IDIOM
PURE DECISIONING

Decisioning
the next generation of Business Rules

Executive Summary:

Today's constraints focused business rules concept has served its proponents well for the last 30 years.
However, as hands-on business ownership of decision automation gains ground it is becoming apparent that fully automating a decision making process requires the current business rules concept to be expanded.

Modern commercial decision making requires:

- **transformation and/or derivation of the supplied transaction data to occur within the decision making process;**
- **production of a wider range of data types than the simple 'true/false' statements being delivered by the current business rules concept;**
- **that generation of all values occurs within a complete and orchestrated set of decisions, which must then execute as a discrete transaction to ensure that the business data and all surrounding systems retain their consistency and integrity.**

These capabilities cannot be met using a constraints oriented concept of business rules.
An expanded business rules concept needs to be adopted.
We have described this concept as '**decisioning**' – the discrete and systematic discovery, definition, assembly and execution of decisions in their broadest sense.
Decisioning as described herein is already implemented on an industrial scale across a wide range of business sectors.
It is significantly and demonstrably improving business efficiency and agility, compliance, and alignment of systems and strategy; while reducing cost, risk, and time to develop and evolve.

Decisioning: the call to action

When this author used Factiva and Google to look for the term 'decisioning' in 2003, there were zero uses of the word outside boxing. Ten years later, there are hundreds of thousands of hits on Google for 'decisioning', mostly related to the IT concepts of decision automation and decision management.
A quick survey of the URL's returned by Google suggests that most of the current cohort of decisioning vendors and commentators have a historical connection with 'business rules', leading us to ask the question:

Is decisioning a synonym for business rules;

is it an evolution of business rules;

or is it something else entirely?

A keynote speaker at the 2004 EuroBizRules conference in Amsterdam observed that use of the Rete algorithm was the defining characteristic of a Business Rules Engine [BRE]. If nothing else, the statement emphasised the important position of Rete in business rules group-think at that time. While the three non-Rete business rules vendors present might have disagreed, there is an argument that the statement is true by definition: If it can be processed by an engine using the Rete algorithm, then it qualifies as a business rule; and if it processes business rules, it qualifies as a business rules engine.
With hindsight, this circular logic makes sense. The original Rete algorithm was not concerned with business rules *per se*; it originated as a pattern matching algorithm that was focused explicitly on 'many patterns/many objects'. Business rules that can be processed by the Rete algorithm are comprised of one or more condition statements that collectively resolve to one or more conclusions. The set of condition statements that lead to each conclusion constitute a 'pattern'.
The evolution of Rete was accompanied by a reframing of patterns into 'business rules', leading to wider acceptance outside of the then struggling Expert Systems fraternity, and ultimately its successful inclusion in the leading BRE's. However, for a business user today who is trying to codify business decision making, Rete style 'business rules' may seem somewhat limiting.
The Business Rules Manifesto[1], published by the Business Rules Group, summarises the business rules concept as practised by most BRE vendors (particularly those who have implemented Rete) and by the business rules community at large. If we look at the manifesto, we can highlight some of these limitations.

First among them is the fundamental assertion that business rules are constraints, as the following clauses from the manifesto imply:

(2.1) Rules are explicit constraints on behavior and/or provide support to behavior.

(7.1) Rules define the boundary between acceptable and unacceptable business activity.

(7.2) Rules often require special or selective handling of detected violations. Such rule violation activity is activity like any other activity.

(8.1) Rules are about business practice and guidance; therefore, rules are motivated by business goals and objectives and are shaped by various influences.

(8.2) Rules always cost the business something.

(8.3) The cost of rule enforcement must be balanced against business risks, and against business opportunities that might otherwise be lost.

That rules are 'constraints' is consistent with other manifesto clauses that limit the rules to predicate logic and truth values (i.e. true/false). Again from the manifesto:

(5.3) Formal logics, such as predicate logic, are fundamental to well-formed expression of rules in business terms, as well as to the technologies that implement business rules.

(6.4) Rules are based on truth values

[1] http://www.businessrulesgroup.org/brmanifesto.htm

We have suggested that the above characteristics echo the Rete architecture. Not surprisingly then, Rete is inferred as a desirable execution strategy by two clauses which seem to discount standard procedural programming as a plausible BRE architecture:

(4.3) A set of statements is declarative only if the set has no implicit sequencing.

(6.2) Executing rules directly – for example in a rules engine – is a better implementation strategy than transcribing the rules into some procedural form.

In presenting these characteristics, we are not suggesting that business rules as so described are in any way wrong or inadequate – they are what they are. The point of this article is to suggest that business led automated decision making cannot be fully achieved using a set of rules and a BRE that only addresses constraints; any assumption that constraints are synonymous with business decision making is incomplete. Constraints are only part of the process of actually making decisions on transactional data, so that the business user trying to automate their decision-making using a constraints only approach is usually required to revert to standard programming, and the SDLC that it implies, for the balance of the decision-making task.

It is sometimes assumed that Rete was successful because it had a unique ability to efficiently process data through large sets of rules. In fact the power of the Rete algorithm was not in processing inputs through to a conclusion, which is easily achieved by other programming approaches, including procedural programming; it's real and self-proclaimed novelty was as a state-full machine that efficiently handled a large number of discrete inputs that could change independently.

As Charles Forgy's original paper[2] stated in its abstract:

"The Rete Match Algorithm is an efficient method for comparing a large collection of patterns to a large collection of objects. It finds all the objects that match the pattern. The algorithm was developed for use in production system interpreters..."

And in the final paragraph (abridged) and final sentence (unabridged):

"Certainly the algorithm should not be used for all match problems; its use is indicated only if the following... conditions are satisfied. ... Since the algorithm maintains state between cycles, it is inefficient in situations where most of the data changes on each cycle."

It is managing a large number of separate inputs that change slowly and independently (relative to each other) that is the important point of difference for Rete.

You might wonder how many commercial rules problems share these characteristics, thus playing to Rete's advantages. Having not encountered one, this author would venture – not many! Modern commercial systems are transaction based, which infers a new and complete set of data every time the 'BRE' is called – exactly the class of problem that Forgy described as being contra-indicative of Rete. And the prevailing architectural trend is towards bigger and bigger transactions – as we transition from the legacy 'system transactions'[3] towards a more complete and all-embracing business transaction (or 'service'), the size of the transaction 'unit of work' is growing – hundreds of thousands of data nodes in a modern commercial transaction dataset is now commonplace, all of which 'changes on each cycle'.

The Rete approach is at a relative disadvantage when processing these transactions – and it is these transactions that are the 'bread and butter' of modern commercial systems. If Rete does not have the dominant edge it was once perceived to have had in processing these transactions, should we re-think 'business rules', which evolved in a Rete dominated world and which still show signs of this legacy?

Perhaps this is why some years later an analyst from the same organisation as our keynote speaker above spoke to this author about a survey that he had conducted on BRE vendors. He was surprised to note a clear trend – new 'BRE' vendors were only offering sequential processing engines, while existing Rete based vendors had also introduced sequential processing options. Perhaps not so surprising when large, real world transactions are well-suited to a sequential processing model rather than a Rete style algorithm.

There are further reasons why a constraints oriented business rules approach is sub-optimal for solving decision automation problems in the context of modern transaction processing.

[2] "Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem", Charles L. Forgy, Artificial Intelligence 19 (1982), 17-37.
Available here: http://www.csl.sri.com/users/mwfong/Technical/RETE%20Match%20Algorithm%20-%20Forgy%20OCR.pdf

[3] "indivisible and irreducible" – the minimum process step.

Service oriented architectures are inexorably trending towards separation of 'doing' components from 'deciding' components. The 'doing' components (processing data and messages to and from external devices and services) tend to be generic and multi-use – ideally, these components should be driven by meta-data and should not be aware of the business content of the messages they are processing (allowing them to be commoditised and outsourced).

The truly proprietary code, the code that captures business knowledge and IP, is being increasingly concentrated in the 'deciding' components. This is not an anomaly; only the decisions require proprietary business knowledge. What we are witnessing is a continuous learning process that progressively distils and concentrates proprietary knowledge and experience into a new higher order system component – the decisioning component.

Further, as the business 'subject matter experts' [SMEs] focus on these 'deciding' components in their systems, they start to think in terms of 'zero touch' or 'straight through processing', which further expands the size and scope of each business transaction.

These assertions are supported by decades of decision oriented development by this author and others. Our experience suggests that the primary purpose of decisioning is to create business value by applying core business knowledge to the subject matter at hand, a process which includes but is not limited to the 'detect and constrain' function of traditional business rules.

In using the term decisioning, we are asserting the idea that decisions are the definitive means by which an organisation changes the state, and therefore the value, of any entity under its control[4]. These 'entities' are real-world things that represent the value proposition for any organisation – by way of example, for a hospital it is patient episodes; for an insurer it is policies and claims; for a government department it might be entitlements, benefits or tax claims; for a superannuation (pension) fund it is member accounts; and so on. These value creating state changes are governed by a set of policies[5] including:

- **internally defined policies (business or management policies);**
- **and/or policies agreed with third parties (contracts);**
- **and/or imposed policies (regulations).**

These formal policy statements are the definitive 'sources of truth' for the business 'knowledge' that prescribes how operational business decisions are made, and therefore how the business will respond to everyday events, particularly in terms of the all-important entity life-cycles.

Codifying this knowledge is a concept that expands on the idea of 'business rules as constraints' by a wide margin. Our experience is that codifying this form of knowledge requires us to expand the concept of rules beyond simple 'truth statements' to embrace a much wider spectrum of assessments; and we need to be able to make these assessments from different policy driven viewpoints.

We need to dynamically transform the input data that is supplied to the decisioning process into terms that align fully with the relevant policy statements, when and as the statements need to be applied. This may require us to derive calculated amounts, buckets, averages, caps, lists, pivots, and all sorts of other derived forms, including derived forms of collections. For the sake of clarity, we need to create new data, and new constructs to hold the data, on-the-fly and within the decision making process itself.

For example, if a policy statement said: 'If the total of all payments is greater than $1000 and any individual payment is above $100, then refer it for approval', then we need to create a) a total; and b) a new indicator derived from the collection of payments that tells us whether there is a payment above $100. Two new data elements, neither of which are directly provided in the input data. These data elements are not an inherent part of the transaction data – they are implicitly defined by the relevant policy statement.

This sort of data manipulation is **not** like the 'doing' components previously discussed because it is not generic. It is an indivisible part of the rules specification, because it is only required in the context of the rules; this derived data is an inherent part of the specification of the rules as defined by policy statements. Knowing these derivation and transformation requirements is as much a part of understanding a decision-making problem as is applying constraints.

And what is this 'decision-making problem' that we are referring to? It is the reduction (through appropriate analysis) of the relevant policies into a series of discrete outcomes (which we now refer to as decisions[6]) that collectively achieve the business **purpose** when responding to an event.

[4] For more discussion on this topic, please see the author's prior articles on the Business Rules Journal [http://www.brcommunity.com/b326a.php] and Modern Analyst [http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/1354/Requirements-and-the-Beast-of-Complexity.aspx]

[5] http://www.thefreedictionary.com/policy A plan or course of action, as of a government, political party, or business, intended to influence and determine decisions, actions, and other matters

[6] http://www.thefreedictionary.com/decision "The act of reaching a conclusion or making up one's mind. A conclusion or judgment reached or pronounced". In fact, 'rule' is probably a more correct English term for a policy defined outcome, and 'decision' a more correct term for a constraint driven outcome, however, that opportunity has long gone!

There are three classes of decisions that operate on the **context data** in response to an event. The context data is a) the data that describes the current state of the primary transaction entity; b) any other data harvested from the triggering event; and c) any reference data that is required for the decision making (e.g. product parameters, channel data, etc).
The three classes of decisions that use this data are:

Precursor decisions – these decisions create interim working data including:

- Validations – new indicators that describe the quality (concise, consistent, correct, complete) of the supplied context data;
- Working data variables – dynamically generated data that is derived from the validated data, to be consumed by the primary and meta decisions. May be simple, such as calculating age or days since the last event, through to complex, such as dynamically generating a set of standardised data containers, and then apportioning a collection of values across them for comparison or totalling.

Primary decisions – these decisions derive permanent new data in accordance with the policies that define the valid states of the entity, including:

- State changes, for instance 'approved', 'referred', 'declined';
- Value calculations, including costs, charges, taxes et al;
- Other business terms or constraints that may (now) be added or removed as required by the context.

Meta decisions – these decisions directly target the external environment, and are used to align all external functions and states with the new state of the entity, including:

- Workflow – these decisions direct the workflow manager that is controlling **this** transaction, which then drives all downstream activities;
- External system values – new data calculated for the benefit of other systems, for instance accounting entries for the GL, alerts for audit, bring-ups for human or automated future actions, emails and communications to third parties, etc.

The data derived by the Meta decisions are more or less private between the decisioning component and the targeted external receiver – they need only be read and understood by the external function involved.
Note that these decisions are not limited to 'truth values'; we must also 'decide' continuous variable values like amounts and dates, and strings. And the entire set of decisions must work as a single unit-of-work to ensure the integrity of not just the entity data, **but the entity data and all surrounding systems**. This complete set of decisions, which collectively achieves the purpose and fully responds to the initiating event, can be collated into a model of decisions – a decision model.

[Note: This term should not be confused with the 'Decision Model'[7] currently being publicly promoted by Knowledge Partners International. The KPI Decision Model is a method for collating constraints oriented business rules to achieve a single outcome (a 'Decision'), and it targets implementation via the traditional BRE; that is, it is a model of the internal rules of a single decision.]

All of which is a problem for the Rete aligned concept of 'business rules', which is limited by its own definition and focus to addressing only the constraints part of the overall decision making requirement, so that it requires the additional power of development and/or process platforms to fully solve the business decision-making problem. Consequently, most 'business rules engines' are developer oriented tools that are increasingly being integrated with business process platforms, or even raw development environments like Visual Studio or Eclipse. Of course, some of these tools also actually allow the business rules expert to change the 'business rules' through a business friendly UI, just like they can change table values. But the advance is incremental at best. Ultimately, the automation of the end to end decision-making is still being achieved through an SDLC that looks very much like it always did; and it still cannot fully delegate the decision-making problem to the business SME.
There is however, an evolutionary path for decisioning tools aside from Rete that does not subscribe to the same limitations, and that is via 'model driven development'.
Model driven development can be described as 100% system generation from a discrete, declarative, and validated 'model'. It has been a practical reality since the 1980's – for instance, at one time, 60% of all code running on IBM's AS400s (now iSeries) was reputedly generated from one such tool. From this author's own experience with these tools, it was very plausible to completely define business decision making logic, including all required data manipulation, and to 'black-box' it within the overall system model.

[7] http://www.kpiusa.com/index.php/The-Decision-Model/the-decision-model.html

If we look to the model driven development world and extract from it what the business SME would need to fully address the decisioning problem, but excluding the 'doing' components, we can achieve real delegation of the entire decision-making process – we can give the business hands-on control of the 'deciding' components.
A de-scoping of the model driven development approach can be used to provide an **expanded concept of business rules**. We need to de-scope all capabilities that deal with:

- Actually producing or interacting with an orchestrated set of processes (other than as a complete and discrete decision-making component in its own right).
- Interfacing to existing processes, people, or devices.
- Acquiring and/or persisting any artefact.

These excluded capabilities are the key – they are the elements of traditional programming that bind to both technology and platform. Because these are explicitly NOT present in our proposed expanded rules concept, we can deal with the decisioning requirements in a technology agnostic manner. In other words, a platform that excludes these elements by definition represents purely business intent. Of course, it also means that we cannot produce a complete system directly from the requirements that are defined in this manner[8]. However, that is not the objective. The objective is to give the business hands-on control of the 'deciding components' and by implication, control of the entire operational environment through the meta-decisions. This requires that they have sufficient power within the decisioning concept and its supporting tools to transform, derive, and interpret the transaction's context data in any way required without recourse to a programmer. This is analogous to Excel users who can add, change, and delete cells at will, but only within the confines of the spreadsheet. While they can't produce a complete system, they can produce a complete decisioning component.
At the same time, we are seeking to reinforce IT ownership of the 'doing components' – all of the platform and technology dependent components and functions that require competent engineering relatively independently of business subject matter expertise. And of course, the decisioning components need to be wired into the platform and technology for execution within a workflow. 'Outsourcing' of the decisioning components back to the business is an act of delegation on the part of IT, not an act of abandonment.
Perhaps it is time for a couple of business rules[9]:

- If it requires business subject matter expertise and 'final say' authority, and does not require knowledge of technical platforms or capabilities, it should be owned and managed by the business.
- If it requires knowledge of and the authority to engineer and operate technical platforms and their capabilities, and does not require business subject matter expertise or approval, it should be owned and managed by IT.

We have referred to the former as the 'deciding components' and the later as the 'doing components'.
Having conceptually separated the deciding components and given them to the business, we need some special features to help the business exercise their ownership in practice. They need a tool that provides:

- A technology agnostic, self-contained, and safe sandpit in which to build the models of decision-making (analogous to using Excel);
- An IT defined meta data schema that is accessible to the decisioning model so that it can remotely control the 'doing components', and in so doing align all external business systems and components;
- A business defined data schema to support the primary decision making;
- An easy to use interface that will allow SME's who are computer literate (say, to the extent of an Excel user) to transform, derive, interpret and otherwise work with the context data as may be required to achieve the business purpose (but explicitly not programmers – these users do not need and should not have knowledge of the implementation platform or technology);
- A user operable testing environment to allow the same SME's to empirically verify that the business purpose has been met – completely, concisely, consistently, and correctly;
- And it would be nice if it produced some documentation too!

[8] Actually this would be a huge step backward – generating whole systems implicitly outsources IT strategy to the vendor and is very risky for the system owner. Also, the involvement of the IT dependencies implicitly reinstates the SDLC.

[9] We agree these are unclear rules. Then again, the separation of business and IT special interests is also somewhat fuzzy!

We would like to finish this article with some examples of real world problems that demonstrate this concept of decisions as something bigger than the traditional concept of business rules. The following are some real examples of decision-making that have been addressed with 'decisioning' concepts as described in this article, each specified and implemented within a single decision model:

- Entity is **"Superannuation (Pension) Fund Member"**, event is **"Month-end"**: Is all of the member data valid (an audit decision); are there any age or contribution based member adjustments to make; what fees/insurances should be charged, and how much to charge for each; calculate tax for the member; derive high level information for reporting.
- Entity is **"Insurance Policy"**, event is **"Application received"**: Do we have enough information about the risk and is it valid and consistent; will we underwrite the risk, at what cost, and under what terms and conditions.
- Entity is **"Patient Episode"**, event is **"Discharge billing required"**: What recoveries need to be made against what funding contracts given the range of patient conditions found and treated, and the services consumed; which costing and allocation methods are fairest given the mix of conditions being treated, and giving due regard to compliance with the funding contracts; how much will we charge each contract and/or the patient.

Most of these 'decisions' have a 'business rules' component; but they are not decisions that can be made using a constraints oriented BRE acting alone.
We finish with some more 'real world' examples of commercial problems, this time including an outline of the in-stream transformations of the context data required before resolution of the primary decisions was possible. We suggest that the inclusion of these transformations within an over-arching decision model is an important differentiator between 'decisioning' and 'business rules'.

Problem Area	Primary Decision	Problem Resolution
Hospital Billing	How much do I charge the patient and/or others?	A single patient episode may be co-funded through multiple contracts, each of which refers to a different view of the underlying episode data. The data is initially transformed into standardised hourly consumption and costs, including pro-rated overheads. Then the hourly source data is transformed to comply with the chargeable units defined by each funding contract, before being finally valued. Entitlements and overlaps in funding have to be identified and resolved according to each contract's terms (no double dipping). Invoice line items are then generated and totalled and taxes calculated.
Government Education Funding	How much do I pay the school?	The restricted funding available has to be apportioned over the available cohort of eligible children on a best fit basis. Several overlapping funding contracts exist per school, each with different entitlement criteria. Each child must be evaluated for entitlement, then aggregated into sets for funding by each available contract. Contracts are capped on a weekly basis according to different criteria, some cumulatively, some individually. The school would expect the best fit application of children to the available funding for maximum funds. When best fit has been achieved, total all funding into monthly payment for the school.
Accident Compensation	What is the beneficiary's entitlement this week?	All factors including overlapping accident claims, personal circumstances and location, social subsidies and policy criteria, and macro conditions (e.g. inflation) have to be overlaid into daily segments. The actual entitlement is derived from the aggregated situation for each beneficiary calculated on a pro-rated daily basis.
Utility Meter Audit	Do I need to physically visit and audit this meter?	For each meter, pro-rate the actual billing amounts into exact standardised billing periods to allow period on period comparisons. Determine the mean daily variation by location (separate model required for this). Calculate the variation from the mean for this meter and aggregate into moving averages. Then flag meters for review that exceed dynamically calculated variance boundaries for these averages.

Finally, in answer to our opening question,

we leave it to the reader to decide whether decisioning is evolutionary,

or different in kind!

About IDIOM

Established in 2001, **IDIOM** is a pioneer in the development of decision automation concepts and approaches, and in their practical application to the design and development of systems of all sizes. **IDIOM** has applied its "pure decisioning" tools and approaches to improve systems development performance and business agility in projects in Europe, Asia, North America and Australasia, across such diverse domains as insurance/finance, health, municipal, state, and central government, telecoms, utilities and logistics. **IDIOM** leverages this experience in developing and marketing the "**IDIOM** Decision Manager", a purpose built decision automation tool. **IDIOM** Decision Manager is a proven, pragmatic and cost effective tool for capturing, managing, automating and deploying business decision making know-how as described in the Modern Analyst article "*Requirements and the Beast of Complexity*". **IDIOM** Decision Manager is used by business users to define and control intelligent business processes through generation of small footprint, non-intrusive decision making software components. **IDIOM** is a regular participant at the EuroBizRules, International Business Rules, and the Building Business Capability Forums.

IDIOM Products

IDIOM Decision Manager is a tool for the SME and/or analyst to graphically model, test, document and deploy complex business decision-making as fully executable, high performance 'decision models' – without programming!

IDIOM Forms is a tool to define and deploy large, complex Web2.0 forms that are tightly bound to **IDIOM** decision models at execution time, field by field. The decision models are used to apply all business logic, to control workflow, and to dynamically control the form's look and feel. This more technical tool is fully programmer extensible.

IDIOM Decision Tracker is a tool to map MS Word and MS Excel documents to **IDIOM** decision models for full bi-directional traceability between corporate policy definitions and their implementation as **IDIOM** generated decision models.

IDIOM Document Builder is a tool to define, test, and deploy MS Word document specifications, and a document generator that uses those specifications to build transactional MS Word documents under the control of **IDIOM** decision models at execution time.

IDIOM Decision Manager Workbench is a user operable application to acquire decision models and other process components, assemble them into scheduled processes, and to run them on a large scale. It also collects outcomes and supporting information per transaction and/or per process for subsequent analysis and/or action.

Contacts:

Mark Norton +64 21 434669 mark.norton@idiomsoftware.com
General enquiries please call +64 9 6308950 or email idiomsales@idiomsoftware.com

For more information please see our website at www.idiomsoftware.com
IDIOM Limited (2001) is a private company based in Auckland, New Zealand.