Business RULES

Show Power, Promise

by Ellen Gottesdiener, President, EBG Consulting, Inc.©

This article was published in the March, 1997 issue of Application Development Trends, vol. 4., no. 3

While a business rules approach can present many challenges to I/S development managers, backers of the approach view it as an effective way to identify core business requirements.

The term, 'business rules' has with different meanings for both business and I/S professionals. Within I/S, 'business rules' has a different meaning for business and I/S professionals. Within I/S, 'business rules' can have different connotations depending on whether the perspective is 'data-oriented', 'object-oriented' or 'expert system-oriented.' The elasticity of 'business rules' is further stretched by vendors using the term in marketing literature and World-Wide Web sites. Nevertheless, 'business rules' has become an accepted I/S buzzword.

An understanding of business rules is spreading, though it is still a newer area of focus. Business rules are beginning to be recognized as a distinct concept, practice, methodology and/or requirements technique.

This article will address several questions on business rules - What they are? Why are they important? How are they used? This article will also examine the debates and products in the business rules world.

What is a Business Rule

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" (see Reference 1).

When a business rule statement is elicited from a business person, however, it is often worded in ambiguous, non-rigorous ways. In such a situation, each business rule statement may actually decompose into numerous discrete business rules.

For example, a business person might state: "the total number of regulated products sold in a country must not exceed the threshold defined by the regulatory limits of the country of origin". This statement is more like a 'business rambling', a term coined by business rule expert, Barbara von Halle, business rule expert and co-founder of Knowledge Partners Inc., Mendham, N.J. This 'business rambling' statement contains many discrete business rules. There are implied definitions ('regulated product', 'country',); derivations ('total number of..'; 'threshold', 'regulatory limit'); facts ('products sold in a country'; 'limits defined by country of origin'); and constraints ('total' not exceeding 'threshold'). (See Six Categories of Business Rules, page xx).

When decomposed to their most elementary form, a business rule is as atomic (indivisible) as possible while still being inclusive enough to be one complete thought. A business rule does *not* contain control flow statements typically found in program logic such as notification, database updates, and sequence (See

"Procedural vs. Declarative", page xx). An atomic business rule written in a declarative (non-procedural) fashion, using standard grammar in a natural language that business people can readily understand, is not ambiguous.

A business rule is independent of the modeling paradigm or technical platform. A rule is defined and owned by business people. To summarize, business rule

are:

- Declarative (i.e. non-procedural);
- Atomic (indivisible yet inclusive);
- Expressed in natural language;
- Distinct, independent constructs;
- Business, not technology, oriented;
- Business, not technology, owned;

Commercial enterprises are comprised of thousands of combinations of such rules, which work at an operational level running the business. Business rules define and control the lifecycle of products and services and the supporting infrastructure. Business rules direct how enterprises buy, create, sell, cultivate, conform, employ, manufacture, research, report, and plan. As such, they are the core of the enterprise (See Figure 1).

Ron Ross, editor, Database Newsletter is the author of The Business Rule Book (see Reference 2), likens business rules to the periodic table of elements. In fact, he has devised an atomic chart of several dozen rule types. These atomic rule types can be combined to form an unlimited number of compound rule types, which he calls derivatives. "Individual business rules make a business understandable, but it's their chemistry that brings it to life", said Ross.

Articles on business rules began to appear in the early 1990's. Today there are publications dedicated to business rules and business rule tools. In addition, there are seminars and a conference on business rules a business rules conference, and seminars on business rules. An active Guide committee completed a three-year effort and published its first white paper on business rules in 1995. Industry analysts are beginning to write about business rules and related tools. Large companies, particularly in the insurance, finance, and banking industries are beginning to invest in building or buying business rule servers. (see References & Resources).

Business Rules and Expert Systems

The fields of business rules and expert systems do overlap. (See "Rule-based Systems have Roots in AI", page xx). "Rules have been used extensively as a way to represent knowledge within the Artificial Intelligence (AI) community since the late 1960's. In the 1970's, expert systems emerged as an identifiable part of AI and logic began to be used as a programming language (Prolog)", noted Margaret Thorpe, President of Tangram, a Washington D.C. consulting firm. Traditionally, expert systems have been used to derive or confirm knowledge based on a complex network of rules. They have not traditionally been associated with operational systems, but that is changing as they become embedded in more conventional business applications. "Business rules are something that by and large are enforced in real time. You want to influence and control behavior as the business occurs", explained Ross.

The technology underlying expert systems may be one way to automate business rules, based on the success of the Knowledge Acquisition and Rule Management Assistant (Karma) application at the Federal National Mortgage Association (Fannie Mae), Washington, DC. (See "Fannie Mae's Karma Helps to Get to Rules", page xx). Karma's Business Rule Server provides business rule validation for Fannie Mae loan applications.

IT's attraction to business rules

Business rules can offer several benefits, including: technical independence; faster application development; better quality requirements; ease of change; and a balance between flexibility and centralized control. The approach starts with a core - the business rules defined by business people. "Business rules are understandable to business people more than any other component we have used in developing systems", says Barbara von Halle, of Knowledge Partners. Business rules place business people at the center of any software activity. "The business rules approach seeks to be a true integration between business people and technology", echoed Bonnie O'Neil, director of data warehousing and business rules at Northern Lights, a Saratoga Springs, NY, consulting firm.

Allowing business rules to be defined and managed separately, and tying software engineering into them by generating and maintaining applications from the business rules, has great potential for evolving the I/S state of the art. The idea of 'separation' is not new. The value of separating business rules from data and presentation is central to the much discussed 3- or n-tier logical client/server application architecture (See Figure 3). Separating business rules from the data and presentation layers facilitates change, promotes reuse, and permits heterogeneity and scalability of the technical architecture.

However, what is often meant when vendors and analysts reference this ideal architecture is *application logic*, not declarative business rules. They frequently mean procedural code within which business rules are embedded. Some products that claim to provide separate business rule management have, in their proprietary 'business rule' specification, language which includes procedural verbs, for example: 'Call', 'Get', 'Forall', 'Replace', 'Display', 'Execute'. Thus the 'business rules' are not separately managed (e.g. in a repository) nor are they

non-declarative.

The widely used term 'business rules' is further diffused when database facilities are touted as business rules management. Database triggers and procedures are used to implement certain types of business rules, such as facts, constraints, actions enablers, and derivations. Triggers, for example, are commonly used to enforce referential integrity, a constraint type of rule (for example, "an order must be placed by only one customer") which is translated into database events (such as an order is being inserted or a deleted). Stored procedures and triggers enhance application performance. They violate the ideal of separating logical application layers, however, by coupling business rules to a particular database.

Separation yields flexibility. Business rules should be logically separated, into what Bonnie O'Neil, calls "a virtual tier". This tier can be moved to whatever technical platform is necessary to suit the performance and platform requirements of an application. "The business rule layer can be physically implemented anywhere, but logically resides between the interface and data layers", said von Halle.

Changes in policies, regulations, recipes, standards, formulas, or any other precept of the business results in changes to one or more atomic business rules. A business rule approach which has captured and automated those business rules should provide the flexibility to change the rule immediately and propagate

it throughout the software applications that subscribes to that rule. Additionally, a business rule approach has huge potential for re-thinking how to business process reengineer. "Business rules allow you to address key business constraints independently of responsibility - who's responsible, where they're enforced, etc. By so doing, you are then in a position to re-optimize the rules", pointed out Ron Ross.

The business rule approach is evolving due to the increased complexity of our business and system development environments, according to Tangram's Thorpe. "We have different applications, running on different platforms, which were developed using different methodologies, coded in different programming languages and accessing a variety of DBMS's. The business requirements are rarely maintained at the level of the business model or the specification. As a result, we have business logic replicated and buried throughout systems, oftentimes out of synch with each other. The business rule approach holds the promise of consistency and control of both specified and implemented business requirements", Thorpe said.

Although I/S may suggest a business rule approach, business people are the benefactors and sponsors. They readily understand the benefits in terms of ease of use and improved communications. An additional benefit is speed of change. "Giving the business community the tools to quickly define, modify, and check business rules for consistency as well as to get those changes quickly implemented in the production environment has made them able to respond much more quickly to changes in the business environment", said Thorpe.

The "P" word

Depending on whether one's orientation is object-oriented, data-oriented, business process-oriented or something else, business rules may or may not represent a new I/S 'paradigm'. Enthusiasts see a business rule approach as a new paradigm for thinking about and organizing models and information systems (See Figure 4). Some in the 'data' world debate the new paradigm suggestion since they have traditionally been representing business rules in their data models and they usually capture other types of business rule in documentation. "Accommodating business rules is not a paradigm shift, but something we've been doing all along", said Tom Bruce, orincipal of Tabset, a Berkeley Califl, consultant and author of Designing Quality Databases with IDEF1X Information Models (See Reference 3).

The weakness in using solely a data modeling approach alone is clear. "In information modeling, we can do a good job of capturing business rule as part of defining entities, attributes and cardinality", said Bruce. "What we don't do a great job of, for example, is defining rules like 'checking account can be provided overdraft protection by a credit card only if the account and credit card has a common owner'. Such rule types are not accommodated in model graphics. Most data modelers will notice this and pass this on - in list of rules or somewhere else".

Interestingly, many of the foremost promoters of the business rule approach hail from the data modeling community. "Data modeling is disciplined. In addition, business rules are a shared resource, like data", observed von Halle. "Unlike other systems development approaches, the need evolved from the business and from people experienced with I/I methodologies. Adaptability is a central need of the business", said Ross.

Object proponents disagree on where to put the business rules in object/class models. Some believe that rules are already implied in the model. Still others in the object-oriented modeling community are working to synergize numerous paradigms, including business rules (See "Merging data and object models with business rules", page xx).

Dan Tasker, Senior I/S consultant in the systems integration unit of Air New Zealand', New Zealand, and author of The Problem Space (see Reference 4), notes: "Each paradigm has a specific set of 'constructs' which it defines and utilizes to model something. No single paradigm covers all aspects. For example, data modeling, and object modeling, do a poor job of modeling the event perspective. Event models do a good job of modeling events, but have limited ability to describe data. The rule paradigm has, as its primary strength, modeling conditions, but is not strong in the area of workflow. By their nature, we want our rules to be declarative, leaving the issue of 'how' hidden or left to the implementation", said Tasker.

Business rules as part of requirements gathering and application enforcement have not been ignored by data, object, information engineering, or other methodologies. Those methodologies, to varying degrees, subsume or represent business rules as part of the notation schemes use to specify application requirements. Business rules are not data, domains, object, classes, messages, events, procedures, or actions. They, as Barbara von Halle says, "just are".

"Business rules are a great classic paradigm integrator which enable you to 'cross paradigms' between object-oriented modeling and relational design", said O'Neil. "Most other shifts like Information Engineering and object-orientation involved technology only, and for the most part are invisible to business people. I/S doesn't need another modeling paradigm. The key is being able to express the business in terms everyone - I/S and business - can understand", said O'Neil.

Defining requirements for application development has traditionally focused on a combination of data and process, or on objects which encapsulate data and process. Business rules break away from those points of view, putting business rules in the center (See Figure 5). The approach asks business and I/s people to look at requirements differently - as a set of business rules that must be enforced. These same rules, however, may change at any time due to external forces or internal policy changes. This presents an opportunity to express and think about those requirements in a different way.

Business Rule Templates

A solid application architecture is based on well-designed models, such as data, process, or class models. Rather than building those models from scratch, I/S can buy the models from a third-party, tweak them for the specific business, and then generate software from those modified design models using a CASE tool (See Reference 5). This presents opportunities for implementation of a business rules approach.

"I hope we'll start seeing templates which include data models and business rules as two deliverables. Then analysts can go in and change the templates and customize the package," said von Halle.

IBM's Insurance Application Architecture (IAA) template uses the construct of business rules. The IAA, first shipped in 1992 and eventually licensed to 25 sites worldwide, was created by a consortium of 40 insurance companies in a project coordinated by IBM. The IAA templates included three types of models: a data model with some 170 entities, a function model with some 500 functions, and a function flow model describing events, control flow, and data conditions.

The IAA data model separates business rules in two entities for generalized treatment - Classification and Specification. Rules around each data element are stored in the Classification entity and can be combined and rolled-up using the Specification entity to form complex business rules. A set of Classifications such as "customer that owns two vehicles "; "a mini-van vehicle" are used to define a Specification such as a product with certain coverages and services. In this way, the Classification and Specification entities are used to store business rules and permit their reuse.

Business customers and I/S, working with the IAA Classifications Manager (a CASE-like tool) in Joint Applications Design (JAD) - facilitated sessions, define and model product-specific business entities. The IAA separately handles derivation rules within the function flow model as part of procedural logic.

In addition to the interest by business customers in managing their business at a business rule level, sales of IAA have been driven by business process reengineering (BPR), said Tom Romeo, asset manager in IBM's Insurance Solutions Department. "Reuse should come from the business process level. Reuse is not coming from using relational technology. We need a model that is abstracted and has business rules separated from the business classes".

Business Rule Automation

"Having the rules doesn't really help that much if you don't have an automated way to enforce them", said O'Neil of Northern Lights. Vendors that have added the term 'business rules' to their product descriptions vary in how they interpret the term. Some use the term to describe how a tool implements business logic (procedural code). Few actually implement rules in a business rules repository. (See Table 1).

Application Generators that integrate business rules

ObjectStar from Antares Alliance Group, Dallas, Texas generates applications making using a proprietary business rules language which reads like conditional statements, according to Erika Kelly, systems supervisor, Farm Bureau, an Indianapolis, Ind., insurance firm. Farm Bureau selected the ObjectStar tool to extend its mainframe applications so the software can be moved to client/server environments when necessary. The Farm Bureau will continue running mainframe software for data management and business processing services.

ObjectPool from Sapiens, Tel Aviv, Israel encapsulates business rules within models it calls objects. The tool includes five types of rules, including two that are more procedural. ObjectPool rules are: local check rules which validate domain values; local computation rules which declaratively define algorithms and derivation rules which constrain or trigger actions. The procedural rules are: Call; and Fetch. The tool uses a proprietary inference engine to execute rules. The declarative approach yields less code, said Peter Barber, executive vice president of Sapiens' North American unit, based in Raleigh, N.C. For example, one Cobol application rebuilt using ObjectPool required one-third the lines of code of the original, he said.

Performer, a model-driven application development tool from Texas Instruments, Inc., Plano, Texas, promotes business rules. Performer produces models that are rooted in information engineering techniques - data model, logic design (essentially structured pseudo-code) and window navigation. The logic design portion of the tool is promoted as managing business rules, but is actually a scripting language comprised of some 40 verbs, both declarative and procedural. Code is generated by the models stored in the repository.

Repository-driven Application Generators

Two products stand out for automating declarative business rules using a centralized repository : Vision Builder from Vision Software, Oakland, Calif., and Usoft Developer from Usoft Corp., Brisbane, Calif. The Vision Software and Usoft tools are based on different architectures and can be coupled with different products. But both use the notion of declarative business rules and a business rules 'engine' to drive the development process and permit automatic regeneration of application components based on business rules.

Vision Builder uses three views, or logical layers - presentation, business rules, and data - to build multitier client/server applications. The presentation layer uses Visual Basic or Vision Builder's own screen painter. Business rules are defined as triggers and stored procedures and are imported from CASE tools, or as declarative statements captured in Vision Builder's Business Rules Designer. The Designer facility captures rules in an English-like format using a set of tabbed pages based on validation rules (domain checking on columns), derivation rules (calculations), relationship rules (referential integrity enforcement) and conditional action rules ("if" checks or circumstances for executing stored procedures). The data layer is defined by the database structure which can also be imported from a CASE tool or generated in the product itself.

The Vision Builder repository is built on MS Access. Business rules located on the client are implemented as Visual Basic code or Java-based HTML. The middle tier is generated as Visual Basic and C++ code. Business rules on the data layer are implemented as triggers and stored procedures using SQL Server or Oracle on the server.

Usoft Developer uses business rules as the central paradigm for all aspects of the application development lifecycle. Using a data model and declarative business rules written in Ansi standard SQL, Usoft Developer builds multitier, distributed client/server and Web-based applications. In addition to partitioning capabilities, Usoft also exploits the use of a centralized repository to control the application. The repository is the source for generating data schema, business rules enforcement and presentation layers.

While Vision Builder uses database stored procedures and triggers, Usoft Developer uses a 'rules engine', which generates application code for all tiers. Create - read - update - delete (Crud) - related rules, including referential integrity, are generated by the engine. The engine uses the rules from the repository as a parameter for controlling the run time environment. All business rules are added, modified and retired only through the centralized repository.

"It's not another tool, it's another way of doing work", says Bob Brown vice president for business improvement and I/S at Burlington Klopman Fabrics Division, a division of Burlington Industries, Greensborough, N.C. The division is using Usoft to migrate code into flexible, multi-tier client/server applications.

Usoft Developer is changing the way Burlington Klopman builds applications, not only by focusing efforts on up-front data modeling and business rules analysis, but also by shaving months-long projects into weeks. "The business rules are not old and obsolete. The processes in the systems that use the rules are what are old and obsolete. Defining business rules does not mean we are going to do business different. That's a fallacy. Usoft lets you re-architect systems and then modify how you do business, if you want to," said Brown.

Burlington Klopman's approach for extrapolating business rules with business users called first for creating an English language process flow model of the current system. I/S determined the business owners of the rules defined in the flows and then validated the rules by implementing them in Usoft Developer prototypes. Teams of three to five developers work in 90 day chunks to develop applications. "You need to want to make the paradigm shift to business rules", said Brown. "This means you do not want 'stovepipe' applications to spread throughout the enterprise."

The Texas Department of Health's Vendor Drug Program built it's Pharmacy Rebate Information Management System using the Usoft tool. The system handles pharmacy rebate billing, collections, and data query and analysis for statewide usage of prescription medication by Medicaid-eligible recipients.

Usoft Developer was picked to respond quickly to changing policies and guidelines from the federal government, state legislature and other agencies, according to Andy Vasquez, the health department's I/S Manager.

Development began by validating the conceptual design in JAD sessions with customers. The data model was designed using Usoft Developer's data modeling facility. Functional requirements in the sessions were recorded using a word processor. Subsequently, the business rules were extracted from the functional requirements, validated with business users, and used as the basis for writing the business rule in Usoft Developer. The functional requirements also served as a basis for determining test criteria. "Using Usoft's iterative methodology, the migration of rules to working prototype was quick. The end-users were very responsive to this approach", said Vasquez.

Moving to business rules

Business rules are categorized in different ways (See Table 2). There is also no standard for expressing atomic business rules. "I believe a taxonomy is not as important as business rule grammar", said Northern Lights' O'Neil. "The key to this grammar is rigorous definitions. We are working to bridge that gap by building a tool that grammatically represents declarative business rules. This way, business people can directly modify a rule which is then loaded into a business rule repository."

Regardless of which categories or formalism is used, discovering and eliciting business rules be integrated into the system development lifecycle. Business rule enthusiasts have different approaches for weaving business rules into the process. "I knew the industry needed a proof of concept that it is possible to state rules independently of how or where they are enforced. Now there are tools available to implement business rules. The next step is to have a business rule methodology", said Ross.

Business rule analysis or modeling, especially when coupled with a business rule automation tool, accelerates the software development lifecycle. "Business rule modeling alone helps to 'cut to the quick' in analysis, reduce business ramblings about requirements into meaningful English language statements that are easily verifiable by business people," said O'Neil. "If done properly, it will extend requirements gathering forever, so systems are always up to date and flexible. In this way, applications change instantaneously with changes in business rules ".

Warren Keuffel is editor Business Rules Alert! (See Reference 6) newsletter contends: "If you're going to build a system using business rules, and if you're going to use a tool that supports business rules, requirements traceability is an absolute necessity. If you follow half-baked methodologies that merely pay lip service to business rules, or employ tools that add business rule support in the superficial manner, you've lost the game before you've started, and 'business rules' can become another almost-meaningless buzzword, such as object orientation and groupware".

Making the transition to a business rule approach requires business commitment. Business people are responsible for defining and maintaining business rules. Ironically, when defining business rules in facilitated sessions, I have discovered that business people themselves often have a conflicting or foggy concept of the myriad of rules that govern their business. (See Reference 7).

In addition to conflicting and ambiguous business rules, exposing the business thorough business rule modeling inevitably reveals what Barbara von Halle calls "suboptimal business rules". Suboptimal rules provide excellent, but often politically charged, business reengineering opportunities.

These phenomenon are not new to analyst or modelers seeking to define the business with some notation or formalism. Resolving deficiencies and conflicts and making decisions about business rules requires strong and continued business ownership.

Enabling better questions

With opportunities to use a business rule approach, one can experience the power and promise of the business rule paradigm. Working with business and I/S personnel, focusing on business rules is both difficult and complex while simultaneously being a liberating endeavor.

The cliché 'paradigm shift' was introduced by physicist turned historian Thomas S. Kuhn. His notion was that at significant points in time, practitioners of science can undergo a transformation of vision, which can alter or shift an established reference framework. "It is rather as if the professional community has been suddenly transported to another planet where familiar objects are seen in a different light and are joined by unfamiliar ones as well", said Kuhn. (See Reference 8).

Business rules are really a familiar concept. When viewed then from a different perspective, however, one can center them and then bind them with existing elements and goals of software applications - separation, flexibility, business ownership, model-driven development, reengineering, rapid development, and enforcement. Business rules can now be seen in a new light. Paradigm shifts do not necessarily provide new answers. They represent new frameworks which enlighten our existing notions and help us to raise better questions. In this sense, a business rule approach enables better questions, a redefinition of application design constructs and the eliciting of business requirements in new and innovative ways.

Sidebar 1:

Six Categories of Business Rules

Term Definitions:

A term definition is a word, phrase, or sentence(s) which has a specific meaning for the business. Since definitions describe how people think about things, they establish a category of business rules. Terms that become entities/objects (or tables on a database) or attributes/properties (database columns) may be written as '<term> is defined as…'.

Facts Relating Terms:

Asserting an association between two or more terms is a 'fact relating term'. Facts connect things in the business. How terms like 'vendor' and 'supplier' related to each other in the phrase "vendor supplies material" and "each supplier must have an address" are business rules. Facts cause a business term to take on roles in the business (the vendor takes on the role of supplier). Expressions like 'x <connecting verb> y'; 'k contains z', 'y is a type of l' are expressing facts relating terms.

In data models (and the structural components of class models), facts are represented as associations, subtypes (subclass), roles, aggregates, and attributes.

Constraints:

Constraints are circumstances under which a transaction, decision, action, or business occurrence should be aborted, incomplete, rejected, or undesirable. They are rules that protect the enterprise from some sort of harm. For example "do not submit an order where the total number of orders for a vendor exceeds the allowable amount of orders for the country" or "each order must placed by one and only one customer".

Some constraints are detected by English expressions such as: 'must', 'must not', 'if', 'only if', 'only x can <action>', 'only when <time-defined> do <action>'; 'only <term> may do <action>'. Constraints that revolve around protecting the integrity of data/class/objects in the enterprise are also business rules. These constraints are detected with English expressions such as: 'must <association verb> at least one'; 'must <association verb> [one or] more than one'; 'must <association verb> one and only one'. These latter integrity constructs can be represented on a data/class model.

An enterprise may develop, agree, and standardize on the same definitions and facts, but have different constraints in different locations.

Action Enablers

Circumstances that enable or trigger the organization to take action are action enablers. For example, "if a vendor delivers material on time for a 6 month period, then establish special payment arrangement with that vendor"; "when inventory reaches at or below the predetermined reorder point, automatically issue a purchase order to replace that inventory". Expressions such as: ' if <condition is true> then do y'; 'when if <condition is true> then automatically do <action>' indicate action enablers.

Derivations:

Whereas constraints stop the business from doing things and action enablers trigger actions, derivations enable the organization to know new knowledge from existing knowledge through explicitly defined algorithms. They are spotted by English expressions such as: 'x is calculated from z'; 'y is summed from a + b'; 'm = total n/j'.

Inferences

Inferences are rules which define how knowledge or information is transformed into another form, often taken from non-numeric data. An example of an inference might be: 'if a vendor has provided more than x material, they are automatically a special status vendor'; ' 'cost of material' is also considered the 'vendor charge' '.

Inferences may be spotted with phrases such as: '<term> is inferred from <condition>'; 'when <condition1 is true>, then <condition2 is [not] true>'; '<condition> is automatically considered <term>; 'when <condition> is true then <term> is considered also {<term> or <condition-true>}'; ' if <condition is true> then it also means {<term> or <condition-true>}'.

source: Ellen Gottesdiener, EBG Consulting, Inc.

Sidebar 2:

Procedural vs. Declarative

Procedural logic has control flow built into the code. Declarative rules, alternatively, remove the control flow and sequencing - the *how's* - from the program.

Consider the procedural logic for calculating a total sale in a book store. The program will test the book type for a variety of types (sale, hardcover, mass market, etc.) then store an interim cost in memory based on the book type. It then tests for any customer discounts (volume, student, preferred card, etc.) and applies the discount to the interim value. Next, the program totals customer sales and gives a message to the sales clerk that if the customer buys one more book, she is eligible for a free one from the sales rack. The procedural logic continues, prescribing tests and actions in a sequence, including updating database entries, logging activities, and notifying end users. Languages such as C, COBOL, C++, Pascal, etc. are procedural.

Declarative statements do not care about how the rules is enforced. Languages used in rule-based systems, such as Prolog, express rules in a declarative fashion. They represent 'what is true'. Declarative statements require one-third to one-sixth the number of statements to represent rules than procedural logic.

source: Ellen Gottesdiener, EBG Consulting, Inc.

Sidebar 3:

Rule-based Systems

The broad field of artificial intelligence (AI) calls for computers to do tasks that require human intelligence. AI marries the computing field with other disciplines such as cognitive psychology, philosophy, logic, linguistics, and even neurophysiology. There are different constructs and mechanisms, such as rules, to represent the way our minds work, rules being one of them. One type of AI application is a the *expert system,* or knowledge-based systems.

Most commercial expert systems are rule-based systems, which capture the knowledge and reasoning of a human expert and stores it in the form of rules. A rule-based system is composed of a *knowledge* or *rules base,* an *inference engine*, and a set of *case-specific data.* (See Figure 2). A rule in one of these systems normally consists of 'if then else' conditions on the left side and actions on the right side of the statement, such as "If x is book and x is listed on NY Times bestseller then cost is discounted'. Each rule consists of a premise (x is a book and x is listed on NY Times bestseller) and a conclusion (book is discounted).

The rule language is typically declarative (non-procedural) - without implied order or flow of control, as the actual processing of the rules is performed by a separate component, the inference engine (See "Procedural vs. Declarative"). The inference engine uses optimized algorithms such as RETE or TREAT to evaluate the conditions of the rules in the rules base, determine which ones are eligible to fire at any point in time, and in cases where there are multiple rules eligible to fire, it determines the order of firing.

Rule based systems emerged in the early 1970's and have become widespread since commercial products became available in the 1980's. In addition, points out Margaret Thorpe of Tangram Washington, D.C. consultancy, a number of the techniques being used for efficient rule processing in the area of active databases (ones that support general integrity constraints and triggers) actually originated in the early AI rule-based systems.

Rule based systems use two techniques, forward chaining and backward chaining.

Forward chaining, or *data driven reasoning,* draws new conclusions from existing data, adding these conclusions to working memory. This approach is most useful when you know all the initial facts, and want to find out what new facts are true. For example, using the facts that x is a book and x is on NY Times Bestseller, deduce that x was discounted.

Backward chaining, or *goal driven reasoning,* is effective when you know what the conclusion might be, or have some specific hypothesis to test. You are seeking the truth value of that goal or hypothesis. For example, the goal might be to prove that x was discounted by examining if x was a book and whether x was listed on the NY Times Bestseller list.

Examples of uses of expert systems:

Industry	**Use**
Construction	• cost control and financial planning of costs involved in construction, planning, and maintenance of commercial and residential buildings • geotechnical data interpretation; determining suitability of soil properties in design based on soil test results and descriptions
Civil Engineering	• power usage/demand, of hydroelectoric plant operation • determining dam safety based on signal traces from the dam • diagnosis of fault conditions in power stations] • noise signals from nuclear reactors
Medical Diagnostics	• interpretations of signals from respiration monitoring • interpretation of acoustic signals from fetal heart ○ interpretation of capnograms (carbon dioxide waveforms) of anaesthetized patients)
Insurance	• determining insurance risk from actuarial data • commercial underwriting
Airline Maintenance	• parts specification • installation timing, priorities
Investments	• stock, bond, money-market, market indices data

Loan	• loan risk factor, underwriting
Credit	• credit checking, detecting stolen credit cards
Legal	• risk assessment in securities law
Manufacturing logistics	• new product planning based on bills of materials and allowable component configurations
Retail	• spending patterns

source: Ellen Gottesdiener, EBG Consulting, Inc.

Sidebar 4:

Fannie Mae's Karma

Helps to Get to Rules

The Federal National Mortgage Association (Fannie Mae), the nation's largest source of conventional home mortgages, turned to business rules to build a critical application for selling and servicing mortgages is critical to their business. The Knowledge Acquisition and Rule Management Assistant (Karma) application allows policy changes to be implemented quickly and provides direct ownership and management of those policies to business users. The application consists of four components: a business rule specification language, Karma, a Business Rule Server, and a data translator.

Karma developers created a business rule specification language -- a grammar-based representation for specifying business rules. In this language, business rules are of the form IF…AND…THEN…. They consist of clauses and each rules has a right-hand side and a left-hand side component. The left side represents a set of conditions and the right side consists represents a single constraint that must be met when all of the conditions on the left side are true (e.g. "If Lien Type is First Mortgage and Mortgage insurer type is not Federal Government then Current Loan-to-Value Ratio must be less than or equal to x%"). Consequently, each statement declaratively states a business rule.

Karma includes a GUI Rule Editor, underlying databases that support the rules, and a code generation component. Business users interact with the GUI Rule Editor to define new rules and modify existing rules or rule properties. Karma performs consistency checking to ensure that rules are consistent with other existing rules. Executable code is generated, readable by the ART*Enterprise expert system engine from Brightware, Inc., Novato, Calif., from the rule representation in the business rule database. Then the rules become available, in executable form, to the Business Rule Server component of the application.

The Business Rule Server component, a client/server application, makes the business rules available to other applications. The other applications, acting as clients, request compliance checking services of the Business Rule Server. Any application using a Fannie Mae business policy can send loan data to the Server for validation and, if applicable, receive violation notification. The Business Rule Server uses control

structures written in ART*Enterprise and C, the executable business rules, an remote procedure call (RPC) and the data translator facility. Loan data is passed to the Business Rule Server, translated and mapped to the knowledge base. This causes the business rules to fire, and any violations are passed back to the application.

The data translator maps application data models to the data models on which the business rules are based, permitting the Business Rule Server to be independent of any of its data sources or requesters. It resides between the server layer of the Business Rule Server and clients. It contains data translation rules to map the source data model to the target (Business Rule Server) data model.

Karma and the Business Rule Server have streamlined the loan delivery process to such a degree that it may soon become a "lights-out" process. "Business people use it to define the rules", said Thorpe. Business rules can be entered, checked for consistency and redundancy and then modified in minutes. Implementing the business rule in the production Business Rules Server, however, still takes a few days as they must go through User Acceptance testing and the standard production migration processes.

source: Ellen Gottesdiener, EBG Consulting, Inc. & Margaret Thorpe, Tangram.

Sidebar 5:

Merging data and object models with business rules

Since objects are responsible for their own data and behavior, business rule in most object-oriented methodologies are implied; they are not distinct classes. Additionally, business rule enforcement in an object-oriented application will often require collaborating objects which will know what objects to interact with to satisfy a rule.

Can data models, object models, and business rule come together? One way to explore this idea is to consider the emerging IDEF1X97 standard. IDEF1X is a data modeling standard used by federal agencies such as the Department of Defense (DOD). Extensions to IDEF1X97 now in development include class/object modeling standards. "We have the concept that an object (an instance of a class) 'has knowledge, exhibits behavior, and obeys rules'" , said Keri Anderson Healy, senior consultant, Model Systems Consultants Inc., Bainbridge Island, WA, who is one of the technical editors of the IDEF1X97 standard. The next release will add the idea of business rule as 'constraints' is being added. "There is not a separate step for listings rules in the form of constraints. They will evolve during modeling sessions" , said Tom Bruce, who chairs the standards committee.

Properties of objects/classes will include attributes, operations, participant properties (arising from relationships to other classes/objects), and constraints. "Some constraints (such as uniqueness constraints) are so prevalent that there is a graphic way to declare them. Most constraints, however, are unique to a particular model. These constraints are named and specified as a property of some object (class)", said Anderson Healy.

The business rule defined by an object's constraint is not independent. It may need to collaborate with other objects. This could result in complex communications amongst objects. "The realization of the constraint may require the collaboration of several objects, possibly of differing classes", pointed out Anderson Healy.

The emerging IDEF1X97 object model, like most object-oriented modeling techniques, subsumes many business rules within the modeling constructs. Many business rules are implied as part of the graphical notation. A business rule might be a property of a class (such as an attribute, association or derived attribute) or be implied in an aspect of the model's structure (such as a subclass).

With an object-oriented paradigm, there are other possible ways to incorporate a business rule approach, including making each rule an object. This is likely to yield significant performance problems. Another approach would be to design artificial objects such as "Service" or "Policy" objects which are delegated the responsibility to enforce business rules.

Should business rules be independent constructs in object-orientation? Depending on your interpretation of object-orientation, the question may be a moot point since there are no independent constructs except objects. In object-orientation, all behaviors and attributes - responsibilities - are encapsulated in a class/object. "I think that many object "modeling" techniques do reflect the notion of 'rule' in some fashion. In a (conceptual) object model, I find that the prevailing debate is not so much whether 'rule' can be represented. Rather, the debate is whether a 'rule' should be represented as an independent (free-floating) construct or whether 'rule' has to 'belong' to some other (business) object/class", said Anderson Healy. "In many object languages, anything that *appears* to be "free-floating" is ultimately attached to the application (itself an object) -- so nothing is really free-floating".

There are other perspectives. Dan Tasker, of Air New Zealand, for example, believes the business rules and objects should be "free-floating in the same way that entities are free-floating. There should be *instances* of business rules which are managed and associated (mapped) to other appropriate things, like entities and operations. These instances of business rules have properties of their own, for example who is the owner."

Business rules are the Achilles' heel of object orientation for business database applications", asserted Ron Ross, an author and business

rules expert.

source: Ellen Gottesdiener, EBG Consulting, Inc.

Table 1:

Representative Sampling of Business Rule Tools

Business Rule-Repository Based Applications Generators

(these tools use a business rules repository to establish and maintain application components)

Vendor Product

Usoft, Brisbane, CA. Usoft Developer

Vision Software, Oakland, Ca Vision Builder

Knowledge-based/Expert System Applications Generators

(these tools use business rules as the core paradigm driven by an inference engine)

Brightware, Novato, Ca. ART*Enterprise

Intellicorp, Mountain View, Ca LiveModel

Intelligent Environments, Burlington, MA Applications Management

Neuron Data,?? Elements

Platinum Technology, Oakbrook Terrace, Ill. AionDS/Rule Server

Applications Generators that use Business rule specification/Modeling

(these tools use business rules as one of several key concepts for modeling and/or specification of the application software)

Antares Alliance Group, Dallas, Tx. ObjectStar

Dynamics Research Corp., Andover, Ma. Visual Magic

Magic Software Enterprises, Inc., Irvine, Ca. Magic

Sapiens, Durham, NC ObjectPool

Texas Instruments, Plano, Tx Perfomer/Composer

Business Rule Extraction of Legacy Applications

(these tools "mine" legacy code for business rules and present the rule in tabular and/or visual format for reusing, re-engineering, and discovering business rules in existing applications)

ReGenisys Corporation, Phoenix, AZ rulefind:R and analyze:r

McCabe & Associates, Columbia, Md. Slice Tool

Case Tool for Business Rules

(this tool use an English language business rules formalism for generating software models and software artifacts (e.g. database schema)

Asymetrix, Bellevue, Wa. InfoModeler

source: Ellen Gottesdiener, EBG Consulting, Inc. **Table 2:**

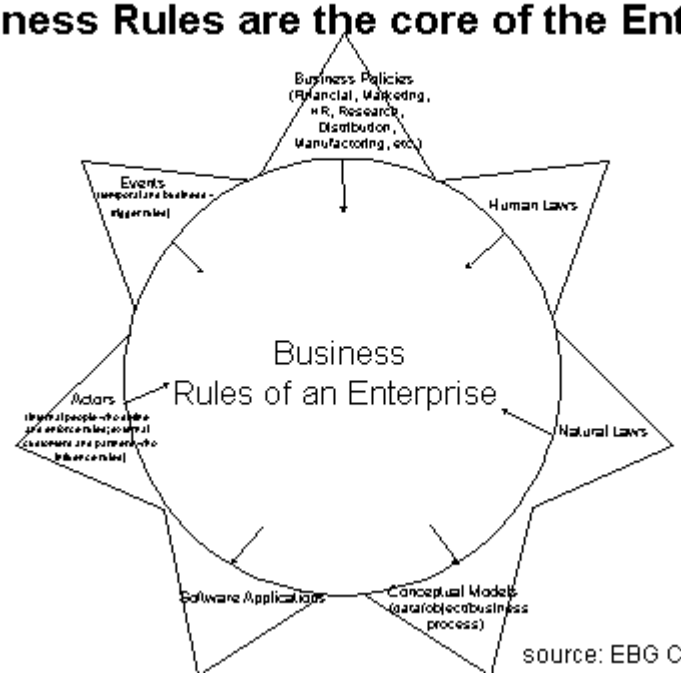
Sampler Business Rules Taxonomies

SOURCE	TAXONOMY
Guide Business Rules Project	• **Structural assertion** (terms, facts) • **Action assertions** (integrity constraints, conditions, authorizations) • **Derivations** (calculations, inferences)
Ron Ross, Database Research Group	• **terms** • **facts** (relationship types, subtyping, attribute**)** • **rules** (computations, rejecters, projectors)
James Odell, independent consultant	• **Constraints:** (stimulus/response, operation, structure) • **Derivations:** (inferences, computations) notes: rules can be global, local, and or temporal
Tom Romeo, IBM	• **structural** (relationships, domains, cardinality, optionality) • **behavioral** (pre-conditions, post conditions, derivations)
Margaret Thorpe, Tangram	• **definitions** • **basic integrity constraints** • **general declarative constraints** • **procedural constraints** • **inferential** • **derivation**
Barbara von Halle, Knowledge Partners	• **definitions** • **facts** • **constraints** • **derivations** • **inferences**
Usoft Corporation	• **restriction** (constraints, domain) • **deduction** • **behavioral** • **representation**
Dan Tasker, Air New Zealand	• **action restricting** • **action triggering** • **constraint**

Brightware	• **business** • **policy** • **workflow** • **decision heuristics**
Vision Software	• **validation** • **derivation** • **relationship** • **conditional actions**

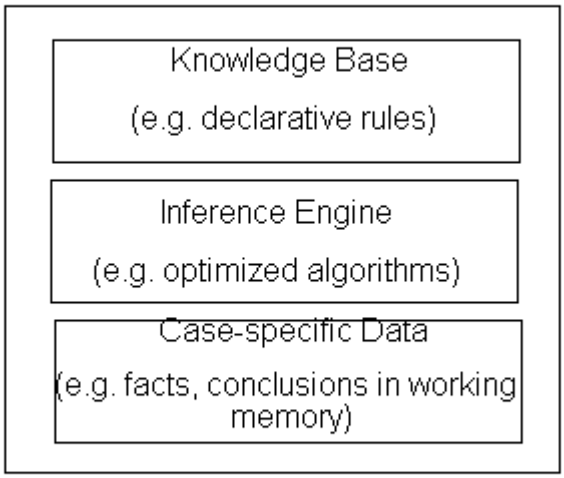
source: Ellen Gottesdiener, EBG Consulting, Inc.

Figure 1:
Business Rules are the core of the Enterprise



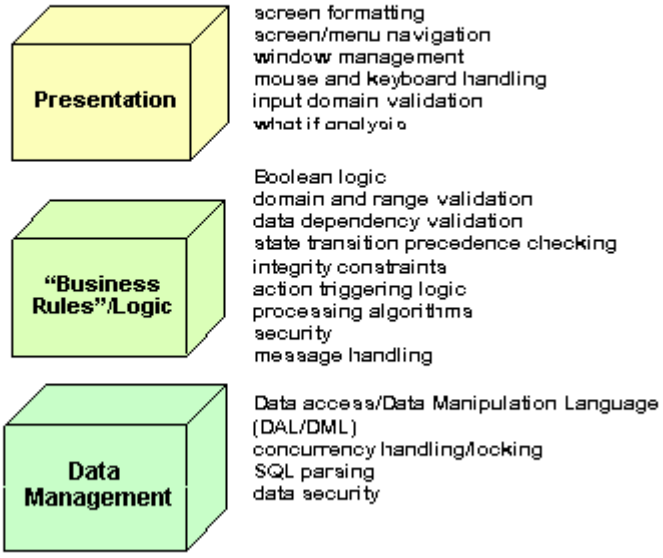
source: EBG Consulting, Inc.

Figure 2:
Expert Systems: Architecture



source: EBG Consulting, Inc.

Figure 3:
3-Tier Logical Client/Server Architecture

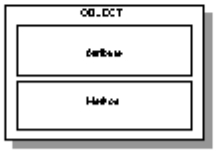


source: EBG Consulting, Inc.

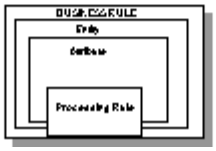
Figure 4:
Paradigm Shifts

Structured Analysis

Information Engineering



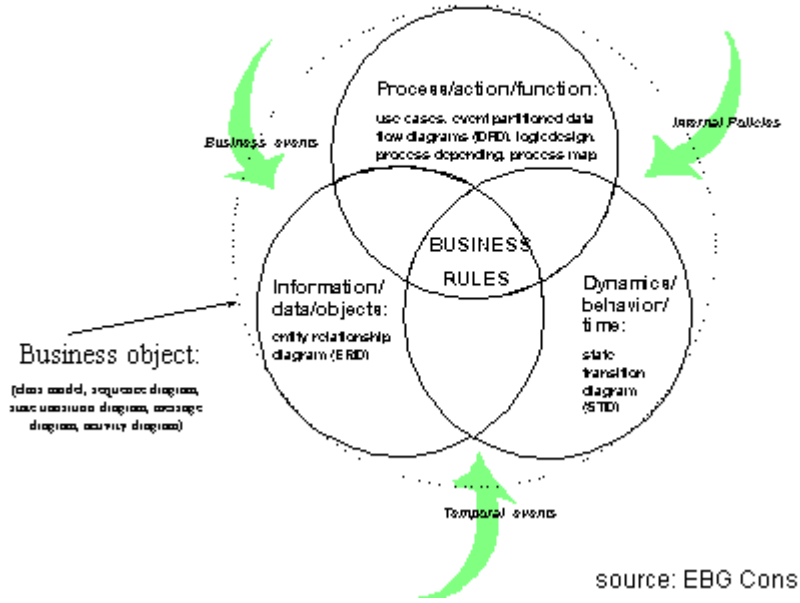
Object-Oriented Analysis/Design



Business Rule Analysis/Design

source: Terry Moriarty

Figure 5:
Business Rules: the center of modeling points-of view



source: EBG Consulting, Inc.

References

1 Guide Business Rules Project, Final Report, 11/95. Go to: http://www.businessrulesgroup.org/brgactv.htm

2 Ross, Ronald, The Business Rule Book: Classifying, Defining and Modeling Rules, Database Research Group, Boston, MA, 2nd edition, 1997.

3 Bruce, Thomas, A., Designing Quality Databases with IDEF1X Information Models, Dorset House, 1992.

4 Tasker, Dan, The Problem Space: Practical Techniques for Gathering and Specifying Requirements using Objects, Events, Rules, Participants and Locations, 1993, electronic book, self-published by the author: Dan Tasker, 100241.2337@compuserve.com

5 Gottesdiener, Ellen, "RAD Realities: Beyond the Hype to How RAD Really Works", Application Development Trends, August, 1995, vol. 2, no. 8.

6 Business Rules Alert!, Database Research Group, Inc., 617.227.2583, (out of print as of 6/98).

7 Gottesdiener, Ellen, "Facilitated Business Rule Workshops: 12 Guidelines for Success", Database Newsletter, Jan/Feb, 1997, vol. 25, no. 1.

8 Gleick, James, "The Paradigm Shifts", New York Times Magazine, p. 25, December 29, 1996

9 Sobieski, J, Krovvidy, S, McClintock, C., and Thorpe, M. "KARMA: Managing Business rules from Specification to Implementation", paper presented at American Association of Artificial Intelligence Conference, Innovative Applications of AI track, Portland, Or, 7/96.

Business Rule Summit, February 26-28, 1996, Miller Freeman Inc., 600 Harrison Street, San Francisco, CA 94107.

Database Newsletter, Database Research Group, Inc.,
617.227.2583,http://www.brsolutions.com/newsletter.html

Gottesdiener, Ellen, and von Halle, Barbara, "Breaking the Rules On Purpose: (An Introduction to the 'whys and hows' of facilitated rule breaking)", Database Programming & Design, September, 1996, pp.9-12, vol. 9. No. 9

Hurwitz, Judith, "When Rules Meet Development", DBMS Magazine, January, 1997.

Kara, Dan "Rules-based tools: business rule specification is job 1", Application Development Trends, November. 1996.

Moriarty, Terry, "The Next Paradigm", Database Programming & Design, February, 1993.

Odell, James, "Business Rules", Object Magazine, republished in Wisdom of the Gurus: A Vision for Object Technology, Charles Bowman, ed., Sigs, 1996.

Ross, Ronald G. and von Halle, Barbara, The Business Rules Approach, Addison-Wesley Publishing Company, to be published in 1997.

Seybold, Patricia, "Start Your Business Rules Engine", Computerworld, December 9, 1996.

von Halle, Barbara, "Data Architect" column covering business rules, Database Programming & Design, Miller Freeman, http://www.dbpd.com

Wright, David, "Business Rules", Data Management Review, December, 1996, http://www.dmreview.com

Ellen Gottesdiener is President of EBG Consulting, Inc., facilitation, training, and consulting company. She provides facilitation services to business and technology customers and conducts training workshops on modeling techniques. Ellen has authored numerous articles and contributed to several books. You can email Ellen at ellen@ebgconsulting.com or visit her web site at http://www.ebgconsulting.com