SBBRENG: Spreadsheet Based Business Rule Engine

Pablo D. Palma

Incentings
Latadia 4623, Santiago, Chile
+562 2207 7158
pablo.palma@incentings.com

ABSTRACT
We developed a software product to replace the use of spreadsheets as a data processing solution within a specific business area. This paper explains the characteristics of the tool and the findings, both resulting from a process of 3 years real life refinement inside the ICM domain, and that we postulate can be valid in other business domains.

General Terms
Documentation, Design, Security, Human Factors, Languages. Verification

Keywords
Spreadsheet, Business Rules Engine, DDD, SEmS'14

1. INTRODUCTION

1.1 Use of spreadsheets for ICM solutions
The last four years our company has been working in the field of Incentive Compensation Management –ICM-. Current solutions, based on calculating performance-based payment for employees, are complex and highly dynamic.

Worldwide, "only a 10% of sales organizations with more than 100 payees deploy prepackaged sales ICM applications" [1]. Almost all the remainder uses Excel. This is a reaction to the combination of factors: high rate of change, short time available for implementation, and typically long cycles in IT development. However, Excel introduces its own limitations. It requires a lot of human intervention that results in overpayment, and user-generated errors that could be reduced "by more than 90%" [1] (see subsection 5.2). In addition, there is "dissatisfaction with the reliability of spreadsheets in adequately supporting compensation processes" [1]. Excel also does not accomplish auditing, accounting and regulation requirements.

In our market, the most important features of ICM software are flexibility, security, auditing capabilities, and allowing the end users to update the product themselves by including changes in business rules.

1.2 Goals for a new ICM software
Some or the issues of pre-existing ICM solutions are:

- World Class ICM software solutions are costly and demand long implementation processes
- In-house developments are slow[1] and rigid[2]
- Excel-based solutions are fragile, difficult to audit and error prone[3]
- Currently available solutions don't attempt to improve problem representation[4] beyond conventional system documentation
- Excel formulas are one-line expressions and are thus difficult to read (e.g. nested if statements)

1.3 Importance and state of our work
There are two elements we consider important. First, we are putting in practice some ideas (see Section 8) that may be useful in other areas of enterprise software development. Second, we want to determine how well our selection of functionalities succeeds in creating a tool that best takes advantage of a mental model of spreadsheets.

Customized ICM applications developed with SBBRENG have been in use for more than a year in several companies from different areas: car dealerships, banks, retail, etc. This happens in the Chilean market where we use the product name IM4

2. BUSINESS RULES ENGINE
Business rules engines aren't a new product category, they started around the 80s [2]. Since then, many products and companies have undergone a cycle of creation, development, merging and death. We will use two currently successful solutions as comparison standards: Drools (see Drools Guvnor Knowledge Base)[5] [3] -a component of the open source platform JBoss BRMS- and ODM [3, 4] by IBM. Both are much larger systems than SBBRENG, sharing the same global objective: make the application more *business agile*.

Drools is a low level programming environment oriented to efficiently manage a large quantity of conditions of any type. It provides APIs for integration with other languages, tools and processing environments. On the other hand, Operational Decision Management –ODM- is a more business oriented solution that conceptually splits systems into two different components, talking to each other under a data contract. One is a traditional Data Processing System for storing, updating and reporting information related to some business domain, and the other is a specialized system for managing and executing the business rules of the same business domain.

[1] in the range of 2 hrs per 2.000 transactions

[2] no provisions for isolation or special management of business logic

[3] http://eusprig.org/horror-stories.htm

[4] *problem representation* has impact on the maintenance agility same as on the ability to preserve application coherence

[5] http://drools.jboss.org/drools-guvnor.html

SBBRENG is closer to ODM with some big differences: domain model is not Object Oriented and input/output documents are simple shared folders for storing interchanging files

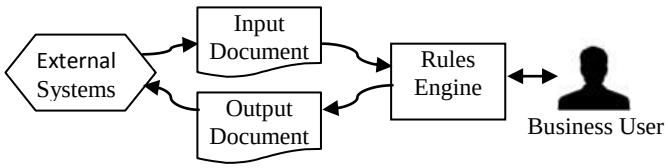

Figure 1: ODM high level view

3. THE SPREADSHEET-LIKE SIDE OF SBBRENG

3.1 The *ApplyRules* operation

A *WorkSheet* -WS- is a set of files and columns such as each column has a unique name and each cell stores an immutable value (current implementation does not yet force this immutability). A SBBRENG Process is a specific sequence of steps that modifies a WorkSheet. There is an operation *ApplyRules* (•) for implementing SBBRENG Processes, following statements of Business Rules. A Business Rule is -in the context of SBBRENG- a directive detailing how to calculate the numeric values used to run the business.

We represent a SBBRENG Process using the formula

$BR_k \bullet WS_p \Rightarrow (WS^k_p, OF_p)$

Where:
- • is the *ApplyRules* operation
- BR_k is a subset of the Business Rules comprising the Application
- WS_p is a current WorkSheet that is part of the Application
- WS^k_p is a new WorkSheet that will be part of the Application
- OF_p is an Output File that consists of a subset of WS^k_p

Operation • adds new columns at the right of WS_p. Business Rules define how to calculate the immutable values of the new cells. New columns can reference any column located at its left (Figure 2).

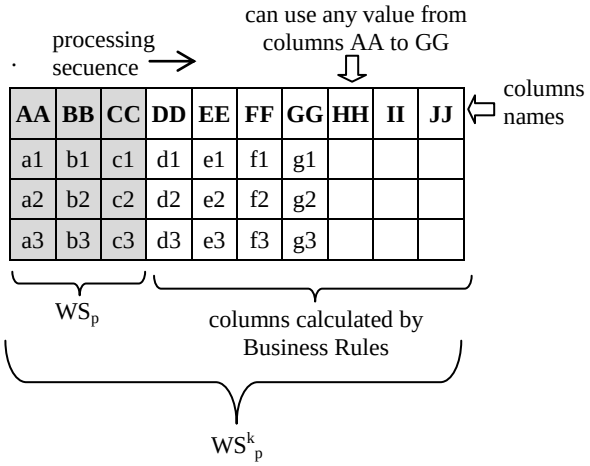

Figure 2: A SBBRENG Process

A SBBRENG Application is a sequence of Processes as seen below:

Process 1: $BR_1 \bullet WS_a$
Process 2: $BR_2 \bullet WS_b$
……
Process k: $BR_k \bullet WS_m$

3.2 The *Assemble* operation

An Input File becomes a WS when it contains all the information referenced by one or more Business Rules. When Business Rule references are contained in several Input Files, it is necessary to build a WorkSheet by assembling several Input Files. We use the *Assemble* operation (**+**) for this purpose, as shown in the following formula.

$$IF_i(p) + IF_k(q) \Rightarrow WS_a$$

Where **p** is a column of IF_i and **q** is a column of IF_k, and they provide a mechanism for matching rows of the Input Files.

Operation **+** produces a WorkSheet out of all the columns of both Input Files. The WorkSheet contains all the IF_i (p) rows and for each of them, only one matching IF_k (q) row. The matching logic is the same of an Excel Table Lookup operation, in which **p** is a column in the data and **q** represents the first column of the table.

The **+** operation is associative but not commutative.

The **+** operation can also be applied to a WorkSheet. In such cases we have a precedence of Processes. Figure 3 shows an example in which $BR_1 \bullet WS_a$ ***precedes*** $BR_2 \bullet WS_b$.

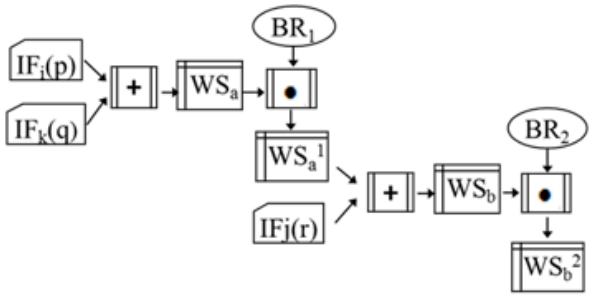

Figure 3: Process precedence

There are situations where it is necessary to assemble the same file more than one time, using different column keys. In such a case SBBRENG adds a prefix to column names to avoid collisions. In the following example, IF_k is applied twice:

$$(IF_i(p) + IF_k(q))(r) + IF_k(s) \Rightarrow WS_b$$

4. NON COMPATIBLE SPREADSHEET FUNTIONALITY

The most important difference with Spreadsheets is spreadsheet interactivity, because SBBRENG follows a batch processing model. Other examples of incompatible features are Table Lookup, Dynamic Tables, external links, totals and other

aggregated values in the same column as the original data, macros, different formulas in the same column, and the programming language.

5. SOME ADITIONAL FEATURES OF SBBRENG

5.1 Referential Transparency

A SBBRENG application offers "referential transparency", which is the base for providing reproducible results. In order to achieve that goal, it is necessary to replace links to external sources (other spreadsheets, Databases, etc.) by static Input Files containing the external linked information.

5.2 Separation of Data and Parameters

In the context of SBBRENG, Parameters are a special type of data: input files are produced by other systems, but parameters are maintained by users. Parameters represent a high level system abstraction, which is required to adapt the system behavior. Data is stored in Files and WorkSheets, and Parameters are stored in a special repository. SBBRENG's IDE provides the means for Parameter editing.

5.3 Iteration over Data

The calculus performed on each column follows a *cycle*. Rows are filtered by conditions and grouped by some column values. The logic applied to the cells belonging to a group, is repeated for each group until reaching the last. Some SBBRENG *core functions* offer aggregated operations over groups, e.g. count, sum, average, max.

5.4 Domain Model

5.4.1 Introduction

Five objects support Domain modeling: *Matrix, Classifier, List, Rules and Files*. *Files* are input/output files. *Rules* are pieces of code that have some specific properties (i.e. name, filter, sequence and granularity). The three remaining objects are the most important, because they store in their structure the values of the Parameters of the application. This allows a direct user interaction with the Domain Model representation, when adjusting Parameters values.

Parameters directly represent elements of the *ubiquitous language* [5] Those elements appear in several real life working documents: memos, agreements, contracts, regulations, etc. The shape of the Parameters as used in SBBRENG mimics its representation in documents. Therefore, business users understand them without requiring further explanations.

As needed, some Parameters may have embedded logic that is executed every time they are used in a Rule.

5.4.2 Matrixes

Matrixes are bi-dimensional arrays of values and conditions that return a value (or several values) based on the evaluation of its embedded logic.

Matrixes have two headings, X and Y. Each heading represents a tree of conditions: sibling nodes make an OR and parent-child nodes make an AND. It is very easy to see the tree as a set of adjacent boxes with the outermost boxes of the headings matching columns or rows of the Matrix. Each box has a label that makes apparent its associated condition.

		New Products			Old Products	
		Spot Clients	Recurrent Clients	Premiun Clients	Recurrent Clients	Premiun Clients
Salesmen type A		5%	6%	7%	7%	8%
Salesmen type B	North Region	4%	5%	6%	0%	0%
	South Region	3%	4%	5%	0%	0%
Salesmen Type C	North Region	5%	6%	7%	7%	8%
	Central Region	4%	5%	5%	6%	6%
	South Region	3%	4%	5%	0%	0%

Figure 4: A Matrix

Matrixes change the way in which complex nested conditions are visualized (see figure 5)

A				B					C
1	2	3	4	5			6		
				a	b	c	d	e	

```
  (A && (1 || 2 || 3))
|| (B && (5 && (a || b || c)) || (6 && (d || e))
||  C
```

Figure 5: Equivalence of nested conditions

Matrixes are self-explanatory for anyone familiar with the Business Domain of the application. Their behaviour doesn't depend on the context in which they are used; it only depends on the values of some of the input data in a clear and explicit manner. Matrixes provide a powerful mechanism of Domain representation, because of its expressivity and because of the way they isolate behavior.

5.4.3 Classifiers

Classifiers are Boolean expressions whose value is automatically set based exclusively on the input data and remain immutable until the input data change. They represent business concepts, mostly corresponding to nouns in the *ubiquitous language*. Regardless of how many relationships input fields have in the system they comes from, Classifiers implements only those conditions required by our application. Classifiers are used by Matrixes to build its embedded logic.

Classifiers create a conceptual layer for mapping a SBBRENG application Domain with the Domain of systems where the input data were generated. Classifiers are used by Matrixes to build its embedded logic. Classifiers increase program readability and improve our ability to adapt to changes in the Input Files.

5.4.4 List and Constants

A List is a Dictionary where a value associated to an entry can be simple or complex. Constants are Lists that use a special syntax.

5.5 Programming Language

We use JavaScript to replace spreadsheets' functions. To improve productivity, we developed a library of *"core functions"*

frequently used in our Domain of applications. It is easy to add new core functions.

We also provide a graphic block language, similar to MIT's Scratch [6] and others [7]. Blocks automatically generate the equivalent JavaScript instructions. Blocks are very well suited to SBBRENG because each Rule is made of a few instructions. Blocks were initially implemented for the *Assemble* operation, and we have plans to extend it to the Rules.

5.6 Auditing

A *Run* is a complete execution of a SBBRENG Application. Each Run is stored as a *backup document* containing all inputs, outputs, parameters and logic utilized. SBBRENG automatically assigns a unique ID to each Run. Later, a Run can be opened as read-only for revision, but it cannot be modified. It is possible to reprocess a backup document, generating a new backup document with a different ID.

Additionally, there is a log of the changes made to the parameters, the input files and the logic, indicating old and new values, the user involved and date/time of all changes.

5.7 IDE

There is a special IDE -Integrated Development Environment- to support all tasks: application development, documentation, design, testing, etc. It also has functions for running applications, for reviewing previous Runs and for downloading results.

The IDE offers two views: a conventional nested folders type and an advanced *mental map* type [8, 9]. The latter is the base for some advanced visualization options that ease the understanding of an Application (pending development).

5.8 Documentation

Documentation is a part of a broader content we call *problem representation.* It includes parameters, code, blocks, ad-hoc descriptions, etc. Additional to the content, there are tools for filtering information, displaying information, and displaying information relationships. Some of this functionality is currently in use; some is pending development. Because documentation is supported by the IDE, it is always available on line when working with the application.

6. SOFTWARE STRUCTURE

On the Server side, there is a Web application than runs on IIS using .NET and SQL Server.

On the Client side, there is the IDE running in any modern browser.

7. RESULTS

Security and auditability of the applications were improved in relationship to spreadsheets, as a result of some new specific functionality (see Subsections 5.1, 5.2 and 5.6).

Documentation was improved when compared to conventional solutions, because of the integration of different types of information into one common repository (see Subsections 5.7, 5.8) and the availability of new capabilities based on the use of a Mental Map.

The use of the Domain Model (see Subsection 5.4) enhanced productivity of development and maintenance, because less code is required to implement the same business logic compared to solutions using spreadsheet (See Sub Subsection 5.4.2)

Performance is good. We were expecting 10 min per 2.000 transactions and 4 hrs per 3.000.000 transactions, but real numbers were 4 min and 1 hr 45min, respectively. We were using a conventional entry level server.

8. KEY LESSONS LEARNED FROM WORKING WITH SBBRENG

Looking at one of the components of the productivity equation, we think we successfully tried some new ideas, like a new approach for representing the Business Rules Domain, a method for avoiding complex nested conditions, an IDE based in a Mental Map, a graphic replacement for the programming language of spreadsheets, some mechanisms to improve security and auditability, etc.

But looking at the other component -the process of getting and agreeing to specifications for building the application- we think it is necessary to achieve important improvements. The ubiquitous language requires more elaboration[6]. The cognitive process that ends with a working application can probably take advantage of the impressive new findings in neuroscience. Focus, resources, new instruments and new methodologies are moving the limits. “Constant development of more sensitive and accurate neuroimaging and data analysis methods creates new research possibilities” [10].

9. FUTURE DEVELOPMENTS

We are interested in two areas for future development. The first is improving automatic analysis capabilities used during the testing phase, and the other is improving visualization capabilities for mental maps in the IDE.

10. REFERENCES

[1] Dunne, M. 2010. *MarketScope for Sales Incentive Compensation Management Software*. Gartner MarketScope Series (March 2010)

[2] Bosh 2010. *The Past, Present, and Future of Business Rules*. Bosch Software Innovations GmbH. (March 2010)

[3] Craggs, S. 2012. *Competitive Review of Operational Decision Management*, Lustratus Research (October 2012)

[4] IBM 2012. *Why IBM Operational Decision Management?* Software. Thought Leadership White Paper (June 2012)

[5] Evans, E. 2003. *Domain-Driven Design*. Addison Wesley; E (August 2003)

[6] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai Y. *Scratch: Programming for all*. Communications of the ACM (November 2009)

[6] it has been apparent that some additional concepts are necessary

[7] Hosick, E. 2014. Visual Programming Languages - Snapshots. (February 2014) http://blog.interfacevision.com/design/design-visual-progarmming-languages-snapshots/

[8] Eppler, M. 2006. A comparison between concept maps, mindmaps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. Faculty of Communication Sciences, University of Lugano (USI), Lugano, Switzerland

[9] Novak, J., and Cañas, A. 2008. *The Theory Underlying Concept Maps and How to Construct and Use Them.* Florida Institute for Human and Machine Cognition (IHMC)

[10] Jääskeläinen, L. 20012. Cognitive Neuroscience: Understanding the neural basis of the human mind . Jääskeläinen & Ventus Publishing ApS (November 2012)| http://bookboon.com/

.