Patterns and the capture of business rules

Most of us recognize people based on their facial characteristics, even when we have met the person only a couple of times or there has been a long period of absence. (I am probably the exception, thus belonging to this minority that refutes this general rule — hence I've developed some other strategies to cope with this disability.)

What makes this ability such a powerful feature of humans is that we do not need an exact description of something to be accurate in assessing it.

Right now I see a 'Eurasian Jay' (in Dutch: *vlaamse gaai*) in my city view … haven't seen one for many years … learned to recognize it when I was 8 years old. Amazing, isn't it!?!

Despite not having an exact and precise description of a tulip, it's easy for us to recognize a tulip after seeing just one example. You will recognize the next tulip even when the leaves and colors are different. Pigeons are even better; they have the least trouble recognizing rotated images of all species in the world (that could be experimented with).

Based on this introduction you have probably already guessed that the theme of this article is *recognizing a pattern*.

When you work on business rules or business requirements it is good to get into the habit of

recording the same kind of knowledge using the same kind of pattern. There are several reasons why this is a good practice:

- It allows the reader to focus on the substantial differences (instead of the differences in writing style).
- It allows the author to focus on the content (instead of the sentence structure).
- It supports a team of different rule authors to deliver a consistent set of rules.
- A pattern description also helps novice authors to focus and learn from the experiences of others.

Pattern descriptions may be exact, allow for some variations, or allow for many variations not described by the pattern.

In the area of business rules we find them all:

- a strict pattern, like a decision table
- a pattern allowing some variations, like a rule syntax for a programming language
- a loose description allowing many variations, like RuleSpeak

Below are some example business rules:

• The price of a one-way ticket must be calculated as travel distance * unit price.
• A two-way ticket's price must be calculated as travel distance * unit price * 0.9.
• A discount of 10% is only applicable to a price when the ticket is a two-way ticket and the travel distance is more than 100 miles.

These are all valid SBVR and RuleSpeak sentences when you have defined your terminology and fact types correctly. But did you note the variety that is allowed in expressing similar logic? Do we want to allow for such variations? No, such variation

- does not help focusing on differences in semantics (but on differences in style).
- makes it hard to check if all rules are consistent.
- is typically written by a team of different rule authors, each having his/her own writing style.
- does not help novice authors to quickly adopt best practices.

Prescriptive pattern descriptions are an important topic of computer science and the study of

computer languages, but little research has been performed on how to support patterns for an un-natural natural language (or controlled language, like SBVR or RuleSpeak) — a language that should be useful for humans (rather than systems). In the remainder of this column, I discuss my experience with three different approaches to supporting humans with prescriptive sentence patterns.

> *Approach 1: a complete syntax describing all valid patterns*
>
> *Approach 2: valid patterns that allow for fill-in-the-blanks*
>
> *Approach 3: free sentence formulation and pattern matching*

My experience so far has been that humans are so incredibly creative and quick that a complete syntax like what we require of computer languages only slows us down. It slows us down for two reasons:

- It takes a lot of time to develop all possible patterns (although my compliments to Graham Witt who really did a tremendous effort with great results[1]) and the underlying software support, e.g., type ahead and syntax checker.
- It forces the author to write complete and correct sentences, which distracts him from the real content — the knowledge — that he would like to record.

Try it yourself with one of the newly-developed and experimental SBVR editors.[2] Even for someone experienced like me (being on the SBVR submission team) it takes at least 5 minutes to enter a valid SBVR sentence. So while I encourage this research we just have to admit that for practical purposes we are not yet there.

We also experimented with a fill in the blanks approach. In this approach the user is confronted with a wizard, providing options that help him choose the right pattern. The selected pattern is then presented and the author fills in the subject and conditions. We assessed this approach in an experimental setup with students in business informatics. The one group worked with the

wizard; the other group only got RuleSpeak writing guidelines. A qualitative comparison of the results on the same policy set concluded that the group using only the guidelines performed better.[3]

It turned out that the 'fill in the blanks' approach blocked the ability of the author to reflect about the pattern. The pattern is chosen prematurely — when the knowledge that needs to be written down is not yet crystalized — and the author too often continued with a wrong pattern, resulting in incorrect rules.

The authors who used the free sentence formulation combined with guidelines about the patterns had more freedom and followed a creative and iterative process consisting of the following phases:

- Explore — *What are the details of the knowledge involved?*
- Design — *Write down the knowledge.*
- Test — *Is it complete and written in the right way?*

During the *design* phase, a non-disruptive way to support this process is to show the user which patterns match the written sentence and to explain why a given sentence does not match a particular pattern. This kind of information will stimulate the creative process rather than block it. The *test* phase should be supported by showing similar rules that are based on the same pattern, using the same concepts. This way the user will easily detect

- when he wrote similar rules based on a different pattern.
- for what situations the same kind of pattern is used.

This information may often lead to a new *exploration* phase, complementing the knowledge with more rules or correcting existing rule sentences to make them consistent with other rules. This process is more natural for people and therefore, given the current state of the art in language technology, free sentence formulations and pattern matching is the preferred way at this point in time.[4]

This article was originally published by BRCommunity (link).

[1] Graham Witt, “Mind Your Language — Developing Natural Language Rule Statements” series, *Business Rules Journal*, http://www.brcommunity.com/language.php

[2] Visit

http://rulemotion.com

http://sourceforge.net/projects/vetis/

http://sourceforge.net/projects/sbeaver/?source=recommended

[3] Silvie Spreeuwenberg, Jeroen van Grondelle, Ronald Heller, Gartjan Grijzen, “Using CNL Techniques and Pattern Sentences to Involve Domain Experts,” *Lecture Notes in Computer Science*, Volume 7175 (2012), pp 175-193. Available from:

http://link.springer.com/chapter/10.1007%2F978-3-642-31175-8_10#page-1

[4] A tool that explores this strategy is RuleArts’s RuleXpress

