Chapter 12 –Business Rules and DROOLS

Katanosh Morovat

Introduction

In the recent decade, the information systems community declares a new concept which is called business rules. This new concept is a formal approach for identifying the rules that encapsulate the structure, constraint, and control the operation as a one package. Before advent of this definition, system analysts have been able to describe the structure of the data and functions that manipulate these data, and almost always the constraints would be neglected.

Business rules are statements that precisely describe, constrain, and control the structure, operations and strategies of a business in an organization. Other terms which come with business rules, are business rules engine, and business rules management system. Business rules engine which is a component of business rules management system is a software system that executes a set of business rules. Business rules management system monitors and maintains the variety and complexity of decision logic that is used by operational systems within an organization or enterprise. This logic is referred to as business rules. One of more widely used business rules management system is Drools, more correctly known as a production rules system. Drools use an enhanced implementation of the Rete algorithm. Drools support the JSR-94 standard for its business rules engine and enterprise framework for the construction, maintenance, and enforcement of business policies in an organization, application, or service.

This paper describes the nature of business rules, business rules engine, and business rules management system. It also prepares some information about the Drools, included several projects, which is a software system prepared by JBoss Community, and different productions made by Logitech which have used the business rules method.

Business rules

Definition – A business rule is a statement that defines or constrains some aspect of business and always resolves to true or false. Business rules declare business structure or behavior of the business. Business rules describe the operations, definitions and constraints that exist in an organization [1]. Business rules include policies, requirements, and conditional statements that are used to determine the tactical actions that take place in applications and systems [5]. Business rules are classified into two groups: Static and Dynamic. Static business rules are constraints or derivations that apply to each individual state of the business. Dynamic business rules are concerned to a request of actions in response to events [9].

While business rules show an organization the detail of operations, strategy shows the methods to focus the business at a high level to optimize results. In other words, a strategy provides high-level direction about what an organization should do; business rules translate strategy to action by defining several rules. These rules can be used to help the organization to achieve its goals, remove limitations to market development, reduce costly fees, and comply with necessary requirements [1].

For example, a business rule could show the computation of taxes for each employee’s wages.

The most important points for effective business rules are the ability to define the rules clearly and make sure that the rules do not conflict.

Business rules must be a term or fact (like structural assertion), or a constraint (like action assertion), or a derivation. They are atomic; it means that they cannot be broken further into more detailed business rules. If broken apart any further, they might be loss of important information about the business [3].

Business rules are an abstraction of the policies and habits of a business organization. We need a methodology to develop the rules which are used by business process management systems. In computer software development, this methodology is called business rules approach.

Business rules approach formalizes business rules in a language that is understandable. Business rules define comprehensible statements about business actions and using the information used to decide an action; this formal definition becomes information for processing and running rules engines [2].

Advantages

Compare to the traditional methods, business rules approach has the following major advantages [6]:

- Decrease the cost of modification of business logic
- Decrease the development time
- Make some changes more faster and easier with less risk
- Share the rules among multiple applications
- Requirements can be easily translated into rules
- Each rule describes a small portion of the business logic and is not part of a large program
- Rules are more understandable by non-technical staff, due to the use of flows, decision tables, and specific languages

Business rules add another layer to systems that automate business processes. This new added layer helps to improve the productivity in the workplace. It also enhances business agility and increases the manageability of business processes by easily accessing the rules.

In traditional systems, if we need to make some changes in business logic located inside of an automated business process, not only it often takes considerable time, but also it tends to create errors. Furthermore, since the life cycle of business models has greatly shortened, ability to adapt to changes in external environment can be worthy. These needs can be answered by business rules [2].

Moreover, in any IT application, compared to the application code, the rate of the changing of business rules is very high. Since business rules engines serve as pluggable software components which execute business rules, a business rules approach can act as an independent component which is out of application code (externalization) and has been attached to the application code. Due to this externalization of business rules, business users are able to modify the rules frequently without the need for IT intervention. Hence, the system becomes more adaptable with business rules that change dynamically [1].

Gathering Business Rules

Gathering business rules for any organizations must be done in one of the following two ways:

- Organizations can proactively describe its business practices and produce a database of rules. Although this activity may be beneficial, it may be expensive and time consuming. Using this method, organizations should hire someone, who has detailed information about the business, to collect and document various standards and methods of the business rules [1].
- Organizations can discover and document business rules informally during the first steps of a project. This business rules gathering is vulnerable to the creation of inconsistent or conflicting business rules between different organizational units, or even within an organizational unit over time. Consequently, this method may create problems that can be difficult to find and solve. If the rules are not collected correctly and if they do not cover the entire business logic, they are not valuable. On the other hand, this method is less costly and easier to perform than the first method [1].

One of the best ways to collect and document business rules is defining a methodology which is called business rules methodology. This methodology defines the process of capturing business rules in a natural language. That is verifiable and understandable way. This process can be performed in real-time. Collecting business rules is also called rules harvesting or business rule mining. Software technologies are designed to extract business rules through the analysis of legacy source code.

Categorizing of Business Rules

A statement of a business rule falls into one of four categories:

- Definitions of business terms: The language for expressing the rules is the most basic element of a business rule. The definition of a term is a business rule that shows how people describe the business. As a result, definitions of terms create a category of business rule. Generally terms have been documented in a Glossary or as entities in a conceptual model or entity-relationship model [3].
- Facts: The behavior of an organization can be described using the facts that relate terms to each other. For instance, to say that a customer can place an order is a business rule. Facts can be documented as natural language sentences, by using a graphical model, and facts can be shown relationships, attributes, and generalization structures [3].
- Constraints (as ‘action assertions’): Constraints describe conditions or limitations in behavior. For example, what data may or may not be updated, or prevent an action to taking place [3].
- Derivations: Derivation refers to how knowledge in one form may be transformed into other knowledge, probably in a different form [3].

Obstacles

Business rules are collected in the following situations:

- When dictated by law
- During business analyses
- As short-lived aid to engineers

The first obstacle of using business rules management system is the cost and effort that is necessary to maintain the set of rules. This set of rules is caused by having an inconsistent approach. If the rules have been rapidly changed, the cost of maintenance will be increased. The next common obstacle is resistance from employees who understand that their knowledge of business rules is the key to their employment [1].

Knowledge Engineering of Business Rules

Generally, due to several communication problems and misinterpretation, it is a long iterative process to model the application domain and then to develop the business rules in organization. Business rules are formal and understandable for customer who generally has limited knowledge of system development. Hence defining an integrated development environment that supports domain specific language (DSL), and visualization is vital. DSL could be helpful for declarative knowledge engineering [15].

Business rules in connection with domain specific language play a role like a bridge between the customer and the developer. Based on the declarative specification, a business analyst makes a communication between business rules and domain specific languages. In domain specific language, the developer can implement the business rules and the customer can understand this formalization which is executable. So during the development phase, the developer and the customer can discuss about DSL specification. Prolog is a language which is evaluated in bottom-up manner, and is appropriate to develop a domain specific language iteratively. Since business rules usually have to be evaluated in bottom-up, forward chaining manner, using prolog might be useful [15].

Domain Specific Language

Recently domain specific language have become popular in knowledge engineering for business rules. Unlike general-purpose programming language such as Java and C, a DSL is a specification language for a special problem domain. DSLs are remarkably used in business process modeling, and help business analyst to develop a formal specification of the business rules based on DSLs. Consequently this formal specification can be corrected and refined by the developers. It can be executable and it might be implemented later in another programming language.

One of the negative point of using DSL is that it is really difficult in practical project. It adds an additional effort at the beginning of the software project. If DSL is not developed carefully enough, the project will be failed [15].

Business Rules Engine

Definition – A Business rules engine is a software system that executes a set of business rules. The rules may come from several sources such as legal regulation (for example, a table for the calculation of taxes), company policies (for example, all employees who work more than 200 hours in month are eligible to receive a bonus). A business rules system defines these company policies and other operational decisions, then tests, executes and maintains these definitions separately from the application code.

Business rules engines typically support rules, facts, priorities, mutual exclusions, preconditions, and other functions [1].

Business rules engine software is generally provided as a component of a business rules management system which provides the ability to register, define, classify, and manage all the rules, verify consistency of rules definitions, define the relationships between different rules, and relate some of these rules to application codes that are affected by, or will enforce one or more of the rules [1]. Based on the context or behavior of the system we need to make a decision about using business rules engine or not.

- The following conditions show how defining rules engine is helpful [6].
- The logic is too complex to be dealt with using the simple condition statement in the code
- The solution might be dependent upon frequent changes
- The solution would comprise of too many nested condition statements
- The hardcode version would be unmaintainable

The following conditions tell us when using rules engines are not an appropriate solution [6].

- The logic behind the rules is simple
- Using a series of simple conditional statements inside the rule files is vital
- Regardless of the problem's complexity, if it is not under frequent changes or does not change at all
- The problem can be divided in to a small set of conditions and actions

<u>Types of Business Rules Engines</u>

Rules engines as a whole might be executed in two different methods such as:

- Forward chaining: This method typically starts with the available data and uses rules to extract more data until a goal is reached. A business rules engine, using forward chaining, searches the rules until it finds one where the antecedent rule (like “If” clause) is known to be true. When such a rule is found, the engine can conclude the consequent (like “Then” clause). Business rules engines will iterate through this process until a goal is reached [4].
- Backward chaining: This method typically starts with a list of goals or a hypothesis and works backwards from the consequent to the antecedent to search for available data that will support any of these consequents. In this case, a rules engine seeks to resolve the facts that fit a particular goal. A business rules engine using backward chaining would search the rules until it finds one which has a consequent (like “Then” clause) that matches a desired goal. It is often called goal driven because it tries to determine if something exists based on existing information [4].

Based on how rules are scheduled for execution, a number of different types of rules engines can be distinguished as follows:

- Production/Inference rules: These types of rules are used to represent behaviors of the type IF condition THEN action [1]. For example, this rule could answer the question: "Should this employee be allowed to receive the mortgage?" This rule for this question would be executed in the form of: "IF some-condition THEN allow-employee-a-mortgage".
- Reaction/Event Condition Action rules: These types of rules detect and react to incoming events and process event patterns [1]. For example, a reactive rule engine could be used

to alert a manager that an employee works in the office generally less than 8 hours almost every day.

- Deterministic rules: These types of rules do not always behave like forward chaining and backward chaining, but instead they use domain-specific language-approaches to describe policies [1]. Domain-specific language is a type of languages which defines its own representation of rules, requirement of translation to generic rules engines or its own custom engines [5]. This approach is often easier to implement and maintain, and provides better performance.

Business Rules Management System

Definition – A business rules management system (BRMS) is a software system that is used to define, deploy, execute, monitor and maintain the variety and complexity of business rules that are used by operational systems within an organization [5]. For example, Drools is a business rule management system that uses both forward chaining and backward chaining as an inference-based rules engine.

A BRMS includes, at minimum

- A repository, which is a storing decision logic to be externalized from application code
- Tools, which are using by both technical developer and business experts to define and manage business rules
- A runtime environment, which is an applications, by using business rules engines, can execute and manage business rules within the BRMS

<u>Advantages</u>

The positive points of a BRMS are as follows [5]:

- Separate business logic management teams from software development team
- Reduce dependence on IT departments for changes in live systems
- Increase control over business rules implementation
- Express business rules with increased precision, by using a business vocabulary syntax, and clarify the business policies using graphical presentation
- Improve the efficiency of processes by increasing of decision automation

<u>Disadvantages</u>

Some disadvantages of the BRMS are as follows [5]:

- Comprehensive subjective matters expertise is required for specific products. On the other hands, technical developers must know how to write rules and integrate software with existing systems
- Due to rule harvesting, integration with existing systems, security constraints, rule migration and rule edit tracking, development cycle might be long.

DROOLS

Definition - Drools is a rules engine implementation based on Charles Forgy's Rete algorithm tailored for the Java language. Rete algorithm has been adapted to an object-oriented interface and empowered to accept more natural expression of business rules with regards to business objects. Drools is written in Java, but able to run on Java and .Net [7]. Drools is designed to accept pluggable language implementations. Rules can be written in Java, and Python. Drools provides Declarative Programming and is flexible enough to match the semantics of all problem domains with Domain Specific Languages (DSL) via XML using a schema defined for the problem domain. DSLs consist of XML elements and attributes that represent the problem domain [7]. Drools introduces the Business Logic integration Platform which provides a unified and integrated platform for Rules, Workflow and Event Processing [8]. This framework provides generic method for functional and non-functional solutions. Drools consists of several projects, such as follow:

- Drools Guvnor (business rules manager)
- Drools Expert (rules engine)
- Drools Flow (process/workflow)
- Drools Fusion (event processing/temporal reasoning)
- Drools Planner(automated planning)

Drools Guvnor

Drools Guvnor is a business rules manager. By using user friendly interfaces, a business rules manager allows managing and changing rules in a multi-user environment.

Guvnor is a web and network components. The business rules manager is a combination of core drools and other tools [10]

Guvnor can be used in the following situations:

- Manage versions or deployment of rules
- Multiple users of different skill levels access and edit rules
- Lack of infrastructure to manage rules
- Exist lots of business rules

Guvnor can be used individually or by using an IDE tools (often both together). Guvnor can be "branded" and made part of the application, or it can be a central rule repository.

Guvnor cannot be used in the following situations [10]:

- Applications have the rules in a database
- Rules management system and user interface are already exist both together
- Rules are used to solve complex algorithmic problems
- Rules are essentially an integral part of the application

Guvnor Features

Include the multiple types of rules editors (GUI, text) as follows:

- Guided Rule Editor

- Rule Templates
- Decision Tables
- Store multiple rule "assets" together as a package
- Support the domain specific language
- Support the complex event processing
- Provide the version control (historical assets)
- Provide tools for testing the rules
- Make validation and verification of the rules
- Categorize the rules
- Build and deploy of its assets including:
    - Assembly of assets into a binary package
    - Assembly of a self-contained camel-server

Drools Flow

Drools Flow provides workflow for the Drools platform. A workflow or business process shows the order of execution of several steps. Describing a complex composition of different tasks is being easier by using flow chart. Moreover, processes are useful in describing state-based, long-running processes. Using these processes, Drools Flow empowers end users to specify, execute and monitor their business logic. Drools Flow is able to easily insert into any Java application or can run standalone in a server environment [13].

Drools Flow is a community project and an official workflow product at JBoss. The two traditional approaches such as process-oriented and rule-oriented make some confusion for users. It brings some ambiguity about which tool users should be using to model which bits. Drools is a move away from a rule-centric or process-centric attitude to a more behavior modeling approach with a lot more flexibility for users to model their problems how they want. Hence using Drools knowledge-oriented platform, Drools Flow provides advanced integration between processes and rules. Drools Flow is designed based on rules, independent process, and events which are integrated into the one engine as a framework with pluggable execution behavior [13].

Drools Expert

Drools Expert is a declarative, rule based, coding environment. This allows users to focus on "what it is they want to do", and not the "how to do this".

To understand the concept of rule based systems and how they work, it might be a good start from defining the Artificial Intelligence concept. Artificial Intelligence is one of a branch of computer science that develops machines and software by intelligence. Computer vision, neural networks, machine learning, knowledge representation and reasoning (KRR), and expert system are branches of AI. Knowledge representation and reasoning (KRR), and expert system have made their way into commercial systems. For example, expert systems is used in the business rules management systems (BRMS) [11].

Knowledge representation is about how we represent our knowledge in symbolic form, i.e. how we describe something. Reasoning is about how we go about the act of thinking using this knowledge.

Over the years researchers have developed approach to represent the world. Web Ontology Language is a result of these types of research. But there is always a gap between what can be theoretically represented and what can be used computationally in practically timely manner. As previous has been shown Reasoning is about how the systems go about thinking. Two types of reasoning techniques are forward chaining, which is reactive and data driven, and backward chaining, which is passive and query driven; other types of reasoning techniques are imperfect reasoning (fuzzy logic, certainty factors), defeasible logic, belief systems, temporal reasoning and correlation which Drools uses some of them. The theory driving Drools R&D comes from KRR which KRR functionalities are defined and delivered to developers by a computer program called rule engines. At a high level KRR has three components [11]:

- Ontology
- Rules
- Data

Ontology is the representation model used for describing "things". The rules perform the reasoning, i.e., they facilitate "thinking". The term "rules engine" is quite ambiguous in that it can be any system that uses rules, in any form that can be applied to data to produce outcomes. This includes simple systems like form validation and dynamic expression engines.

Drools started life as a specific type of rule engine called a Production Rule System (PRS) and it was based on the Rete algorithm. The Rete algorithm is core of a Production Rule System and is able to scale to a large number of rules and facts. A Production Rule is a two-part structure, as follows,: the engine matches facts and data against Production Rules - also called Productions or just Rules - to infer conclusions which result in actions [11].

```
When <conditions> then <actions>;
```

The process of matching the new or existing facts against Production Rules is called pattern matching, which is performed by the inference engine. Actions execute in response to changes in data, like a database trigger; this is a data driven approach to reasoning. The actions themselves can change data, which in turn could match against other rules causing them to fire; this is referred to as forward chaining [11].

Drools Fusion

Drools Fusion is the module which is responsible for enabling of an event processing capabilities. An event processing concept deals with the processing of multiple events with the goal of identifying the meaningful events among the all events. Event processing uses some techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes. It also uses the technology for building and managing information systems including [12]:

- Business activity monitoring
- Business process management
- Enterprise application integration
- Event-driven architecture
- Network and business level security
- Real time conformance to regulation and policies

Drools Fusion is a unified behavioral modeling platform which can be achieved by getting together three modeling such as Rules, or Processes, or Events modeling as their main modeling concept. In this regards a platform must understand all of these concepts as primary concepts and allow them to leverage on each other strengths. Some features of Drools Fusion, as follows, are [12]:

- Events as first class citizens
- Support asynchronous multi-thread streams
- Support for temporal reasoning
- Support events garbage collection
- Support reasoning over absence of events
- Support of sliding Windows

Drools Fusion has two goals. The first is to increase the capabilities of the Drools Expert module with features like temporal reasoning that are useful when dealing with events, and regular facts. The second allows Drools to enable modeling of event processing scenarios. Drools allows more flexibility on modeling scenario that range from batch to real time processing. Scenarios that are very frequent in business environments like [12]:

- Algorithm Trading
- Telecom Rating
- Credit Approval
- Insurance Pricing
- Risk Management

<u>Drools Planner</u>

Every organization faces several planning problems such as providing products or services with a limited set of constrained resources (employees, assets, time and money). Drools Planner or OptaPlanner is able to optimize a planning in order to do more business with less resource.

OptaPlanner is a lightweight, embeddable planning engine written in Java. It could be used to solve constraint satisfaction problems efficiently [14].

Drools Planner solves use case, such as Agenda scheduling, Educational timetabling, Job shop scheduling and so on. These use cases are probably NP-complete, this means:

- It's easy to verify a given solution to a problem in reasonable time.
- There might not find the optimal solution of a problem in reasonable time.

The suggestion of this planner is a tough task; and solving the problem is probably more difficult than anticipation of it. Advanced optimization algorithms help the planner to find a good solution for these types of problems in reasonable time by using limited recourses.

A planning problem has 2 levels of negative constraints in minimum [14]:

- A hard constraint must not be broken. For example, one teacher can not teach two different lessons at the same time.
- A soft constraint should not be broken if it possible to be avoided. For example, teacher X does not like to teach on Friday afternoon.

Some problems have positive constraints as follows:

- A soft constraint should be fulfilled if possible. For example, teacher Y likes to teach on Monday morning.

Each solution of a planning problem can be graded with a score that is result of the constraints definition. Score constraints are described by using an Object Orientated language, such as Java code or Drools rules that is easy, flexible and scalable [14].

A planning problem has a variety of solutions. The following shows several categories of solutions [14]:

- A possible solution is a solution, no matter it breaks any number of constraints or not. Planning problems could have an incredibly large number of possible solutions that numerous of them are worthless.
- A feasible solution is a solution that does not break into hard constraints. The number of feasible solutions could be relative to the number of possible solutions that some of them occasionally are no feasible solutions. Every feasible solution is a possible solution.
- An optimal solution is a solution with the highest score. Planning problems could have one or a few optimal solutions, but at least there is always one optimal solution, even in the case that there are no feasible solutions and the optimal solution isn't feasible.
- The best solution is the solution with having the highest score. This has been found by an implementation in a certain amount of time. The best solution that is likely to be feasible and, given enough time, it's an optimal solution.

Consequently, there are a huge number of possible solutions (if calculated correctly), even they have a small dataset. Drools Planner supports several optimization algorithms to efficiently go through that incredibly large number of possible solutions. Although depending on the use case, some optimization algorithms perform better than others, it's impossible to tell in advance. Changing the solver configuration in a several lines of code makes easily switch from one optimization algorithm to the other optimization algorithm in a planner.

Conclusion

Currently Business rules have been declared by Information Systems Community. Business rules are statements that describe business process. They model business structure, and they can control the behavior of process. Business rules might be appropriate to persons, processes, business behavior, and computer systems in organizations. Business rules are careful, unambiguous, and consistent approach for describing rules. Business rules engine is a software system that is responsible to execute the set of rules, and business rules management system monitors and maintains these set of rules.

Recently one of the most popular tool for business rules implementation is DOOLS that is an expert system framework; it uses rules as knowledge representation. This framework provides generic method for functional and non-functional solutions. Drools consists of several projects that they are in charge of managing the business rules, defining rules engine, providing the workflow, executing and controlling the event processing, and making automated planning.

References

[1] http://en.wikipedia.org/wiki/Business_rule

[2] http://en.wikipedia.org/wiki/Business_rules_engine
[3] David Hay, Allan Kolber, “GUIDE Business Rules Project,”, The Business Rules Group, final report, revision 1.3, July 2000
[4] http://en.wikipedia.org/wiki/Forward_chaining
[5] http://en.wikipedia.org/wiki/Business_rule_management_system
[6] Marcin Grzejszczak, Mario Fusco, “Business Rules Management Systems with Drools,”, http://www.jboss.org/drools/drools-expert.html
[7] http://legacy.drools.codehouse.org
[8] www.Jboss.org/drools
[9] Nasser Karimi, Junichi Iijima,”A Logical Approach for Implementing Dynamic Business Rules,”, Contemporary Management Research, Pages 29-52, Vol. 6, No. 1, March 2010
[10] JBoss Drools team, “Guvnor User Guide, For users and administrators of Guvnor,”, Version 5.5.0.Final, http://www.jboss.org/drools/team.html
[11] JBoss Drools team, “Drools Expert User Guide,”, Version 5.5.0.Final, http://www.jboss.org/drools/team.html
[12] JBoss Drools team, “Drools Fusion User Guide,”, Version 5.5.0.Final, http://www.jboss.org/drools/team.html
[13] www.jboss.org/drools/documentations/flow
[14] JBoss Drools team, “Drools Planner User Guide,”, Version 5.5.0.Final, http://www.jboss.org/drools/team.html
[15] Ludwig Ostermayer, Dietmar Seipel, “Knowledge Engineering for Business Rules in PROLOG,”, University of Würzburg, Department of Computer Science, Würzburg, Germany