

Author: Rik Gerrits, TCO at LibRT, Amsterdam, the Netherlands

Title: Business rules, can they be re-used?

Keywords: Business Rules, Business Rules Approach, Re-Use

Introduction

What a silly question, you might say. Of course! Business people do it all the time. And obviously you are right. A business rule can be used in more ways than the original inventor of the rule could ever have imagined. So why ask this question? Fact is that even a junior businessperson is infinitely smarter than a computer. And with computers we are aiming to help the business! So the question is really: if a computer is used to apply business rules the way business people do (as some of the technology vendors claim they do), is it also able to re-use rules, the way business people do? Or: if a computer is used to model the business and its rules, is it also able to re-use these models?

This article will discuss the traditional problems around re-usability in the context of information technology, problems that occur even in abstract models. Everybody can see the value of re-usable business models or -software but reality is that they are very hard to find. Is it difficult or is it just a lot of work? Is it a matter of skills or a matter of time and money? As usual it is mix but I think that the business rules approach has some properties that will improve modeling (and software engineering) with respect to re-usability.

The article was intended to be published in two parts. One part dealing with the re-use of business rules in software and the other part dealing with business rules in models. By the time I finished the first part it became clear that most of the problems arising in the models had been discussed in the first part. I would really like to encourage business oriented people to read the entire article; it is not technical at all. At least it will give you an idea why developers need to rewrite (large parts of) business applications *and models* all the time (although they probably told you a whole different story).

Re-use in the scope of this article

This article deals with the problems that arise when IT persons want to re-use models and/or software for a *different task*. It is relatively easy to re-use a model or software if you are just upgrading a system to a new platform, or new language (although a great deal of the IT budget is spent for exactly this type of projects). The problems that are addressed in this article are issues arising in existing models and software when a new task needs to be modeled or automated. A lot of the functionality that we need for a new application has been developed for earlier applications. At a glance we should have a head start by re-using these existing business models and solutions.

There are obvious non-re-usable parts in existing models and software that include application specific user interfaces and particular data interfaces. There are also obvious re-usable parts in existing models and software that include general user interface and behavior components or calculations. It's the specific business oriented parts of models and software that we need to look at.

Here are a few common reasons for not being able to re-use existing business software:

1. The existing software was written, using a specific technical platform or language that is very hard to transform to the new platform or language. The re-usability of software improves if an organization is homogeneous with respect to development platform + language.
2. The existing software has undesirable side effects. The re-usability of software improves if it is component based.
3. The existing software has prerequisites that cannot be met. Again, component bases development will help developers write re-usable atomic functions.

In this article I will focus on other re-usability issues.

A business case

A company produces various products. Their products have a complex pricing strategy. There are a lot of rules that deal with discounts related to the size of the order but also related to combinations of products. They have developed models and software in the past to calculate the total price of an order. Now they want to create a new service for their customers that enables them to configure their most optimal order. They have all the calculations in their existing application and want to re-use these calculations for the new application. Note: it is not an option to disclose the rules to the customer.

In this business case we have to use the same business rules in both a computation task and an optimization task. I deliberately chose these quite different tasks to be able to emphasize the problems that will occur. After reading this article you will (hopefully) see that these problems will arise even if business rules are to be re-used in more similar tasks. In that case the issues will appear on a lower level.

What if the business logic was implemented with an imperative language?

If the rules are implemented in an imperative, e.g. procedural, language such as COBOL, C, C++ or Java, it will be very hard, if not impossible, to re-use the software for the new task. The main disadvantages of the imperative approach (in the context of re-use) are:

1. The software is designed to calculate the value of the total price; each and every step in that program is subject to that goal. In the new task it is required to reason the other way around.

2. There is an explicit order of processing in imperative software that is very hard to change. If a certain value depends on other values, these other values must be calculated first.

What if the business logic was implemented in a declarative language?

Some developing environments allow business logic to be expressed in a declarative manner. This means that you can write the rules in a random order; the application will take care of the processing order, based on the availability of data. The component that takes care of this processing is often referred to as an Inference Engine. An Inference Engine uses rules and data to derive new data. But unfortunately most Inference Engines are not able to derive *any goal from any data*. Each and every rule, as declarative as it is, has its own distinguished purpose.

Take this computation for example: **Price After Discount = Price Before Discount – Discount**.

Most Inference Engines can use this rule to calculate the **Price After Discount** as soon as **Price Before Discount** and **Discount** are computed, but the rule cannot be used to calculate the **Price Before Discount** if the **Price After Discount** and **Discount** are known. (You will now understand that a junior businessperson is infinitely smarter than a computer).

There are Inference Engines, however, that are able to cope with this kind of complexity. These kinds of engines, often referred to as constraint logic processors or propagators, treat rules as *constraints*. Their goal is to derive new data that comply with the rules. This approach seems to add more to the chances of re-usage than other approaches. There are two common reasons for **not** using this technology: Constraint propagators are ideally used in optimization, configuration, planning and scheduling solutions and are optimized for solving these kinds of problems. For straightforward calculations, this technology is often considered as 'overkill'. The second reason seems to be that it is considered difficult to express all your business logic with constraints. Constraints are intuitive expressions for describing optimization problems but not for premium calculation, for example.

What if the business logic wasn't implemented at all? What if they generated the code using a case tool?

Some of the readers may think: what really matters is the business **model**. I tend to believe that they are right. There are, however, some considerations that need to be addressed.

If the company in our business case uses a case tool to generate code from a model, they may have to create a new model for the new task. It all depends on the ability of the code generation tool to *give the application the desired direction and goal(s)*. Or even better: whether the code generation tool is able to generate code with which you can *derive any goal from any data* as I explained before. The model may contain the expression: **Price After Discount = Price Before Discount – Discount**.

Now, is the to-be-generated application used to calculate **Price After Discount**? In that case the generation process is pretty straightforward. But what if the application needs to calculate the **Discount** based on the other two variables?

So, if the code-generation-tool does not help the user to give the application the right direction, the model cannot be used to generate a new application for a different task. In other words: the code-generation-process needs to be goal-driven.

Please note that it is already assumed that the business expressions in the models are **not directed**. If the modeler was forced to specify the expression in such a way that the rules get an explicit goal or purpose, all problems of software I discussed in previous paragraphs apply to such model. Beware: this problem may be hidden in a methodology! If the modeler has to decide whether a rule is a constraint or a computation or an event-rule or whichever types of rules the methodology distinguishes, he may have to reconsider his decisions when a new application needs to be generated.

Since there are not a lot of modeling tools that are able to generate applications from *rule expressions* anyway, most organizations are not concerned with these problems. The object model, the data definitions and the function signatures can be generated, but the actual business rules need to be implemented manually. Business Rule Models are used as specifications for a developer and since even a junior developer is infinitely smarter than a computer, he/she will figure out the best way to implement a particular business expression. In this regard a model can be re-used time and time again...

So, how to model the business rules in a re-usable way?

To answer this question, we need to look at the business itself. Are business people facing these problems? And if yes, how do they deal with them? If you look at the governmental 'business', for example, you will see that they already separated the business rule definitions from their implementation, or enforcement. The people who invent the rules (legal drafters) are typically not concerned with the enforcement of these rules. So, when they have figured out what the rules must be, they let other people decide how to make sure that the citizens will obey these rules, or how citizens can claim their rights. The law enforcement departments obviously are only able to do their job well, when the rules are precise, consistent and complete. The art of drafting legal texts is already quite a few centuries old so the quality of these rules is usually pretty good, although it may take a lot of time to do it right. And, although legal texts are sometimes hard to read, it's still natural language so the majority of the rules are a perfect means of communication between the two disciplines. Also note that legal drafters should not be concerned with drafting re-usable rules, they can always refer to existing definitions or rules.

The law enforcement is free to make a choice in their implementation, as long as it is compliant with the rules. It is important to note that it is not necessary to change the law when the enforcer decides to change his methods, for example to collect more tax.

I think this is exactly the way it should work with business rules as well, as far as the computer aided management and execution are concerned. The business should be able to design their business independently from the IT department in such a way that

1. They don't have to bother about the actual implementation of their rules.
2. They don't have to focus on a single application of the rules.

The IT people, in return, can choose their implementation independently from the business rules and thus should be able to reuse the rules for different applications.

So, why doesn't it work that way, today? Well, the IT people require that these rules are precise, consistent and complete, just as the enforcers of law require from their legal drafters! When law enforcement needs to 'implement' a new task, they get legal documents that are supposed to be precise, consistent and complete, so why can't business people deliver their specifications to their IT department in that manner?

First of all the communication between business and IT has not such a long tradition as the legal department has with their enforcement counterpart. The legal people have developed extensive processes to maintain the law. The rules are well documented in books and there are dedicated programs at universities to educate legal experts.

Secondly there's this huge language problem that business and IT people seem to have. Legal departments and their enforcement counterpart share the same language, their natural language. Well, at least to a sufficient level, that is. I do not want to claim that there aren't any communication problems at the governmental level.

Legal departments and their enforcement counterpart have the same understanding of preciseness with respect to legal statements. Each and every word is either defined elsewhere in the legal system, or defined in a dictionary. When it comes to the communication between the business and the IT department, the business has to elaborate time and time again on the definitions of their terms and how terms are related. The most common reason for this inefficiency is that the business has never cared much about formalizing their definitions and was kind of hoping that the IT people would do that for them by means of their extensive analyses. But unfortunately, the IT people have only defined and documented just enough for the application at hand. And in time, when market demands change, when policies change, or the definitions of terms change, other IT people are going to interview other business people...

Here's where the Business Rules Approach (BRA) comes to the story.

1. **The BRA wants to accomplish that companies think of rule management as an ongoing activity.** Today business rules are 'harvested' in the scope of an IT project in order to get specifications for a particular application. Earlier in this article I emphasized the risk that

these rules will be biased to the application which will cause problems once the rules need to be re-used. Business departments should capture and maintain their rules and vocabulary in an application independent manner.

2. **The BRA wants to provide a common business rules language or syntax.** This syntax can serve as a standard for exchanging business rules between all kinds of environments for multiple purposes. It may be used to publish rules but it can also serve as high-level specifications for an application. It should not be too hard either, to generate software from this syntax. The syntax is declarative, of course, but it also described the business in an application independent way. This means that the statements are not directed to a particular application goal. Their purpose is to describe the business goals and requirements. In the context of an IT application, the application specific goal(s) can be used to generate efficient software.
3. **The BRA wants the business people to be more in control of their business rules, especially when it comes to business rules applications.** The business must be able to specify their goals and requirements to the IT department in such a fashion that they can be sure that an eventual application contains *their* rules. The business will be responsible for delivering their knowledge in a precise and complete manner. In return, if the IT department can rely on these specifications, the architects, developers and testers can devote their time to their own expertise. Both disciplines have their own methods of re-use; re-use of business rules is not an issue at all in the scope of the business and making technical components re-usable is already a well established technique.

Conclusion

When business rules are only harvested for, and maintained in application environments the re-usability of business rules is a big challenge. Only if rules management is separated from application development, both business and IT departments will achieve maximum flexibility.

The ability to generate different business rules applications from a single (re-usable) business rules model can only be achieved successfully if

- The business rules model is declarative;
- The business application generation process is goal-driven.

The business rules approach provides the necessary prerequisites for this flexibility.