The Two Faces of Business Rules

Tom Cushing, BA -- Joseph Gulla, Ph.D -- Ray Venoski, MS, MBA

Abstract

In this paper, the authors explain two processes used in IT regarding business rules. One process, that they call “going forward”, is about the orderly process of identifying and placing business rules in a real-time repository where they are accessed in running programs. The other process, which is the main focus of the associated presentation they call “going backward” is used to extract business rules from program logic in existing applications.

Business rule extraction exposes business rules so they can be used for application modernization or application replacement projects. While it is true that you ultimately have to look at code to identify most business rules, you can dramatically reduce the time it takes to do this if you know what code to look at and where in the code to look. This is the benefit that PathPoint’s QuickStart approach provides for business rules extraction from legacy mainframe systems. Following this approach, a desk procedure written by Tom Cushing is explained and a toolset involving CICS, TSO, batch and its PC Component, and a database for results gathering and reporting is discussed. The QuickStart approach uses PathPoint Software to gather data about applications in a business context.

Introduction

Today, two waves of activities are happening with business rules. One wave is to utilize a business-rule solution with middleware like IBM Operational Decision Manager Advanced for z/OS. Supported by this tool, the application program uses the rules that have been placed in a repository and when maintenance to the rules is needed, the application just uses the changed rules. Lets call this “going forward”. In contrast, some companies want to derive business rules from an existing application by starting with the existing program code to derive the root business rules. Let's call this “going backward”.

Part One: Going Forward--Business Rules as a System

This part of the paper celebrates an important idea in the development of business systems and the applications that support them. This idea is business rules, which are definitions resolved to be true or false that constrain or control the behavior of the business.

In the Beginning

When application programming was a new discipline, the business rules that constrained an application were likely to be embodied in the collective experience of the design team. The rules were in their heads and not necessarily written down. This isn’t to say that they were thoughtless or primitive, it’s just that these rules weren’t always organized and certainly not in a repository as that software didn’t yet exist.

How Things Have Changed
Today's businesses have the opportunity to place their business rules into a repository and use them in the execution of the operational applications that they support. The rules can be definitions and constraints and derivations related to the constraints; i.e., they contain the necessary level of detail and practicality to be helpful to operational applications.

Software Solutions
A rich example of a business-rule solution is IBM Operational Decision Manager Advanced for z/OS that's designed to be a comprehensive real-time platform used to capture, automate and manage frequently occurring, repeatable business decisions. It has support for business decisions invoked from COBOL, PL/I and Java applications executing in CICS, IMS and Batch environments.

When Does It Make Sense?
Using a business rule management system makes sense when you make frequent changes to business systems that require immediate attention and response like new regulations, procedures or policies. Or, if you are in the situation where highly variable decisions need to be automated like working with product offerings that require configuration and availability. If change is frequent and consequential than a software solution makes sense. Business-rule complexity is also an important factor in deciding to use a software solution.

Cloud Services and Business Rules
As cloud services take on more production workloads, providers have implemented the necessary services. For example, the IBM Business Rules for Bluemix service is used capture the business logic into structured business rules written in natural language instead of hardcoding this logic in the application program. Richie (2015) explained the multistep process to refine and use the rules in your cloud application under Bluemix.

What an interesting journey business rules have taken over the decades from the heads of designers and the code of programmers to a repository, rules engine and monitoring tools supported by user tools to automate and test policies. The next section of the paper explores this middleware in a deeper way.

Middleware and BRMS (Business Rules Management Systems)
Middleware is a hugely diverse and important category of software. Gartner groups application infrastructure and middleware into a market with annual revenue in excess of $24 billion (Gartner Says Worldwide Application, 2015). Within middleware, BRMS is a kind of integration software with a specific focus on business rules and the associated rules engine. According to industry estimates, BRMS is estimated to grow in size by 18 percent from 2015 to 2020 (Global Business Rules Management, 2016).

How Is it Used?
BRMS software isn't for every company but is very useful, often strategic in situations where you need to respond quickly or frequently to changes in rules and policies. If important business policy and regulation-compliance logic is imbedded in programs, then you to have to find, change and test it to make an update. With BRMS software, the changes are made to the rules in the repository not directly in the code. This change, from using a tool for business rules versus changing program code opens the door to having the rules administered by people responsible for carrying out the policies and less by programmers.

How Is it Implemented?
There are a number of components that make up a BRMS. Each BRMS has a rules repository or a place where the business rules are stored. They also have a component that is responsible for handling the execution or access to the rules at runtime. This component also included monitoring of the execution component to make sure that the rules engine is up and running. Finally, there are tools for the end user to enter business rules and to assist with testing. The web page What is Business Rules Management (n.d.) describes this in more detail and includes a useful diagram of the components.

How Does the Program Access the Rule?
There are two main constructs. Jackson (2013) explains that one technique is to imbed access to the rule in ordinary IF THEN ELSE logic. Another way is use of a decision table construct supported by the BRMS. This approach greatly simplifies program logic when processing options can be put in a table and used to look up a distinct answer or solution.

Open-Source Tools
There are many tools to survey, each with its own focus or use. Drools (Drools Overview, 2016) is a frequently mentioned open-source BRMS tool. It has an engine and a Web authoring and rules management application for end users. It also has an Eclipse integrated development environment plugin. Another tool, OpenRules (OpenRules Homepage, 2016) , created by OpenRules, Inc, makes the software available and provides technical and support services, consulting, rules compression and training on a fee basis. The software distribution is licensed in two ways: General Public License (GPL) for open-source projects and non-GPL for commercial projects.

Many Tools on and for Java
Mandarax (The Mandarax Manual, 2003) is a Java implementation of a rule engine. It supports multiple types of facts and rules based on objects like databases and Enterprise JavaBeans. It also has support of XML standards, which gives it additional implementation options. Another Java related tool is JLisa (JLisa Homepage, n.d.) that is a framework for building business rules accessible to Java. This tool is compatible with JSR 94, the Java Rule Engine API that provides a straightforward way to access and use business rules within

an application. JRuleEngine (JRuleEngine Homepage, 2016) is a Java rule engine where rules can be loaded by an XML file or stored externally in a database. The JRuleEngine distribution library can be embedded into a Java application, so it can be used in any kind of application environment.

Open Source BRMS and Cloud

The OpenStack Policy Engine, called Congress, (Congress Wiki, n.d.) enables IT services to grow their OpenStack Cloud environment with new applications while keeping the compliance and governance required by their business policies. Congress provides an open-source framework for governance and regulatory compliance across cloud services. The OpenStack Policy Engine isn't the only open-source policy engine that runs on Cloud but it has some benefits from being specifically targeted at this environment like "policy as a service" availability.

Business Rule Organizations

The Business Rules Group (Business Rules, 2016) is a non-commercial peer group of IT professionals focused on a wide range of issues related to business rules. Their website includes links to their main papers and standards and a very useful list of references.

Another group is the Business Rules (Business Rules Community Homepage, n.d.) which is another non-commercial community for business rule professionals. It provides articles, commentary, discussion areas and a variety of other hands-on resources. The Business Rules Journal appears on the site at no charge. This continues and expands the long-running publishing success of the DataToKnowledge (D2K) Newsletter which itself was formerly called the Data Base Newsletter. You can join this community at no charge.

Part Two: Going Backward--Starting with the Program Code to Derive the Root Business Rules

This part of the paper explains how you might use PathPoint's QuickStart approach - a repeatable procedure that uses PathPoint software and desk procedure to identify business rules from executing legacy mainframe applications. The underlying premise of this approach is that the business logic for the business transactions being entered is on their respective processing paths as they execute. As this approach also underscores, it is important to narrow the scope of analysis by ruling out Input Instances and programs that not likely to contain business logic of interest.

Before we explain the details, here are the guidelines to follow:

The terms

Business "logic" and business "rules" have confusion in the industry. In general, business rules are considered "a step up" (meta level) from business logic (program logic). Sometimes they are used interchangeably. Sometimes a distinction is attempted but it is not consistent.

Business rules are a narrative summary of underlying business (program) logic. See the following example:

Business rule: Give a 5% discount to any buyer aged 60 or older
Business logic in the program:

```
IF AGE-OF-BUYER > 59
   MOVE 0.05 TO PRICE-DISCOUNT
END-IF
```

The PathPoint Scenario

The Scenario is a discreet unit of business processing (a business transaction) and is comprised of one or more Input Instances that are entered in a specific order to complete the business transaction. The business logic for each Input Instance is on the activity path (program and file call statements) captured by PathPoint, and this activity path can be displayed on the Input Instance report.

Business logic/rules must be ordered in the sequence that they execute

On an Input Instance activity path, business rules are determined from meaningful groupings of business (program) logic and must remain in the sequence in which they occur, since a later business rule may depend on the execution of a prior business rule. For example, when adjudicating a claim, processing is different for a “participating“ or “non-participating” provider so a prior check is necessary, e.g., prior business rule = check for type of provider (business (program) logic: IF FLD A =’P’, GO TO PAR-PROVIDER-PROCESSING ELSE GO TO NON-PAR-PROVIDER-PROCESSING.

Use captured data where appropriate to help find/confirm desired business logic

PathPoint also captures data for COMMAREAs and files on an Input Instance path, and this data can be used to help in identifying business logic/rules on that path. Note that programs can temporarily save data to screens, COMMAREAs or temp storage files and this data can be used as input for subsequent Input Instances.

Beware of differences in design/coding techniques within Scenarios

Ideally the design/programming technique employed for a scenario (a business transaction) has an edit/validation process up front for the data that was entered, followed by an update process where the final data results (after condition checking and data manipulation) are externalized to tables/files. This would show up as file updating taking place only in the last few Input Instances. So in an ideal world, edit/validation logic is all done first, then the files are updated to complete a business transaction (a good design/coding technique).

Due to changes/enhancements that have been made over the years, however, we may see edit/validation logic for the first piece of data entered, then some file updating to save that data; then edit/validation of the next piece of data entered,

followed by some file updating, etc. This would show up as file updates in multiple Input Instances as the Scenario is being processed.

Identify useful Input Instances for business logic/rules extraction
Here are some general points to note regarding presentation, edit/validation and processing logic.

Presentation logic - In many cases, the first Input Instance simply provides an output screen for entering data that will drive the processing for the rest of the Input Instances for that Scenario (business transaction). So you may not have to look for business rules for this type of Input Instance (for this type of presentation logic).

Edit/validation logic - The next few Input Instances may be used to edit and validate the data that was entered, so the business rules in these Input Instances may be data format and data value related. Where data is being edited and validated, it may only be necessary to show the final format of that data with a business rule that describes the format and validation requirements (we may not need to show HOW editing and validation was done in the code but only state the resulting format and validation requirements as a business rule for input data). Input Instances with data errors that were corrected in a subsequent Input Instance can probably be excluded.

Processing logic - Where files are ultimately updated, you are likely to find most of the meaningful business (program) logic for this Scenario.

Desk Procedure for Deriving Business Rules from an Existing Application
One of our colleagues, Tom Cushing studied the literature on business rule extraction and developed a desk procedure that guided our analysis of the application and its business systems. Here are the main steps divided into two phases.

Phase 1: Establish a Business Overview and Starting Point for the Selected Application

Steps 1 through 6 below involve information gathering and planning to capture the data necessary to analyze where to find business logic to gather and convert into high-level business rules.

1. Overview of the application
A. What does the application do from a business perspective?
- What documentation is available?
- Who are the current SMEs, if any?
- Confirm the name of the application

2. Major Business Functions
A. What are the major business functions in this application?

B. Are they implemented in online and batch processing?
C. Which platforms? (local CICS, CTG-CICS, DBCTL, DB2, etc.)
D. Decide on the names for the business functions.

3. Prioritize the business functions for investigation by:

1. The ones best understood?
2. The ones best documented?
3. The ones with the highest client priority?
4. Can any be eliminated? e.g., we won't be providing that function in the new system, etc.

4. Prioritize the Scenarios in each Business Function for investigation by

1. Size?
   a. The smallest one first to test out and refine the business rules mining approach/technique?
      - The one with the fewest Scenarios/Input Instances?
      - Input Instances with the fewest statements?
2. Business flow?
   a. All Scenarios used to enter a certain type of order, etc. Those Scenarios can be grouped together and given a Business Function name.
3. Perceived importance?

5. Decide on Scenario and Business Function names

6. Establish the operational environment

A. Where does the application run?
   1. Which LPAR, CICS region(s), DB2 subsystems, IMSDBCTL?
   2. Is there an entry defined in the PathPoint Environment Table for this environment?
B. Are there compile listings in a PDS?
C. Are there Copy books?
   1. If so, can they be related to files or COMMAREAs?
D. If using DB2:
   a. What Plan(s) will be used?
   b. Will Explain=Yes be used?
      1. If so, is there a PLAN_TABLE for this Plan?
         - Are all the packages in this PLAN_TABLE current?
         - Use batch BINDCHEK to confirm

Steps 7 and 8 below involve gathering the application data that will be key to the analysis steps that follow. The data gathered will be analyzed through the use of the PC component.

7. Run a Collection Session(s) to capture the Scenarios for that Business Function

A. Determine who will be entering the Scenarios
B. Start a Collection session(s) selecting the COBOL Parser option and enter the Scenarios using the Scenario Naming Facility.
C. Repeat Steps 3 through 7 until all required Scenario activity for a given business function has been captured.

8. Load the resulting file into the PC component, selecting the "include source code parser results" and using the Copybook option.

Phase 2: Approach to finding business logic/rules using PathPoint

Steps 1 through 3 below explain how to use PathPoint to identify to important program logic to gather and convert to Business Rules.

1. Using the hierarchic view for each Scenario, gather PathPoint documentation:

At the Scenario level:

A. Do "**Report Scenario Objects**" to:

1. Identify all programs used by the Scenario.

a. This will determine all the compiled listings you might need for this Scenario. Take note of programs that are not likely to have business rules exercised such as error handlers, I/O modules, event logger and program router. Remove the compile listings for those programs from the PDS.

b. Let you assess how big the effort is.

2. Identify how many files are involved and which files were updated.

B. Do **Search** by "**Type of calls**" (**click on Creates, Updates, Deletes**) for "**Table/File I/O by Table/File**"

The display will show you the Input Instances that have updates and which programs update which files (print out this display). The Search will also place checks in the Input Instance boxes where updates occur. When you subsequently do "Show Statements" for an Input Instance with a checked box, the create/update/delete statements will be highlighted.

C. Do "**Report Scenario Overview**" using the "**include source code references**" option to display the activity path for all Input Instances for this Scenario. This activity path contains navigation logic with compile listing line number references that point to the programs and locations where business logic can be found.

2. Business logic/rule investigation using PathPoint: First pass – Input Instance/program assessment

First, determine meaningful Input Instances and programs, eliminating others from further investigation, where appropriate:

1. Using the hierarchic view, go to the desired Scenario and (using the drop down plus sign to the left of the Scenario box) display the Input Instances for that Scenario.
   - At this point, you know which Input Instances update files (they have checks in their adjacent boxes from the prior Search in 1.B)
2. Try to determine a "business picture" for this Scenario by using the "**Show User Screens**" display. Try to assess the business purpose for each Input Instance.
   - Look at each input and output screen (or COMMAREA) for each Input Instance and use the description field at the bottom to record any observations you make. To assist in this effort, do "**Show Statements**" as

required to see the type of processing that was done and use the file call statements to see what data was required.

3. When done with all Input Instances, mentally step back and assess where most meaningful processing is taking place for this Scenario and eliminate any Input Instances that are not useful for meaningful business rule investigation.

3. Business logic/rule investigation using PathPoint: Second pass – Detail investigation

a. Using the Input Instance Report for the first meaningful Input Instance, start at the beginning of the path (statement sequence number 1)

b. Locate meaningful groups of business logic

Follow each program/file call statement using the navigation logic between statements, together with appropriate compile listings, to locate/identify the first meaningful group of program (business) logic that contains the first business rule. Use data from the COMMAREA or files, where available and appropriate.

NOTE: For CICS-DB2 listings, do a Find on "enterprise" to get you to the COBOL compile listing (past the SQL precompile and Command Language Translator listings) and place your cursor at that point.

1. Using values for "Stmt #" from the Input Instance report, do a **Find** on the statement number in the compile listing to get you to that location in the compile listing.

   NOTE: The current Input instance report only shows the FIRST potential path to the following statement. To see additional paths, do "Show Statements" to get the statement list and use the "+" sign at the left of each statement to follow the path(s) to the next statement. An alternate view of the statement list is shown by doing "ALT-A". This view may be easier to use for analysis and also provides a Comment field to make notes on any statement.

2. Encapsulate the business logic and develop its business rule:
   a. Decide on what code is relevant, copy that code to a Business Rules database (BRDB). The BRDB should include Business Function, Input Instance number, program name, compiled statements containing the business logic, related program comments, , working storage fields referenced in the business logic and their definitions, and meaningful meta-data and comment.
   b. Examine this group of business logic and describe the business rule that expresses it.
   c. Write the business rule for this business logic in the Word document.
3. Perform steps 1 and 2 for each meaningful group of business logic in this Input Instance and subsequent Input Instances in this Scenario.
4. Consolidate business rules.

   In many cases, multiple rules can be consolidated into a more generic business rule by combining business rules having similar meta-data and descriptions or are simply duplicates. e.g., the same or similar business rule may be repeated in different programs and Input Instances or

appear in batch processing. This effort often requires multiple passes of the BRDB by sorting the Business Rules by Business Funciton, meta-data, comment and the rule itself.

5. Print out or export the business rules to a BRMS.

Summary

The entire concept of business rules processing is fascinating regardless if you are extracting or building an application that uses rules in a repository. One interesting recent development is software that extracts rules from existing applications and then uses the extracted rules to build a repository for the future system.

References

Business Rues Community Homepage. (n.d.). Accessed in May 2016 from www.BRCommunity.com

Business Rules Group Homepage. (2016). Accessed in May 2016 from http://www.businessrulesgroup.org/home-brg.shtml

Congress Wiki. (n.d.). Accessed in May 2016 from https://wiki.openstack.org/wiki/Congress

Drools Overview. (2016). Accessed in May 2016 from http://www.drools.org

Gartner Says Worldwide Application Infrastructure and Middleware Market Revenue Grew 8.8 Percent in 2014. (2015). Accessed in May 2016 from http://www.gartner.com/newsroom/id/3034519

Global Business Rules Management System (BRMS) Market 2015-2020. (2016). Accessed in May 2016 from http://www.marketwired.com/press-release/global-business-rules-management-system-brms-market-2015-2020-key-players-include-bosch-2098103.htm

Jackson, C. (2013). Create and execute business rules in IBM Integration Bus. Accessed in May 2016 from http://www.ibm.com/developerworks/bpm/bpmjournal/1308_jackson/1308_jackson.html

JLisa Homepage. (n.d.). Accessed in May 2016 from http://jlisa.sourceforge.net

JRuleEngine Homepage. (2016). Accessed in May 2016 from http://jruleengine.sourceforge.net

The Mandarax Manual. (2003). Accessed in May 2016 from http://mandarax.sourceforge.net/docs/mandarax.pdf

Openrules Homepage. (2016). Accessed in May 2016 from http://openrules.com/index.htm

Richie, A. (2015). Using Business Rules Service on IBM Bluemix. Accessed May 2016 from https://developer.ibm.com/odm/2015/03/09/using-business-rules-service-ibm-bluemix/

What is Business Rules Management? (n.d.). Accessed in May 2016 from https://www-01.ibm.com/software/websphere/products/business-rule-management/whatis/