MANAGING BUSINESS BY THE RULES

David C. Hay, Essential Strategies, Inc.

I had an aunt in Yucatan
Who bought a Python from a man
And kept it for a pet.
She died, because she never knew
These simple little rules and few; –
The Snake is living yet.

– Hilaire Belloc

Introduction

It has been interesting watching the development of methodology for systems development grow over the last thirty years. When I started in this industry, there was just programming. A programmer was an artiste, with no one really knowing what he did or how he did it. In the early seventies came structured programming, which finally made it possible to test programs. Structured design followed, providing some guidelines as to how to organize a program into modules. Structured analysis in the late seventies provided the beginnings of a way to define more clearly what it is that programs are supposed to be doing in the first place.

These techniques were elaborations on the problem of how to address business and program processes more effectively. During the 1980's, things went in a different direction: information engineering and object-oriented programming moved the emphasis in both analysis and design away from program processes to the organization and structure of *data*. Data it was discovered, better a reflected the organization and structure of the company being served. Data and object modeling moved the focus from what the company does to the objects of significance it manipulates while doing it.

In the 1990's we have discovered that this still isn't enough. In capturing processes and data, we still haven't completely captured the essence of what an organization is about. And by not learning "these simple little rules and few", we left the programmers to improvise and fill in the gaps.

What we have discovered is that there is yet another body of knowledge that is critical to understanding the nature of an organization – its ***business rules***.

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. A business rule is created in response to constraints imposed upon the business. It in turn then constrains some action within the business. In order to understand business rules, it is important to understand the different perspectives held by different players in the development process.

The Zachman Framework

By 1987, even though the industry was awash with methodologies, modeling notations, and different ways to "communicate", we in fact rarely communicated very well at all – with each other or with our clients. In response, John Zachman came up with some important insights, and from them developed his "Framework for Information Systems Architecture".[1] Among other things, Mr. Zachman realized that a source of our

---

[1] Zachman, John, "A Framework for Information Systems Architecture", *IBM Systems Journal*, Vol 26, No 3, 1987. Also see Hay, David C., "The Zachman Framework", *Real World/Client Server – ECO 96*, Oracle User Resource, 1996. Available at *http://www.essentialstrategies.com/publications/methodology/zachman.htm.*

communications problems was that each of us views the problem of information system development from a quite different point of view. Mr. Zachman's recognition of these points of view was important, because it finally clarified why it is that we have such difficulty talking to each other. In particular, he identified six different perspectives in any system development project:

? ***Scope*** – The understanding of why the organization exists, how it is like other organizations in the same industry, and what makes it distinctive.
? ***Business owner's view*** – This refers to the perspective not of the stockholders, but of the people who operate the business. This encompasses all the jargon of the business, as well as an understanding of how everything actually works.
? ***Architect's view*** – Recognition that there are fundamental, technologically-independent structures present, and representation of the business in terms of these structures.
? ***Designer's view*** – The application of technology to address any information system requirements discovered in the views above. The perspective here is technological: relational data base systems, object orientation, network protocols, and so forth.
? ***Builder's view*** – This is the view of the inside of the programs and technology. The builder knows the finer points of the programming language, or the communications technology, or whatever.
? ***Production view*** – The view of the completed system.

If two people are coming at the problem from different perspectives, their language and terms of reference will be different, and they will be working at cross purposes, unless they recognize the differences in their points of view – and translate. With the translation it becomes clear to both why they believe what they do – and it becomes clear that, as long as they recognize and respect each other's positions, this doesn't matter. Each has a different job to do, and the perspective of each is appropriate to the carrying out of those jobs.

If these are the different perspectives in the system development process, what is it that these perspectives are looking at? In the history of systems development, we have already identified two: process and data. In the era of distributed systems, it's not hard to extend this to a third – location. Where is data to be stored and viewed, and where is processing to take place? When John Zachman first laid out his framework, he did it in terms of these three, as columns in a matrix, with six rows representing the six perspectives. Each cell, then, represented a perspective's view of a particular aspect of system development. The data model was used for the architect's view of data, for example, while the physical data flow diagram showed the business owner's view of process.

Then he realized that these columns were nothing other than the journalist's "what", "how", and "where". As a frustrated journalism student, he knew that he needed three more columns: "who", "when", and "why".

The first two were no problem. Of course when we plan a system, we have to take into account who the players will be, both in terms of the overall organizational hierarchy, and in terms of the specific roles to be played in doing each thing in the business. Addressing "when" was consistent with the movement to event analysis, state transition diagrams, and so forth.

The problem in 1987 was "why?" To be sure, at the top row, you have business objectives and plans, but how do these translate into the concerns of business owners, architects and designers?

Enter ***business rules.*** In the course of the last ten years or so, the industry has begun to develop and take seriously a body of knowledge describing how to address the constraints of a business. The IBM User Group, GUIDE, published a report in 1995 to articulate the requirements and possible approaches to specifying business rules.[2] Ron Ross has written a comprehensive book on the subject.[3] Barbara von

---

[2] Hay, David C. and Keri Anderson-Healy, *Defining Business Rules – What are they really?* GUIDE Project on Business Rules, *http://www.guide.org/ap/apbrules.htm*, 1995.

[3] Ronald G. Ross, *The Business Rule Book, Second Edition,* (Boston:Database Research Group, Inc., 1997).

Halle has written numerous articles for *Database Programming and Design*.[4] More people are writing about business rules every day.

Finally, John Zachman could fill out his matrix, as shown in Figure 1.

	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motiva-tion (Why)
Objectives / Scope	List of things important to the enterprise	List of processes the enterprise performs	List of locations where the enterprise operates	List of organization al units	List of business events / cycles	List of business goals / strategies
Model of the Business	Entity relationship diagram (including m:m, n-ary, attributed relationships)	Business process model (physical data flow diagram)	Logistics network (nodes and links)	Organization chart, with roles; skill sets; security issues.	Business master schedule	Business constraints
Model of the Infor-mation System	Data model (converged entities, fully normalized)	Essential Data flow diagram; application architecture	Distributed system architecture	Human interface architecture (roles, data, access)	Dependency diagram, entity life history (process structure)	Business rule model
Techno-logy Model	Data architecture (tables and columns); map to legacy data	System design: structure chart, pseudo-code	System architecture (hardware, software types)	User interface (how the system will behave); security design	"Control flow" diagram (control structure)	Business rule design
Detailed Represent ation	Data design (denormal-ized), physical storage design	Detailed Program Design	Network architecture	Screens, security architecture (who can see what?)	Timing definitions	Rule specification in program logic
Function-	(Working systems)					
ing system	Converted data	Executable programs	Communica-tions facilities	Trained people	Business events	Enforced rules

Figure 1: The Zachman Framework

Specifically, column six of the matrix turned out to be all about business rules. Based on the goals and objectives of the organization, as well as on external constraints operating on it, business rules are formulated to determine exactly what the company may, must, or must not do.

---

[4] Her first three were: Barbara von Halle, *Database Programming and Design*, "Back to the Business Rule Basics", October, 1994, "Living by the Rules", November, 1994, "Lessons to Learn from Tee-ball", December, 1994.

If row one specifies the overall goals and objectives, in row two, you have the constraints on actions that are perceived by the people who run the business. In row three, you have the implications of these constraints on an information system architecture. Row four and below are concerned with how business rules are implemented in a system design.

The rest of this paper discusses how the various cells of the Zachman Framework interact with each other to define the rules by which the business operates.

Rules and Columns

To understand business rules, it is important to understand the relationship of column six to the other columns in the framework – specifically column one.

In 1995, a project organized by GUIDE, the IBM user group, published the paper mentioned above. It defines a set of terminology for referring to business rules.[5] In it, the overall definition of business rules encompassed both columns one and six. That is, there are actually three different kinds of business rules:

? ***Structural assertions***, encompassing *terms* and *facts*.
? ***Action assertions***, constraining what a business can do.
? ***Derivations***, creating data from other data.

Structural assertions are what a data model is all about. A structural assertion is a statement that something of importance to the business either exists as a concept of interest, or exists in relationship to another thing of interest. It details a specific, static aspect of the business, expressing things known or how known things fit together.[6] What are the ***terms*** used in the business, and what do they mean? It turns out that the definitions of words are the first set of constraints operating on an enterprise. ***Facts*** are the second – represented in a data model as either relationships between entities or between entities and attributes. That the entity PERSON has "Given name" as an attribute is a fact. That a PARTY is "buyer in" some number of contracts is a fact.

A ***derived fact*** is created by a ***derivation***, which is an inference or a mathematical calculation from terms, facts, other derivations, or even action assertions. Derivations traditionally have not been shown on data models. It was considered bad form in relational theory to show derived columns, since their information content is carried in their component rows. In point of fact, however, derived columns are often important to understanding the nature of the data, and are very much part of system requirements. It is not a big stretch, however, to include them in the data model, with an appropriate annotation (such as preceding the attribute name with a "/").

An ***action assertion*** is a statement that concerns some dynamic aspect of the business. It specifies constraints on the results that actions can produce. Where the structural assertions describe possibilities, action assertions impose constraints – "must" (or, "should") or "must not" (or, "should not").[7]

Note that a data model is an excellent way to represent structural assertions. An entity represents a term. Facts are either represented by relationships between entities, or by the relationships between attributes and entities. A data model is a very poor way to represent action assertions, with a couple of exceptions: Whenever you say that an occurrence of an entity must be related to *at least one* occurrence of another, you are making an action assertion, not a structural one. All you can say from a structural point of view is that the possibility exists that an occurrence of entity 1 may be related to an occurrence of entity 2. Similarly, you are making an action assertion when you say that an occurrence of the first entity may be related to *no more than one* occurrence of a second entity. In the Oracle notation, you can also assert that one entity must be related to *either* one occurrence of a second entity, *or* an occurrence of a third entity.

---

5 Hay, Anderson-Healy, *op cit.*

6 *Ibid., http://www.guide.org/ap/apbrc4.htm#s4.*

7 *Ibid., http://www.guide.org/ap/apbrc5.htm#s5.*

But that's it. There are many action assertions that cannot easily be expressed in data models.[8] You cannot constrain a recursive relationship, for example, to assert that it shouldn't loop. You cannot coordinate …TYPE entities, to insure that if, for example, an ORGANIZATION is part of another ORGANIZATION, the ORGANIZATION TYPES of these ORGANIZATIONS have the same structure. And any time you have multiple paths through a model between two entities, you cannot (in the model) require that occurrences of the entities and relationships involved be consistent with each other.

Note by the way, that the more generic a model is, the fewer action assertions it can represent. This is not so remarkable, since models are made more generic to insulate them from problems associated with changing requirements. The ideal data model should represent only that which is fundamental and unchanging about the business. Business rules, by and large, are not fundamental and unchanging. Since they are derived from business policy, they are, in effect, derived from company management's present view of the world. There is every reason to believe that this view will change tomorrow, and with it the business rules.

For the action assertions, you have to move to a column six model. Ron Ross has published just such a technique for doing this.[9]

He sees two basic kinds of rule elements:

? One describes an ***integrity constraint.*** This is something that *must be true* about an entity relationship or attribute, by definition.
? The other describes a ***condition.*** This may be true or false. Depending on the condition other constraints may apply.

Each rule describes the effect of a ***correspondent*** (constraining object) upon an ***anchor*** (constrained object). In the upper part of Figure 2 is an *integrity constraint*, represented by the arrowhead with the "X" in it. (This is a modification to Mr. Ross's notation, to adapt it to the Oracle Method. In his notation, attributes are shown in circles, outside the entity box.) An integrity constraint must be true. The constraint shown says that any occurrence of CUSTOMER must have a value for the attribute "address". CUSTOMER is the anchor, and "address" is the correspondent. The "X" in the symbol is one of 42 ***rule types*** that Mr. Ross describes. It indicates the rule type "mandatory". That is, it asserts that the anchor "must have" an occurrence of the correspondent.

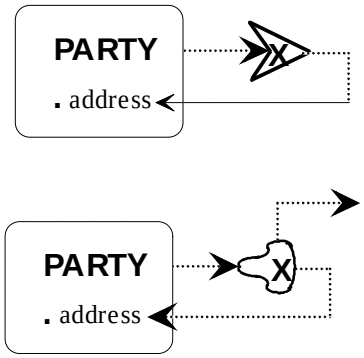

Figure 2: A Business Rule

8 See, Hay, David C., "Data Modeling by the Rules (or was that Next to the Rules?)", *Client/Server Fast Forward Procedings – ECO 95*, Oracle User Resource, 1995. Also available at *http://www.essentialstrategies.com/publications/businessrules/brules.htm.*

9 Ross, *op cit.*

In the lower part of Figure 2 is a *condition.* This says that *if* an occurrence of CUSTOMER has a value for the attribute "address", then carry out the next part of the rule. Note that rule elements can indeed be strung together.

Each of the 42 rule types may be used as an integrity constraint or a condition to describe a particular situation. Mr. Ross contends that this list of rule types is atomic and fundamental. While extensive, Mr. Ross does not claim that the list is exhaustive. There are undoubtedly more to be discovered. The list is divided into nine categories:

? ***Instance verifiers***, requiring an occurrence of an entity to be present at the creation of, during the life of, etc. an occurrence of another entity. For example, "The 'Shipping destination' of an ORDER with more than ten LINE ITEMS may not be changed."
? ***Type verifiers***, linking multiple conditions to constitute a rule. For example, "An ORDER must indicate the 'Date placed' and CUSTOMER that is the buyer in that ORDER."
? ***Position verifiers,*** requiring occurrences to be created in a particular order. For example, "If this SECTION is among the first five in a REPORT, then . . . "
? ***Functional verifiers,*** requiring occurrences of an entity to be unique, ascending, fluctuating, etc. For example, "Period" for two instances of RENTAL AGREEMENT for one FACILITY may not overlap."
? ***Comparative evaluators*** allow for comparisons (less than, equal, etc.) between the attributes of two occurrences of the same or different entities. For example, "The "Outstanding balance" owed by a PARTY must be less than or equal to its "Credit authorization".
? ***Mathematical evaluators***, are used in a rule to derive values used by the rule. For example, "The salary cap for a DEPARTMENT must exceed the sum of the "SALARY" of all EMPLOYEES working for that DEPARTMENT.
? ***Projection controllers*** cause something to be updated when the anchor is true. For example, "An EMPLOYEE assigned to manage a PROJECT is automatically a MANAGER.

As stated above, Mr. Ross's notation is atomic. That is, each of the rule types he defines allows one to assert a truth about the business at a very low level of detail. This has the effect of making models of rules using this notation very intricate – more intricate than some have patience for. Short of drawing these pictures, however, it is possible to take advantage of Mr. Ross's insights by at least documenting rules associated with the data model. Thus, the narrative that documents the data model can document rules as well.

Rules and Rows

Row 2: Owner's View

As described above, the "Scope" view of the "why" column is concerned with overall corporate objectives. It is when you move to the "Owner's" view that you have to contend with the specifics of how the objectives are to be realized. Clearly they are partly being addressed by collecting data, establishing procedures, and so forth. But most directly they are addressed by imposing constraints on the actions of the business. An salesman must call on ten prospects in a week. Completed production lots must pass a specified set of quality control inspections. Male employees must wear a coat and tie. And so forth.

Note that these business rules are actually derived from business policies. A business policy may be a mission, strategy, or tactic. Business policies in turn are formulated in response to either opportunities or influences by other constraints. An influence may be either a risk or a threat. Constraints may be one of two kinds: Internal constraints, which include not only corporate objectives, but also corporate culture, business rules imposed at a higher level in the organization; and external constraints, imposed by the marketplace, government regulations, the culture of the community, and so forth.

That is, a particular department creates a business rule to constrain the operation of its components, based upon the business policies under which it must operate. Business rules, in turn, require, allow, or prevent actions by components of the department.

Row 3: Architect's view

Where, in the Owner's view, a business rule constrains action, in the Architect's view, the great insight here is that a business rule constrains data. Note that it does not directly control a process, although at lower rows in the Framework, processes may be used to implement rules. Ultimately, a business rule is expressed declaratively – in terms of what data may be, must be or must not be updated. This may seem counter-intuitive, since rules have traditionally been *implemented* in program code – processes. If two different processes update a value, it is in the nature of the rule that the value must be within certain limits for both processes.

Mr. Ross's notation, described above, specifically is appropriate for describing these row three rules. That is, an attribute of an entity in the row three, column one cell is constrained from having certain values by a rule expressed in row three, column six.

Conclusions

It is only recently that we have come to appreciate the importance of business rules to our understanding of just how a business works. This understanding promises to improve significantly our ability to respond to the needs of our system development clients.

About the Author . . .

A thirty year veteran of the Information Industry, Dave Hay has been producing data models to support strategic information planning and requirements planning for over thirteen years. He has worked in a variety of industries, including, among others, power generation, clinical pharmaceutical research, oil refining, forestry, and broadcast. He is President of Essential Strategies, Inc., a consulting firm dedicated to helping clients define corporate information architecture, identify requirements, and plan strategies for the implementation of new systems.

He is the author of the book from Dorset House Publishers, *Data Model Patterns: Conventions of Thought*. He may be reached at 800-597-4553 or davehay@essentialstrategies.com.