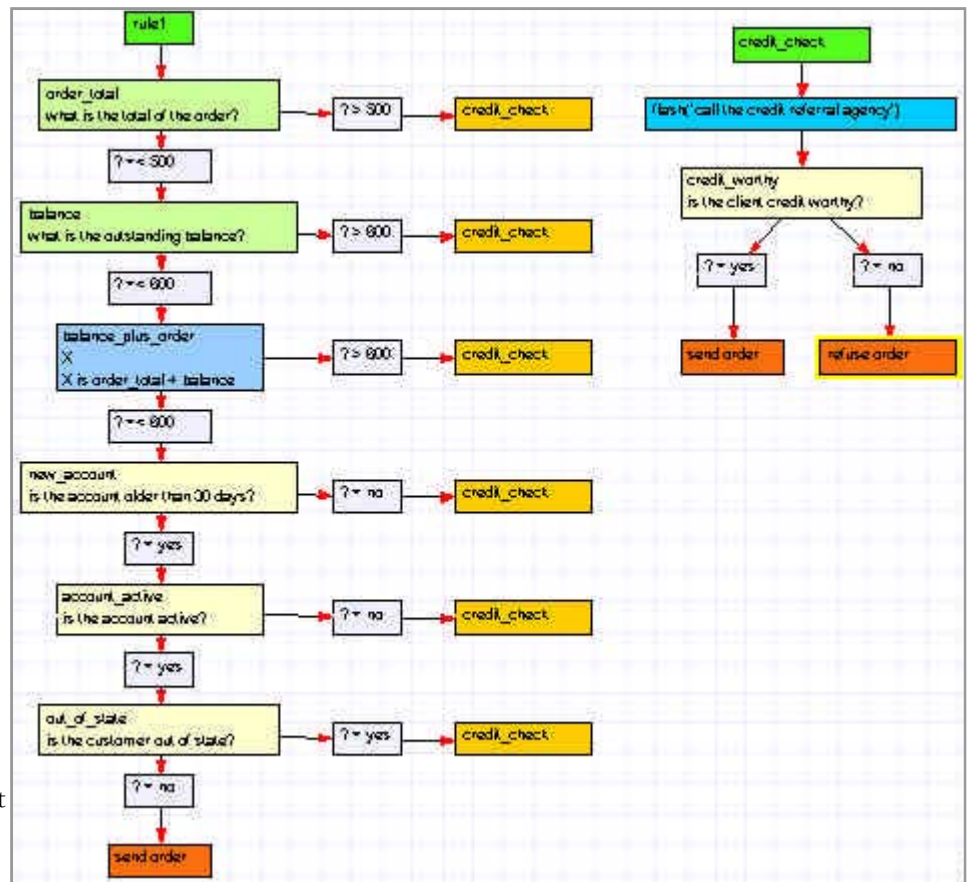


Figure 5

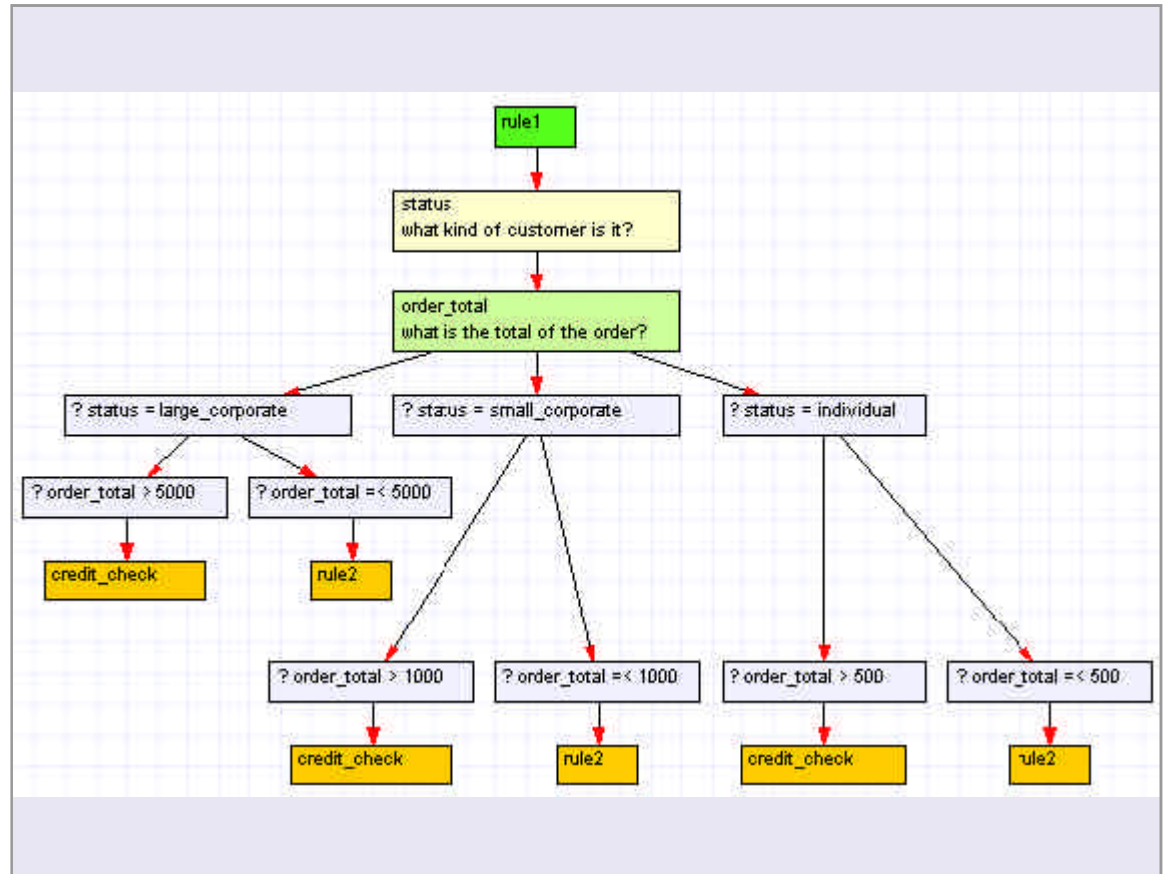


Figure 6

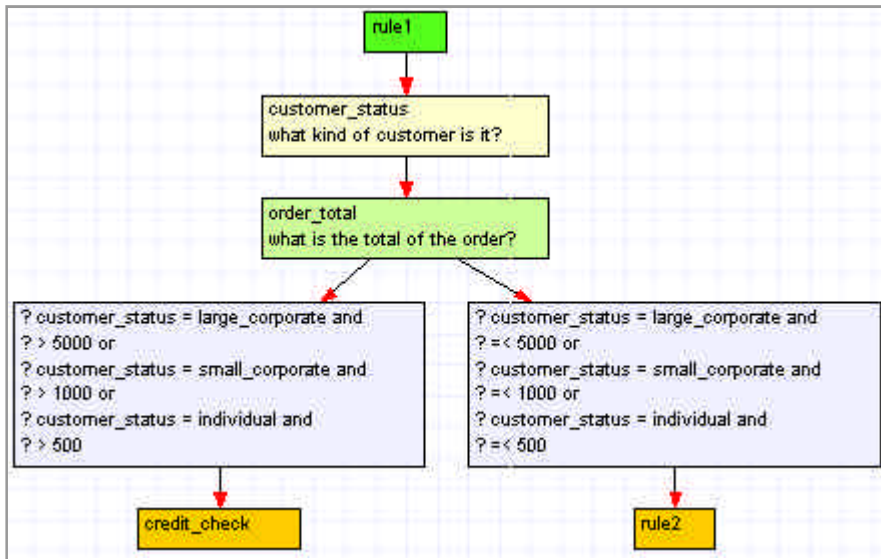


Figure 7


Visit the PC AI Store at
www.pcai.com/store

- * Customer status is large corporate and the order total is more than \$5000
- * Customer status is small corporate and the order total is more than \$1000
- * Customer status is individual and the order total is more than \$500

We could do this using Expression boxes as in Figure 6.

One problem with this representation is that it is far from compact. We can improve on this by using more complex expressions as in the chart in Figure 7. This however, is not ideal either, as the construct order limit is not represented explicitly and therefore cannot be referenced elsewhere in the chart. This leads to unnecessary duplication. Such duplication reduces the modularity of the chart and makes the order limit for each customer type harder to update or revise. To represent order limit we can use a Statement box as in figure 8. This

Visual Prolog




Good News for Programmers

We gave ourselves the challenge to rethink the object paradigm, with the purpose of creating a simple and clear, but yet extremely powerful object system for Visual Prolog.

The result is a very powerful and safe programming language, which combines some of the best features of many programming paradigms in a consistent and elegant way.

Free for Personal, Education or Research use

[click here](http://www.visual-prolog.com)



www.visual-prolog.com pdc@pdc.dk

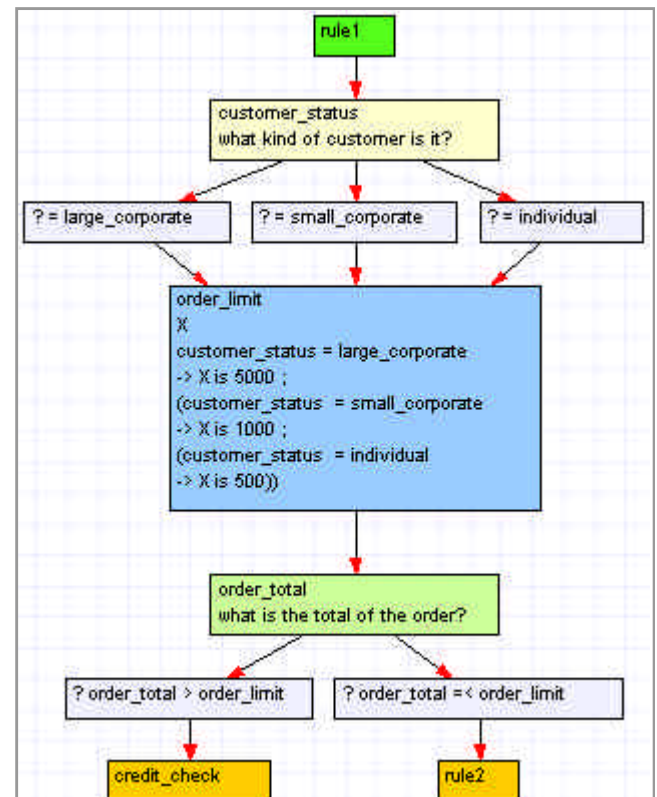


Figure 8



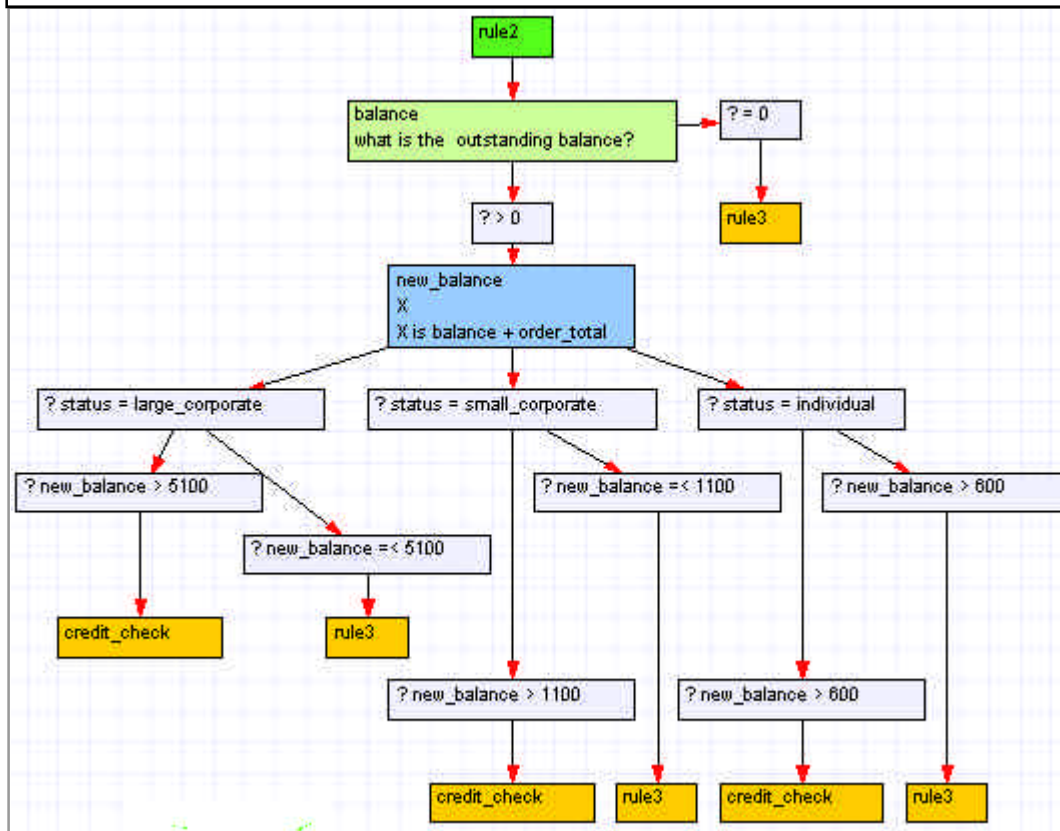
For Over 22 Years - The No. 1 Proven, Selected Expert System Software and Knowledge Automation Services for Businesses, Government and Universities Worldwide



www.exsys.com +1.505.888.9494

Low-Cost, Highly Successful Pilot Project Packages
Professional Assistance in Design - Development - Delivery

Download Free 30-Day Software Evaluation.



combines the compactness of the previous chart with an explicit order limit which we can refer to later. For example, suppose we modified rule2 in the following way:

Rule2: An order must be credit-checked if the following is true:

- * The outstanding balance of the customer's account plus the order amount is more than \$100 over the customer's order limit.

Figure 9 shows the simple version using expression boxes while figure 10

Figure 9

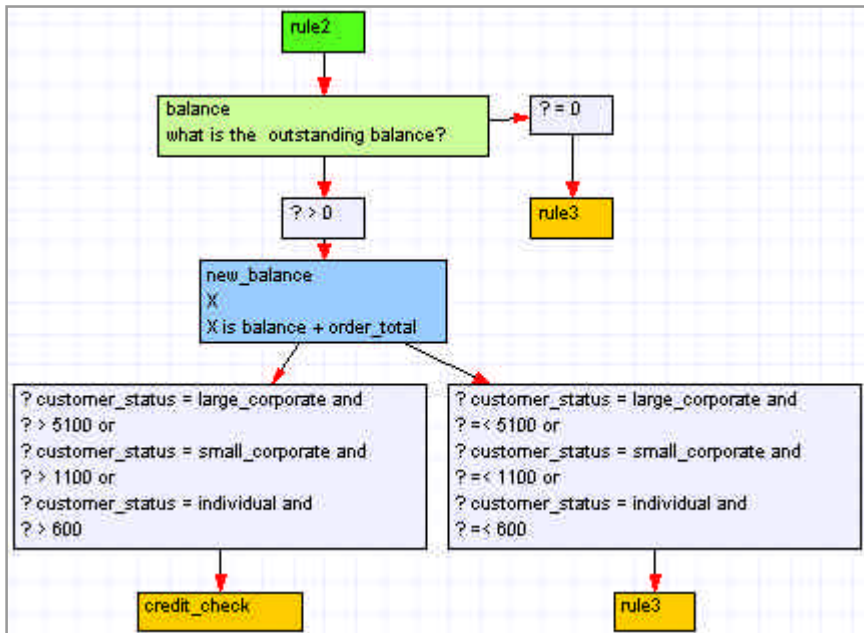


Figure 10

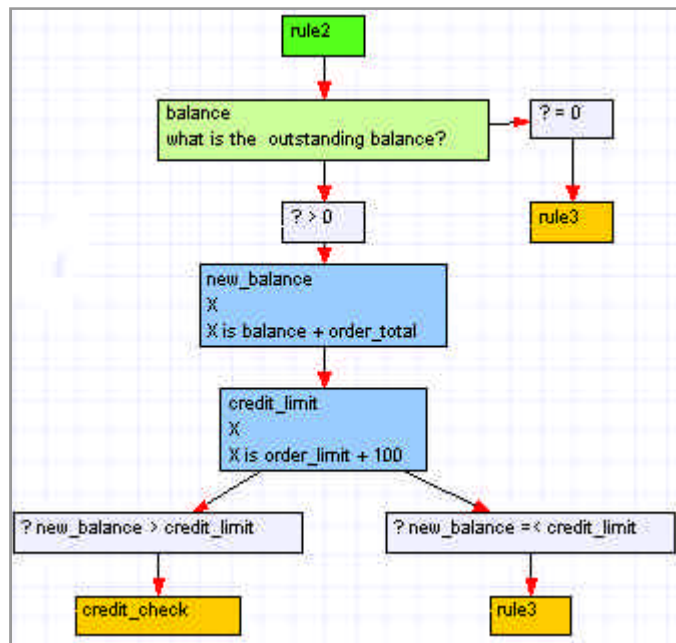


Figure 11

shows the more compact version using expression boxes.

Figure 11 illustrates the version which uses a Statement box. It is because we can reference `order_limit` that this is so much simpler than the flowcharts shown in figures 9 and 10. Furthermore, the credit limit updates itself automatically if we revise the order limits in rule1.

A Statement box is needed not just to perform calculations

also to reference them for testing if the questions appear in a separate chart. Figure 12 shows this for a revised ruleset which recasts rule5 as:

Rule5: An order must be credit-checked if the following is true:

- * The order is out of state and the customer status is not large corporate

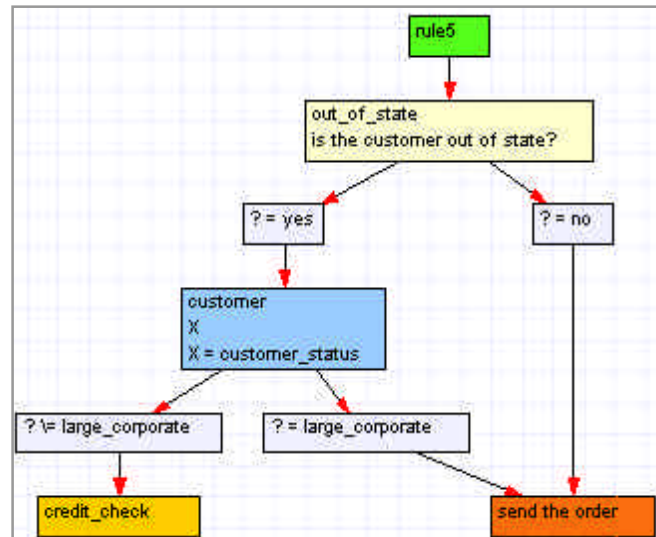


Figure 12

on the answers to previously asked questions, but

Visit the PC AI Store to
order Volume 16 and 17 CDs

Issue Topics Include:

- *Intelligent Applications
- *Intelligent Web Applications and Agents
- *Neural Networks
- *Expert Systems and Knowledge Representation
- *Data Analysis and Mining, Modeling and Simulation
- *Business Applications

Visit the PC AI store today:

www.pcai.com/store

Our modifications to Ross' original ruleset can be represented textually as follows:

Rule: An order must be credit-checked if any of the following is true:

- * Customer status is large corporate and the order total is more than \$5000
- * Customer status is small corporate and the order total is more than \$1000
- * Customer status is individual and the order total is more than \$500

- * The outstanding balance of the customer's account plus the order amount is more than \$100 over the customer's order limit.
- * The customer's account is not older than 30 days
- * The customer's account is inactive
- * The order is out of state and the customer status is not large corporate

This is a disjunction of eight separate sub-rules rather than a network of five rules as in the VisiRule representation (figure 13).

The reason that the VisiRule version requires less rules than the textual representation is that the first four clauses above are represented as a single disjunctive rule.

Knowledge Sharing

One great advantage of VisiRule charts is that they enable knowledge sharing and collaborative development of rule-based systems by non-technical personnel.

People intuitively understand

visual representations more easily than complex textual representations as they can 'see at a glance' what is going on. This is why Powerpoint presentations are so ubiquitous in the business world. Tools such as VisiRule take this one step further in that they enable the creation of executable visual representations.

This is different from paradigms such as Visual Basic which enable the visual presentation of predominantly text based dialogues. With VisiRule the user creates graphical machines, the appearance of which shows how they structure and encode knowledge.

The emphasis on visual representation as opposed to textual representation provides three major advantages since such graphical constructs are:

- * fast to construct
- * intuitive
- * safe (because the system verifies them in the background)

Whereas textually based constructs are:

- * slow
- * unobvious
- * error prone

Much like a PowerPoint slide show that can be executed, VisiRule's graphical emphasis makes the knowledge accessible to both technical and non-technical personnel. VisiRule generates Flex code from the flowcharts and compiles it, allowing the user to test the system as it is being constructed.

Language Independence

It doesn't matter whether the final rule-based system is to be ultimately implemented in Flex or Prolog. VisiRule can be used as a rapid prototyping development environment in which non-technical managers can communicate what they want to their technical staff who can then implement it in any language.

In fact, LPA intends to equip future versions of the tool with a code generation utility which would enable the user to select Java, Visual Basic or C++ as the executable code to be generated.



Charles Langley
can be reached at
chaslangley@ad.com

Clive Spenser can
be reached at
dive@lpa.co.uk

References

Ross 2003, Ronald G Ross, 'Principles of the Business Rules Approach', Addison Wesley, ISBN 0-201-78893-4, page 139.

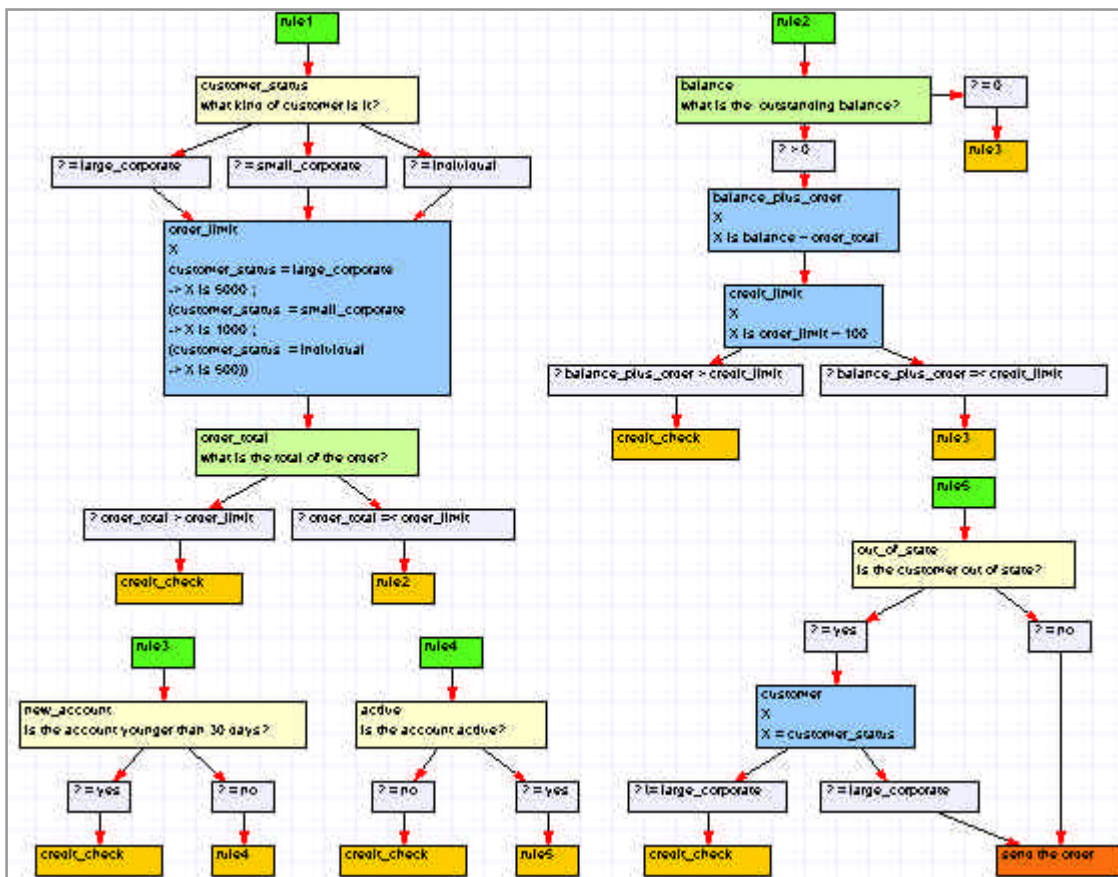


Figure 13