Chapter 4
Rule Harvesting

Target audience

- *(Must) business analyst; (optional) project manager, application architect, rule author*

In this chapter you will learn

- *What are the different types of rules, and why it is important to understand them*
- *How to set in place the rule harvesting process according to the source of rules and the team structure*
- *How to extract a data model for the rules from the rule description*
- *How to prepare the rules for implementation*
- *How to put into practice these techniques with a claim processing application*

Key points

- *Start by a decision point that is simple but still brings business value to the stakeholders.*
- *Describe rules using the business domain vocabulary, and future map it to a logical data model.*

4.1 Introduction

Rule harvesting includes the two main activities of rule discovery and analysis, with the goal to understand the business entities (conceptual data model [CDM]) within the scope of the application and to identify and extract the rules. A key activity in the rule harvesting phase is to formalize the decisions made during the execution of the business process by defining the different decision point candidates for business rule implementation.

Agile Business Rule Development (ABRD) puts the emphasis on developing the system through short iterations. Each iteration produces a working set of rules.

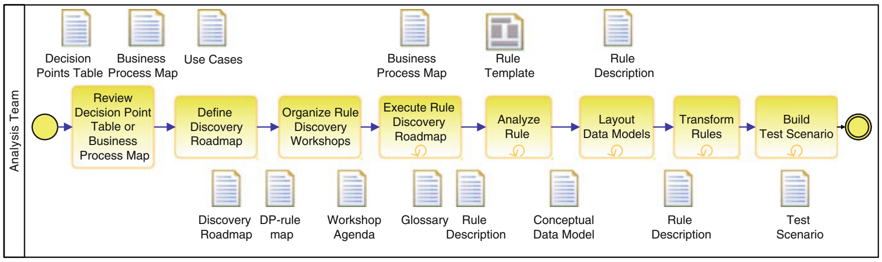

Fig. 4.1 Rule harvesting activities

Feedback from the harvesting and prototyping phases forces the subject matter experts (SMEs) to better understand their own business processes and help them to adapt those processes for more efficiency. Rule harvesting is a short project activity executed for each decision point in scope for implementation. The process flow may look like in Fig. 4.1.

Section 4.2 will discuss rule discovery. First, we start by identifying the different kinds of rules, then we describe the discovery activities. There are different ways of conducting rule discovery, depending mostly on rule sources, i.e., where we are going to discover the rules from, and the organization's modeling and requirements tradition (e.g., using use case, or process maps, or business event analysis, etc.). Thus, as Barbara von Halle suggests, part of rule discovery activities is . . . figuring out how to discover rules, i.e., *defining the rule discovery roadmap*. Section 4.2 will discuss common activities (e.g., review decision point table, define discovery roadmap, gather documents, document rules), as well as roadmap specific activities (e.g., discover rules from SMEs, discover rules from documents, discover rules from code). Section 4.3 shows rule discovery for our case study.

Rule analysis is discussed in Sect. 4.4; many of the techniques presented are based on vonHalle's STEP methodology. Analysis activities include (1) reviewing rule descriptions and fact models (Sect. 4.1), (2) transforming rules to obtain unambiguous, atomic, nonredundant, and consistent rules (Sect. 4.2), (3) building test scenarios (Sect. 4.3), and (4) verifying the rules against the data model (Sect. 4.4). Section 4.5 shows rule analysis for our case study. We conclude in Sect. 4.6.

4.2 Rule Discovery

Rule discovery, also called Business Rules Modeling in the industry, aims to develop simple modeling artifacts like rule descriptions, business entity diagrams, and business process maps. As described in Chap. 3, the development team executes this activity on a regular basis during the development of the business application. Rule discovery is an iterative process that will identify a subset of rules and document them as opposed to spending months figuring out all the rules up front and producing a huge document.

Business rule discovery techniques are similar to those used for traditional requirements elicitation, with one main difference: focus on those special needs that support decisions on how the business is executed in the company. From the inception phase the project team gets a set of work products that are used during rule discovery. These artifacts include:

1. A high-level description of the business process
2. A high-level description of the current and future architectures
3. A list of data sources and data models
4. The decision points table

The decision points table, in particular, helps define where to find the rules (rule sources) and which method to use for rule harvesting. The rule discovery process changes according to the sources used. For example, working from a legal document implies a different discovery process from the discovery based on interviewing subject matter experts.

4.2.1 Classification of Business Rules

Before deciding how to write rules and where to implement them, you first need to understand which types of rules your team will be harvesting. In early 2008, the Object Management Group (OMG) finalized a specification for documenting the semantics of business vocabularies and business rules, entitled *Semantics of Business Vocabulary and Business Rules* (*SBVR*).

The specification describes SBVR as part of the OMG's Model Driven Architecture (MDA). Its purpose is to capture specifications in natural language and to represent them formally to facilitate automation. SBVR includes two specialized vocabularies:

- One to define business terms and meanings from the perspective of the business teams. It is named in the SBVR specification as *Business Vocabulary*.
- One used to describe business rules in an unambiguous way leveraging the business vocabulary.

The *meaning* is what someone understands or intends to express. The meanings are derived into concepts, questions and propositions. A phrase such as "We deny the invoice if the medical treatment was done after one year of the accident" has a clear meaning for a claim processor (CP) within a car insurance company. Analysts need to logically transform this meaning into concepts that have a unique interpretation so that we can represent the business knowledge within a comprehensive vocabulary. Concepts include a unique combination of characteristics or properties.

Within the Business Motivation Model (BMM),[1] the OMG has also defined the relation between business policies, directives, business processes, and business rules. This work is very important to clearly classify each of those concepts. The OMG definition of business policy is: "A non-actionable directive whose purpose is to govern or guide the enterprise. Business policies govern business processes." A Business rule is – "A directive, intended to govern, guide, or influence business behavior, in support of business policy that has been formulated in response to an opportunity, threat, strength, or weakness. It is a single directive that does not require additional interpretation to undertake strategies or tactics. Often, a business rule is derived from business policy. Business rules guide a business process." For the purpose of rule harvesting, keep in mind that business rules are actionable, unambiguous, and derived from the business policy. Considering rules as semantically meaningful, rather than business policies, is key to making them executable.

The OMG BMM reuses some classifications from the SBVR: business rules are separated into two possible classes:

- Structural (definitional) business rules which are describing the structure of the business entities used by the line of business organization. Such rules describe constraints on the model, like possible value, or mandatory inclusion or associations.
- Operational (behavioral) business rules are developed to enforce business policies, seen as obligations to execute efficiently the business. When considering operational business rules it is important to look at the level of enforcement and where the rule enforcement occurs.

In SBVR, rules are always constructed by documenting conditions to business entities defined in the *business vocabulary*. A fact is a relationship between two or more concepts.

Another approach to define facts is to use the Ontology Web Language (OWL) and Resource Description Framework (RDF). Developed to specify semantic web,[2] OWL and RDF can be used to model the enterprise ontology. The ontology is the source for data models used by the rules as an alternate to traditional Object-Oriented Analysis (OOA) and SBVR. OWL and RDF implement an object-relational model allowing creation of a directed graph, a network of objects and relationships describing data.

Using a mix of the SBVR classification for business rules, OWL–RDF to describe the domain and an older rule classification model which we have used for years in consulting engagements, the different types of business rules can be presented as shown in Fig. 4.2.

[1] *Business Motivation Model V 1.1,* Object Management Group at http://www.omg.org/spec/BMM/1.1/.

[2] From Wikipedia, semantic web is defined as an extension to the WWW in which the meaning of information and services on the web is defined, making it possible for the web to "understand" and satisfy the requests of people and machines to use the web content.

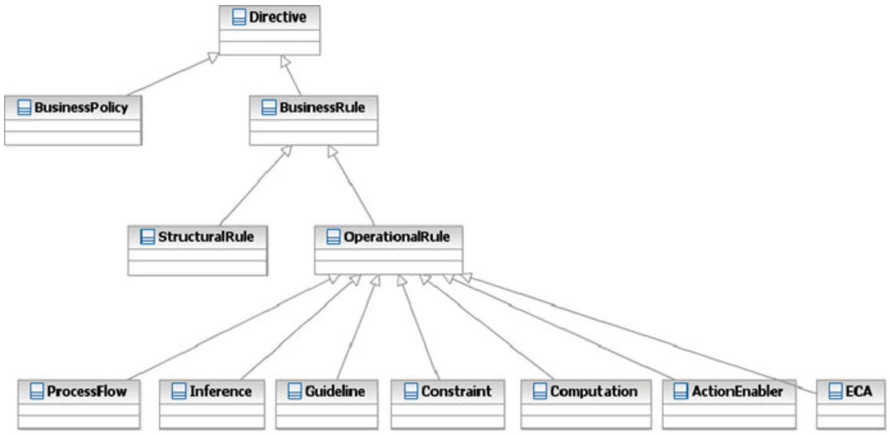

Fig. 4.2 Business rule schema

This schema represents the different types of rules that are relevant to the business, including structural and operational rules. Structural rules define the terms used by the business in expressing their business rules and the relationships (facts) among those terms. These include the vocabulary used in rule authoring. As an example a statement like: *An insurance policy includes a set of coverage. The policy is effective at a given date and needs to be renewed every six months.* Transforming this statement implies defining a structure in the insurance domain, where an insurance policy entity has an effective date, expiration date, and a list of coverages.

Operational rules are the rules that implement business decision logic. When a business decision is made (e.g., whether to sell a given insurance policy, whether to accept or reject a claim), the business rules are the individual statements of business logic that are evaluated by the rule engine to determine the decision result.

The following table is adapted from the work by Barbara Von Halle and is a simplified view of the business rules group classification. It details those categories:

Rule classification	Explanation
Mandatory constraints	Rules that reject the attempted business transaction. Grammar to use during rule documentation not implementation. <term> MUST HAVE <at least, at most, exactly n of> <term>; <term> MUST BE IN LIST <a,b,c>; SBVR expression: **it is [not] necessary that** <**fact**>
Guidelines	Rules that does not reject the transaction; they merely warn about an undesirable circumstance. Usually translates to warning messages. <term> SHOULD HAVE <at least, at most, exactly n of> <term>; <term> SHOULD BE IN LIST <a,b,c> SBVR expression: **it is [not] possible that** < **fact** > **It is possible but not necessary** < **fact** >

(*continued*)

Rule classification	Explanation
Action-enablers	Rules that tests conditions and upon finding them true, initiate another business event, message, business process, or other activity IF <condition> THEN action
Computations	Rules that create new information from existing information based on mathematical computation <term> IS COMPUTED AS <formula>
Inferences	Rules that create new information from existing information. The result is a piece of knowledge used as a new fact for the rule engine to consider IF <term> <operator> <term> THEN <term> <operator> <term>
Event Condition Action (ECA)	Rules where the condition is evaluated once the occurrence of an event is found. Most ECA rules use temporal operators to search events related to their timestamp of creation or occurrence On <event> when <condition> then <action>

To implement guidelines and constraints, you need to consider what happens when they are violated. Most of the time, the action raises an exception or a high priority issue to be managed later in the business process, which may reject the business event. The rule needs to report clearly on the selected decision so that a human can understand and act on the business transaction.

A guideline written as: *The date of loss should be before the expiration date of the policy* may translate to the following rule: *if the date of loss is after the expiration date of the policy, then create a warning ticket for the claim processor to handle*. This implementation allows the insurer to make allowances for an insured person who has a history of regularly renewing the policy but for some reason forgot to renew on time.

A constraint written as: *The borrower must put a minimum cash down of 5%* translates to this rule: *if the minimum cash is below 5% then reject the loan application with the reason "The borrower must put minimum cash down of 5%."*

Action enabler rules modify, create, or delete terms or association between terms, or execute methods, which can be web service. For example, a rule like: *if a driver has made one or more at-fault claims in the current year, decrease the discount rate by 3%* changes an attribute ("discount rate") of an object.

Computation rules implement mathematical equations and assign values to variables according to a set of given criteria. For example, a risk factor variable can be computed according to the age of the driver. It is important to note that management of computation rules may require managing the entire ruleset together if there are rules that are required to be managed prior to those calculations and at the terms of the calculation.

Process flow routing rules direct the movement through a process flow or workflow. Process flow rules are distinct from business logic rules. It may be helpful to distinguish process flow rules from the business logic rules that determine the values of the parameters on which the process flow is directed as such rules are more often complex and numerous than routing rules. Routing rules may be written as: *if there is at least one exception in previous activity of the process goes to this task if not continue on the main path*. The business logic to define if there is an exception is made within a rule engine with a lot of rules to evaluate and execute.

Inference rules use syntax similar to action enabler rules, but they create new objects or facts which may bring the engine to re-evaluate some other rule's eligibility. During discovery, it is important to understand the execution context as seen by the business user and be able to answer questions like: "If executing this rule modifies the state of the claim, will the eligibility rules that have already executed need to be reevaluated?" For example, an insurance policy underwriting rule that says *if the age of the driving license is below 3, add a risk factor of 50 and reevaluate the total risk score* modifies the risk scoring variables, which requires that other rules be reevaluated.

It is possible to continue the decomposition of those rules. For example, transformation rules in ETL (Extract Transform Load) are often considered separate from other business rules; although in pattern, they are essentially inference rules and computation rules. Data transformation rules, while important to the business, are a side effect of system implementation rather than a reflection of core business logic. For implementation, the decision to use a rule engine for data transformation rules depends on whether the rules are static, dynamic, or business driven. Some implementations use a rule engine to easily implement transformation rules between two data models instead of using a complex scripting language when the transformations have to be maintained by business users.

Recently, a new subcategory of ECA rule appeared within the IT horizon: the Complex Event Processing (CEP) statements (or rules) which support a category of business rules related to real-time event filtering, event aggregation and correlation, and applying pattern matching conditions on attributes of events. CEP applies business rules to a stream of data. A business rule to detect fraud on banking cards may be written as: *Raise a warning alarm if more than one transaction for an amount over $100 is received from the same merchant on the same card number within the last 15-minutes*. According to the preceding rule classifications, this rule would be considered a mix of ECA and inference rules. However, one important dimension of this type of rule is the time window constraint on which the rules apply and the type of language used to write the rule. Today most of those languages are based on SQL and include operators to look at time window. The action part of the rule creates new events (called the complex events), which are managed by a downstream application. The convergence of a CEP engine with a BRMS platform starts to happen, as rule engines excel in pattern matching, and can apply more complex decisions on already aggregated and filtered events. In fact, alarm filtering and correlation applications in telecommunications network management are examples of complex event processing implemented with a BRMS (see Sect. 2.2.1).

In addition to industry standards, here are other rule patterns commonly found in business applications:

Rule classification	Type of application
Compliance rules	Rules that reject the attempted business transaction Yes/no result but completed with reason code and explanation Underwriting Fraud detection Data and form validation Example: *Whoever receives a commission paid by an insurance company for the sale of an insurance policy needs an insurance license*
Rating	Strongly interrelated rules that compute metrics for a complex object model Scoring and rating Contracts and allocation Pure calculations on an object providing a final value (or rating) Example: *if the driver is between 16 and 22 years old the risk is set to 400* *If the driver is between 23 and 27 the risk is set to 300 . . ..*
Correlation	Strongly interrelated rules that correlate information from a set of objects to Compute some complex metrics Billing and cost estimation Complement by inserting information Example: *if the medical bill references a patient, and the patient is not declared in the related claim then there is an issue on the claim description or the invoice is not related to a patient covered*
Stateful	Strongly interrelated rules that correlate events in a stateful way. Stateful in this context means the internal states of the engine are maintained between rule execution invocations Alarm filtering and correlation Web page navigation GUI customization Example: *if there is an alarm L1 on network element A and an alarm L2 on network element B and a direct link exists between A and B then the alarm L1 is linked to L2, remove alarm L2*

Classifying rules facilitates the design cycle, which focuses on deciding the best implementation for a given rule. Inference and action enabler rules are good candidates for a rule engine. Pure computation will most likely be implemented in code unless computation rules are subject to frequent changes in the criteria of applicability or are linked to others business rules. The classification also helps to evaluate the complexity of the rules and the workload to implement it.

4.2.2 Discovery Activities

The discovery phase contains some preparation tasks, such as reviewing the decision point table, the use case model or the business process, defining the discovery roadmap, and organizing the elicitation workshops. Some activities are recurring such as executing the discovery itself. These are basic steps, which you can extend to your own project needs (see Fig. 4.1). The remaining subsections detail some of those activities. The work products listed in this process are part of ABRD EPF plug-in.

The discovery activities are conducted during the elaboration phase of the project, but the same process is conducted even after the system has gone into production when there is a new business event, or when there is a need to modify some decision or some business policy. Companies have been operating with business rules for many years, but the form of these rules is not externalized and managed as standalone artefact. Capturing business rules relies on a combination of eliciting business requirements, reverse engineering existing applications, and expert's knowledge. Business rules are not just requirements: they specify how a business process is executed, how decisions are made, and how domain knowledge is structured. When using a business rule approach for business requirements elicitation, we are working at the business process and policies and procedures level to understand the constraints and the behaviours of the process.

The most unique aspect of the rules discovery phase is the perception of a business event as a set of decision-rich activities. We unfold the processing of a business event as a set of decisions and policies. We then dissect the decisions and policies into executable and precise business rules that guide the business event in a predictable and desirable manner. We will detail these concepts in Sect. 4.4.

There are two dimensions to consider when preparing the rule discovery activities or roadmap.[3]

- The source of rule, which can be:
  - The *documentation* which includes all sorts of documents (business plans, deliverables of earlier projects, legislation, regulations, standards, and business requirements for the current project)
  - The tacit *know-how*: the unwritten "way we do things", embodied as a collective way of doing things (organizational intelligence), or as individual expertise
  - The *legacy system*, which includes an operational system that implements some of the business logic (code mining)
  - The *Business records* as the way particular customer requirements have been satisfied, pricing formulas used in previous contracts
- The type of analysis techniques used by the project team:
  - Business event analysis
  - Use case elaboration
  - Business process modeling
  - Business mission and strategy analysis
  - Data analysis

Obviously the *know-how* discovery is the most difficult elicitation task to execute and the one that usually takes the longest time. We will provide details later in this section on how to conduct such an elicitation workshop.

[3]The term "rule discovery" roadmap is also used in the industry to present the journey the analysts go through.

The following table is giving the different possible starting points for the discovery activities based on the analysis method used:

Starting point	Analysis description
Business events	Start with the business events listed in the inception artifacts. Some example can be: a claim or invoice is received, the loan application is submitted, the call data record is posted ... Each business event is processed by a set of activities that can be described in document or business process. This approach is rarely used
Use case	Analyze use case description to find decision points and then rules. Preferred approach, for teams familiar with use cases, or user stories
Business process – workflow	Evaluate individual process steps and tasks to define the decision behind activity and then the rules. Used when the organization uses process decomposition for the requirements gathering and analysis phase. We group workflow in this category
Data analysis	Used in case of data change-oriented rules project. The project team looks at the life cycle of the major business objects and extracts the processes and decisions used to change the state of the data. It can start with the logical data model and how the business entities are created, updated, deleted and what their states are. One example of such an approach is to look at the states of an insurance policy, and how, who, when changes are made
Business mission and strategy	Based on a top-down rules and business policies approach. Rules sources are high-level manager, decision makers, legal documentation ...

It is important to set the expectation among the stakeholders that not all the rules will be discovered during this phase. The goal is to complete the rule discovery up to 40–60% so we can have some tangible decisions on standard business events to process. The rule writers and the development team will increase the amount of rules in scope during future iterations of the implementation.

4.2.2.1 Review Decision Point Table or Business Process Map

When the business modeling activity of the Inception phase is completed, the project team should have a decision point table document as a source for the rule discovery phase. If not, it is still possible to build it from the description of the business process. There are different ways to extract the decision points table.

Use Case Approach

If the project team uses use cases approach to document requirements, the rule discovery team can study the use case descriptions to identify those tasks or activities where the system makes a nontrivial decision. In her book "Business Rule Applied", Barbara Von Halle suggests looking for verbal cues in task/activity descriptions that might suggest a nontrivial decision making. For example, verbs such as check, qualify, compute, calculate, estimate, evaluate, determine, assess, compare, verify, validate, confirm, decide, diagnose, or process may hint at some "intelligent" processing. Behind these verbs lurk lots of business knowledge and business rules.

Here is an example of a use case description for a basic loan underwriting process:

Use case name	Check mortgage eligibility	**Version**	0.9
Problem domain	Handling mortgage applications	**Author**	
Purpose	A **Borrower** has submitted a mortgage application, along with supporting documentation. A **loan officer/clerk** has verified the supporting documentation, and input the application into the system. The data of the application has been validated. We check three sets of criteria, in this order: (mortgage) loan eligibility, borrower eligibility, and property eligibility. When one of them fails, we exit the use case.		
Actors	**Mortgage Loan Officer** (or clerk), on behalf of **Borrower**		
Trigger events	The data of a mortgage application has passed validation, and is now submitted through eligibility		

System response	**Decision**
Start the eligibility verification	
Verify the eligibility of the loan	• Check the type of loan
	• Check the transaction type
	• Check the loan amount
	• Check the down payment
	• Check the term of the loan
	• Check the loan to value ratio
The loan is eligible.	• Check the age of the borrower
Verify the eligibility of the borrower	• Check the immigration/citizenship status of the borrower
	• Check the financial situation of the borrower
	• Check the number of mortgages

This use case template includes a decision column used to drive the discussion during the rule discovery.

Business Process Modeling Approach

Business process modeling involves the same approach and should include at least the following activities:

- Define the actors of the process – by roles. Clearly list the different human actors of the process and classify them by role.
- Design the as-is process with tasks and dependencies. Do not attempt to analyze the full process in one shot, instead, use an incremental approach. Use BPMS editor to design the process and simulate it. BPMS includes some simulation tool that helps to verify the process being analyzed.
- Identify branch points in the process which lead to different subpaths from the current point. The decisions to route to one of the sub branches can be considered as business rules: in Fig. 4.2, those routing rules are simply expressed in the process model as a set of conditions leading to different branches. The simplest response will be providing a binary response, but it is possible to define responses

as a set of predefined values of an enumeration such as {good, average, bad}, {gold, silver, and platinum} …. From our experience we do not recommend having a lot of branches coming out of a conditional node in a process map. A typical process will have from 2 to 6–8 branches.

With a business process modeling approach, the analysis team looks at task descriptions to search for mental thinking verbs, the same. Then the analyst works with the subject matter experts to understand how the decisions on those activities are made. If there are decisions based on business practices and decisions, we need to log in a table format the task reference, what are the sources for rule, who is the owner, etc. Each row of the table forms a decision point. Once decision points are identified, a review of each decision point is needed. This review should take less than half a day to conduct. This session allows stakeholders to review the decision points and to set the priority for rule harvesting at each decision point level. To complete this task, you may need to get answers to at least the following questions:

- What is the current process to define, document, implement, test, and update the business rules?
- Who owns the rules and the business policy definitions within the business organization?
- Are there any classifications such as country/geography or product category with some specific rules we need to take care of? Is the same team defining them? We were working on projects where at the beginning of the project we were dealing with the core business team, but a lot of rules were overridden by the branch offices in the different countries, so the elicitation process has to be adapted to get such information.
- What is the number of rules for each decision point?
- What are examples of actual rules?
- Is there a rule sharing policy?

A good practice is to start with a simple, well-understood decision point, to help train the team on the elicitation practices, but keep in mind that the management will want to see the business value of what the team is working on. So a decision point which brings a lot of business value should be at the top of the list. With the iterative approach of ABRD, we can develop an executable ruleset in one time boxed iteration of 20–25 days. This is important to show the value of the approach with tangible results.

The purpose of this activity is to preset the roadmap definition phase and to verify that we have the important information on the business process and the related decision points.

4.2.2.2 Define Discovery Roadmap

The definition of the discovery roadmap is an important step to understand how the analyst team will extract the rules from the different kind of sources. The selection of the type of roadmap is linked to the rule source. Tony Morgan in his book

"Business Rules and Information Systems: Aligning IT with Business Goals" proposes the following discovery processes:

- The *static analysis* process uses reading and highlighting the rules within documentation, which can be legal, internal policies, procedure. The team has to gather all the related documents with the reference on version, date of creation, and validity. The elicitation is based on reading sessions completed with Question/Answer workshop sessions.
- *Interactive* involves working sessions with subject matter experts who have the knowledge of the business process and the decisions within a process task. Also a person doing the day-to-day activity is a very good source to understand how decisions are made and how exceptions to the main business process are handled. The process to elicit rules from people will be accomplished by using elicitation workshop.
- *Automated* involve using a computer and special applications to search for rule statement within procedure code, SQL procedures, code listing, and so forth. When using rule mining technology, we have to be careful to not lose the context of execution in which the if-then-else statement was implemented. Therefore, code review should always be complemented by workshop sessions for Q&A.

Code mining is one activity our customers or prospects request quite often, but which ends up being less efficient than expected. Care needs to be taken on that matter as responding and addressing the following items can be time consuming:

- Who has the knowledge of the current code? Is this person still in the company?
- Should the current business context use the same business rules as 15 or 20 years ago? If those rules are still valuable and valid they should be well known by the company and no code mining is required.
- Not all "If-then-else" statements in legacy code represent business rules, sometimes procedures, functions, and algorithms may be an implementation of business rules. The context of execution is a very important dimension to understand before reusing a coded (business) rule as-is.
- Variable names were often limited to eight characters in a flat data model. There is no need to keep it that way. You may want to think about designing an efficient object-oriented model.
- Most of the time automatic translation of badly coded business rules will generate bad business rules in the new environment.
- Business rules implemented for a business rule engine have a different structure than procedural code. They should be more atomic and isolated[4] (see also Concept: Atomic Rule in a later section), and the rule writer may leverage the inference capacity of the engine. Therefore, the automatic translation will produce poor results.

[4]A business rule is said to be "atomic" in that it cannot be broken down or decomposed further into more detailed business rules. If reduced any further, there would be loss of important information about the business (Source: http://www. businessrulesgroup .org/first_paper/br01c3.htm).

The following table summarizes the different techniques classified per type of source, based on Morgan (2002):

Source	Static analysis	Interactive	Automated
Documentation	Very good fit	As a complement of static analysis	Not yet possible
Know-how	Not applicable	Unique solution	Not applicable
Code	Efficient	As a complement of the other processes	Gives good result
Business record	Depends on the source	Moderate or a complement	Depends on the source (may be impossible)

When the source of the business rules is people, individual interviews are required to get the core of the knowledge and then followed up with workshops to resolve outstanding issues and process exception paths with the team.

Once we understand the type of elicitation roadmap, we can move to the preparation and execution of the rule discovery activities.

4.2.2.3 Gather the Related Documents

For rule discovery based on documentation or code, the project team must gather all the applicable documents and add (and version) them in a central document repository for traceability purpose. The more information the team can gather at the beginning of the discovery the easier the elicitation job is. There is a common pattern of human thinking that the system is working a certain way, but no document or even code can prove it really works as expected.

4.2.2.4 Studying Decision Point

It is a good practice to automate, with rule processing, the decision points of the business process, leaving the exceptions to humans. Over time some exception handling can be added to the rules. A typical case can be seen in loan underwriting rules: An expert may quickly extract the main rules to support the loan application (the loan to value should be under 85%), but over time, market conditions, regulations, new legislations, and competition may enforce the line of business to define exceptions to the core rules. Those exceptions are added to the ruleset as new rules.

From the decision point table extracted in the previous activity, it is important to complete its description by specifying the list of decisions required at this point of the process. This table may be completed by logging the outcomes of conversations with the different experts or by reading legal documents.

4.2.2.5 Organize Rule Discovery

To make a better use of the development and business teams' time, it is important to plan in advance the workshop sessions and to clearly state what is in the agenda. We recommend organizing the day in two parts:

- Use the morning for discovery workshops using elicitation techniques with the project stakeholders and subject matter experts. During the rule harvesting cycle of ABRD, the analyst team may want to use the rule template document to enter the rule description and use some simple diagramming techniques to define the business entities as conceptual data model[5] (A good tool to use is a UML class diagram editor, by adding entities as class and attributes and omitting the details of the methods and the associations). Ensure the tool, notation used are clearly understood by team members.
- Use the second part of the day to perform the analysis activities.

As explained in the previous chapter, the discovery workshops are executed using different frequency of occurrence. In ABRD harvesting and prototyping cycles, the workshops can be set every morning, but when starting with the implementation cycle, they could occur only every 2 days or more, but never more than a week apart to keep the team focused and enforce feedbacks.

The team should have access to a dedicated meeting room with white boards, pencils, paper, post it notes, and potentially a UML tool to quickly develop diagrams. To organize the sessions, the project team may need to name a moderator responsible for managing the meetings and keeping the team on track. The moderator role is to:

- Establish a professional and objective tone to the meetings
- Start and end the meetings on time
- Establish and enforce the "rules of conduct" of the meetings
- Introduce the goals and agenda for the meetings
- Facilitate a process of decision and consensus making, but avoid participating in the content
- Make certain that all stakeholders participate and have their input heard
- Control disruptive or unproductive behavior
- Gather "Open Points" and follow up actions between sessions (use a simple Excel sheet for instance or "Meeting Minutes" template document)

To organize the workshop, the project manager has to set a strict agenda inviting all the domain experts who will help to formalize the rules. Gather the required documents and explain how the meetings will be managed. The agenda may have at least the following information:

[5]A conceptual data model defines the meaning of the things in an organization and includes business entities and their associations.

- Which decision point is being discussed in this meeting
- Which documents to use
  - Rule template
  - Glossary of terms document
  - Business process map or use case documents
  - Conceptual data model
  - Any additional helpful documents/resources
- The meeting room and the schedule
- The name of the moderator
- The high-level rules to follow during the meeting like:
  - Be on time: you will have one "joker" for one time late. A fee of $5 will be taken after that towards a conclusion party
  - In each session all the members should participate
  - We will use brainstorming techniques
  - The moderator controls the time
  - Everyone can have their opinion
  - No criticism

The session should not last more than 2 h, typically from 9 to 11. This can be scheduled for 2 or more consecutive days.

4.2.2.6 Execute Rule Discovery Roadmap

This activity supports the three types of rule discovery: business users and experts workshop session, document study, and legacy code mining. Even if the main sources of rules are documents or code, it is still important to come back to an SME to get feedback on what the team discovered. Note that access to SMEs is quite often challenging because they are typically engaged in other production projects. To reduce this impact to a minimum, it is very important to do a lot of preparation work to optimize the meeting time.

Rule elicitation is an *ongoing* activity you perform throughout the project. *Collaboration* with your stakeholders is critical. They will change their minds as the project proceeds and that's perfectly fine.

It is important at this stage to remember that there are different types of languages for expressing business rules:

- Natural language
- Restricted Language
- Formal expression using a specific grammar

The natural language is initially used during business conversations to describe the rules, informally, without trying to impose any structure, for example with people sitting around a table. At this stage, we do not have any templates or guidelines for structure that we need to abide to. Using this language, we may have redundancy and inconstancy in the rule expressions and in the business terms used.

A second evolution is using a restricted language, still consumable by both analysts and developers, but where we have imposed some structure and grammar to the language so we can express rule statements with proper form. SBVR proposes the *restricted English* for this purpose. The statement may not be correct semantically (redundancy, consistency, etc.), but we can formalize the business term and glossary of terms. Templates such as the one below can be used to also define some meta-data attached to the rule:

Business Activity: use case # decision **Decision:** **Policies:**			
Owner	*Person or team owner of the business policies*		
Candidate rule project	*Used later during the prototyping and building phases*		
Candidate Package	*Sometimes a decision point will be mapped to a group called package and be part of a decision service. A decision service will have multiple packages with the orchestration of execution handled by a rule flow*		
History			
Rule Name	***Rule***		***Comment***
Accident Prone Customer	*Use the raw natural language of the business conversation. Later we may need to use a more strict language like the restricted English of SBVR.* A customer who had an accident report in the past is marked as accident prone		*Use comment for example to describe the type of rule* **inference**
R2	It is necessary that only one deductible be attached to a coverage		
R3			
Business entities referenced	*List the business entities used, this will help to build the conceptual data model*		
Who can change the rules?	*Can be filled during analysis, it helps to understand the velocity of the rule and prepare in the design of the ruleset and the rule governance process*		
When the change can occur?	*Same comment as above.*		

The third type of language is precise and there are no ambiguities: the rule refers exactly to information system objects. This language is parseable and nonambiguous and can be executed by a computer.

A formal language features sentences which have a clear and unambiguous interpretation. There are different kinds of formal languages:

- Predicate logic using syntax like: $(\forall X,Y)$ [Claim(X) Λ MedicalInvoice(Y) Λ Relation(X,Y) => (claimRefNumber(Y) = claimNumber(X))]
- Object Constraint Language (OCL): is an addition to UML to express constraints between objects that must be satisfied
- Truth tables or decision table which present rule as row and columns representing conditions and actions
- Semantics of Business Vocabulary and Business Rules or SBVR which defines structural and operational rules as well a vocabulary to define business concepts
- JRules Technical Rule Language executable by a rule engine
- JRules Business Action Language, high-level language close to English, which is formal as it is using a unique interpretation and unique translation. Rule writers pick among a set of predefined sentences

4.2.2.7 Discovering Rules from SMEs

Interviews and analysis workshops are the two types of interaction used with subject matter expert. For interviews, the typical number of people in the same room is around two or three and for workshops six to ten people are involved. Workshops can last several days. Interviews are used at the beginning of the discovery phase and will most likely address one area of the business process. The analysis workshop is perhaps the most powerful technique for eliciting a lot of requirements. It gathers all key stakeholders together for a short but intensely focused period. The use of a facilitator experienced in requirements management can ensure the success of the workshop. Brainstorming is the most efficient technique used during the sessions.

Brainstorming involves both idea generation and idea reduction. Voting techniques may be used to prioritize the ideas created during a brainstorming session. The workshop facilitator should enforce some rules of conduct during these workshops:

- Do not "attack" other members.
- Do not come back late from a break, even if *key* stakeholders may be late returning because they have other things to do. The sessions are short so they should be able to do other activities during the day.
- Avoid domineering position.

Some authors have suggested the following to improve the process:

- Facilitator keeps a timer for all breaks and fines anyone that is late, everyone gets one free pass.
- Facilitator encourages everyone to use 5-min position statement.
- In case of a long discussion without reaching a firm conclusion or an agreement it is good to use the business concerns to drive the elicitation.

- If a rule is not clear, then it is a good idea to try it out/prototype it.
- Use concrete scenarios to illustrate some rules. These scenarios can later be leveraged for tests.

The following table lists the standard questions the analyst team may ask during the workshop, depending of the source:

Type of input document	Questions	Type of artifacts impacted
Use case or business process map	In this activity, what kind of control the worker is responsible to perform the task? What kind of decisions? On this use case step, the person assess the application, what kind of assessment is he doing? Is there a standard check list?	Use case or BPM Rule description document
Rule description Conceptual data model	What do you mean by .... (a business term to clearly define) How does it relate to .... (other business term)	Conceptual data model
Rule statement	What about the other ranges of possible values for this condition? How often does this condition change? Do you have some other cases?	Business Entities Diagram Rule description document

Between sessions, verify that business terms are well defined and the rules make sense and do not have logical conflicts. Log all the questions related to this analysis in an issue tracking document (Open Points).

4.2.2.8 Discovering Rules from Documents

This approach is used when Governmental Administration or policy group issues legal documents. We did observe this work requires courage and rigor. When using electronic documents, we used the following practices:

- Annotate the document on anything that needs some future discussion
- Copy and paste the business policy declared in the document to the rule template to clearly isolate it for future analysis
- Work in a consistent/systematic way to ensure a good coverage
- Check for agreement with the current business model as you go along
- Investigate discrepancies and log them
- Focus on stakeholder understanding (communication is key) and insist to clarify how a legal rule is interpreted by the line of business

One risk with this approach is that the reader is making his own interpretation of the context, and the document may not include all the cases and criteria leading to interpretations. It is sometimes difficult to get the business motivation behind a written policy. We recommend applying a rigorous method to be able to achieve the following goals:

- Get an exhaustive list of the business events under scope: log them in a table
- Get the activities, tasks, and processes that support the processing of those business events
- Identify where the business rules could be enforced in the process
- Get the business motivation behind the rules
- Get explanation on rules if they are unclear, ambiguous
- Try to extract the object model under scope, domain values by looking at the terms used by the rules . . .

We should still apply agile modeling by involving the SMEs to get feedbacks on the findings, assumptions, and issues. Use simple diagrams to communicate with the project stakeholders.

4.2.2.9 Discovering Rules from Code

Discovering rules from application code is time consuming and does not lead to great results. The analyst needs to study a lot of lines of code and procedures to find the conditional operators which are linked to business decisions. Depending on the design and code structure of the application, this work can be very time consuming. It is important to remember the context of execution when the "if statement" is executed, some variables may change the context of this "business rules." With some languages using limited length to define variable names it is more difficult to relate such variables to business entities. A variable in one procedure can have a different name but the same meaning. Code mining tools exist on the market and help to extract business rules and the variables meanings. It is important to keep in mind that rules written in the past may not be relevant any more. Lastly, as stated previously, most of the rules implemented as procedural function need a deep refactoring before deployment to a rule engine.

Code mining is commonly requested by people as it reassures the business team that the rule harvesting starts by the existing behavior of the legacy code. Code mining is usually better used to confirm behavior of some litigious points identified from using other techniques than to try to extract all of the rules as a whole. Rule discovery with SME, using workshop sessions, may conduct to ambiguities or misconceptions. Trying to understand how the rules are implemented in the current system helps to resolve such situations.

4.2.2.10 Documenting the Business Rules

We suggest that a template as presented above should be used for documenting rule details during the harvesting phase. To document the rule, try to use the language of the business ("problem domain") rather than the language of the technology ("solution domain"). The following rule is as stated by a business user in a car rental industry:

A driver authorized to drive a car of group K must be over 29

A rule developer may think to document the rule as:

If the age of the driver is less than 29 and the requested group of the reservation is K, modify the authorized attribute of the driver accordingly.

As stated above it is important to identify the different languages used to document the rule. The rule statements may evolve with time. We use different templates for documenting rules, depending of the type of discovery roadmap. ABRD includes different templates you can leverage.

4.3 Rule Discovery: Case Study

To illustrate all the concepts described in this book, we use a simplified business process for the claim processing application in a fictional insurance property and casualty company named MyWebInsurance. Currently, the claim processing application is using a mix of a legacy COBOL application, which has been doing an excellent job during recent years using a data processing approach, and packaged applications, which are not easily adaptable to support new requirements. In the last few months, an increase in the number of claims to process has led the business executives to address the following business problems:

- Supporting better user experience by giving clear information on the claim processing state
- Supporting a dramatic increase of the demands: more claims to process without hiring more staff
- Supporting new regulatory rules or financial audit policies that force, for example, to pay a claim within 30 days or to be able to re-play an old processed claim for audit purposes

As business grows, customer quality concerns arise because the legacy application could not easily and quickly be modified to support new demands and changes to the process. We can imagine many more drivers for the change but those important business requirements force the enterprise architect of MyWebInsurance to work on the future evolution of this claim processing application, based on agile technologies such as BPM, BRMS, MDM, ESB, and leveraging a Service Oriented Architecture.

The current business process starts with an *insured person* sending a *claim* or a *medical service provider* sending a *medical invoice* to the company. The following actors or stakeholders are part of the process. Each actor is accessing the current legacy application using different menus depending on his role.

Actor	Role	Type of interface
Claimant	The insured person	Use standard paper forms to fill the claim
Patient	The person related to the insured person who receives medical treatments after an accident	May mail medical invoice
Medical service provider	The medical provider or other service provider who can invoice.	Enter information on a legal paper form.
Mail processor (MP)	The claim or bill request is received by mail, so the mail processor enters some information in the system. The system returns the claim number and the claim processor candidate to handle this claim. The paper form is routed internally to the claim processor	Legacy text-based screen – claim entry
Claim processor (CP)	Responsible to complete the data entry, to make first level of investigation, to pay simple claim, and additionally, some are responsible to analyze customer records and determine if the billed level of service is appropriate	Legacy text-based screen – claim processor access
Claim adjuster (CA)	Responsible for coverage, liability, and damage investigations. The CA authorizes payments for both indemnity and expense payments and is responsible for providing the direction to bring claims to a timely and accurate conclusion	Legacy text-based screen – adjuster access
Branch manager	Manage the claim processing employees within a branch. He is involved in specific claim reviews typically with invoices above a certain dollar amount	Legacy text-based screen – manager access

The process below is a simplified version of a real insurance claim application, but illustrates the major activities we need to consider for our case study. The analyst is using the current application main user menu to initiate the process modeling task.

When MyWebInsurance receives the paper documents, a *Mail Processor* starts the process by looking at the paper sent and by assigning the document to a *Claim Processor*. He assigns a claim number at this moment by using a legacy system to get new claim number. Then the medical invoice or the claim follows a set of activities to assess the customer eligibility, the coverage as defined in the policy and to evaluate the amount of money to pay. The set of issues found by the different applications and by the people are resolved during the life cycle of the claim. This process can take a lot of time and it is possible that the bill may not get paid on time and to get penalties.

The claim validation and coverage verification are completed partially manually by the claim processor visiting a set of screens and data fields to verify if coding is entered correctly. There are some communication protocols, using mail to route the work item to a different person in the process. The file moves from an input basket to the output basket of the claim processor. The process is hard coded

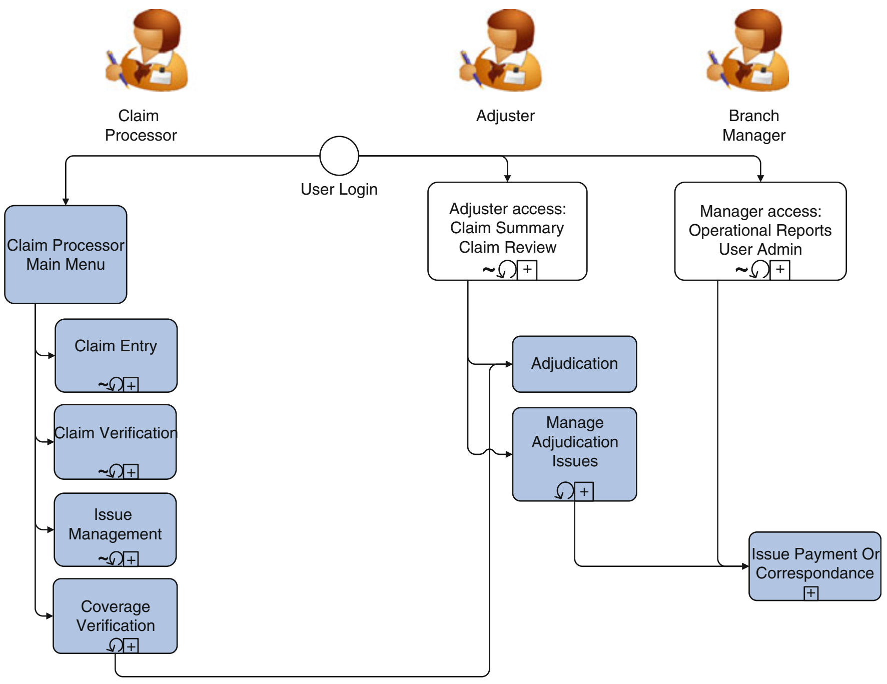
Claim Processor
Adjuster
Branch Manager
User Login
Claim Processor Main Menu
Adjuster access: Claim Summary Claim Review
Manager access: Operational Reports User Admin
Claim Entry
Claim Verification
Issue Management
Coverage Verification
Adjudication
Manage Adjudication Issues
Issue Payment Or Correspondance

Fig. 4.3 As-is claim process as described by claim processor

in the application. Changes are difficult to make and take a long time to release. Some functions are already revamped with a web interface to reduce the cost of maintenance. When the claim is verified successfully, the adjudication can be calculated by one of the adjusters. The process in our case ends with the payment. In fact all the issues are well managed by the application, and correspondence is generated to ask questions to service providers or claimants to get more information. All correspondence is persisted in a legacy data store and can be retrieved from the screen to see the progress of the claim within the process and what information we are waiting for. Figure 4.3 was used to present this process.

We will not spend too much time describing the process in detail, but from this process the analysis team members can define the following *decision point table*:

Decision point name	Description	Source for rule discovery	Current state of automation	Rule owner – SME
Claim verification	Validate that the claim or medical invoice entered in the system contains valid data	Interview SME and insurance legal document and policies like the one related to UB92 legal form	Manual	Adjuster department

(*continued*)

Decision point name	Description	Source for rule discovery	Current state of automation	Rule owner – SME
Coverage verification	The system needs to verify what coverage and deductible apply to the given claim	Interview SME, query the policy data base for coverage and deductible types	Manual	Adjuster department
Adjudication	Claims adjudication in health insurance refers to the determination of a member's payment, or financial responsibility, after a medical claim is applied to the member's insurance benefits	Interview and legal rule	Manual	Adjuster department
Route issue	If there is an issue in the automated process, it will create an issue that needs to be handled manually. Decisions on who to route this issue can be made. Claims can follow at least three paths: • Automatic processing • Exception-issues to be resolved by claim processor • Exception-issues to be resolved by claim manager	Interview claim operator manager	Manual	Management department

We will focus on two decision points: claim validation and claim adjudication. For the claim processing application, the development team decided to apply a rule discovery process based on a business process analysis roadmap. Sources for the rules are divided between expert know-how and some legal documentation and forms. The execution of rule discovery workshops with the different claim processors and managers provided the following important information:

- Claims and medical invoices are received by mail. A "mail processor" (human) assigns a unique claim number and then routes the claim to a claim processor (a person). This data is manually written on the claim. The claim processor enters the claim in the system. The description is based on a simplified version of the UB92 (or HCFA1450) American standard form. In this example, we keep the simplest version so that the process can easily be adapted to other countries. We do not aim to develop a real business application.

Patient control number: xxxxx		Statement covers period From	Through	Admission Date Hour Type Src		Patient name
Type of bill						
Patient's Birthday		Sex	Patient's Address			
Conditions codes:						
RCC	description	Rate	Unit	Serv.date	Total Charge	Non-covered
Payer name				Est. amount due		Due from patient
Provider no.		Provider address				
Insured's name		Insurance company ref		Employer name		Employer location
Treatment authorization codes		Principal procedure Code	Date	Other procedure Code	Date	Other proc. Code Date
remarks				Provider signature		Date

- Certain types of claims are either calculated manually or processed through stand-alone software applications. A small minority of claims are paid in full, requiring no adjudication.

Here is an example of a rule from a legal statements at the back page of the UB92 form: *If the patient has indicated that other health insurance or a state medical assistance agency will pay part of his/her medical expenses and he/she wants information about his/her claim released to them upon their request, necessary authorization is on file.* This rule may land behind the claim validation decision service. Some business terms like patient, other health insurance, claim information, release authorization, need to be integrated in the conceptual data model and somewhere the process needs to include a notification activity to exchange correspondence with the other health insurance party. Studying the UB 92 form leads us to extract the following business entities:

- This form represents a medical bill: MedicalBill. A code supports the type of bill. This code is a legal reference number and can be retrieved from a reference data source.
- Provider name, address, and telephone number are required; the minimum entry is the provider's name, city, state, and ZIP code.
- The patient's unique alphanumeric number assigned by the provider to facilitate retrieval of individual financial records and posting of payment.
- The type of bill is a three-digit alphanumeric, with the first digit specifying the type of facility:
  - 1 – Hospital
  - 2 – Skilled nursing
  - 3 – Home health
  - 4 – Religious nonmedical (hospital)
  - 5 – Religious nonmedical (extended care)
  - 6 – Intermediate care
  - 7 – Clinic or hospital-based renal dialysis facility (requires special information in second digit)
  - 8 – Special facility or hospital ASC surgery (requires special information in second digit below)
  - 9 – Reserved for national assignment

The second digit is for classification outside of a clinic, and the third digit is for frequency.

- The medical bill includes the patient information, like control number (required number assigned by the provider), name, address, and his status related to the insurance policy; patient sex is M or F. The month, day, and year of birth is shown numerically as MMDDYYYY.
- The coverage period: beginning and ending dates of the period of the injury.
- The admission type to identify if this is an emergency (severe, life threatening, or potentially disabling conditions), urgent, elective, or NA.
- The conditions code: codes identifying medical conditions related to this bill which may affect processing.
- The line item includes a medical procedure code, revenue description, a rate, service data ... The medical procedure code is a very important element to identify the type of invoice. A large portion of the rules will have conditions that look at this code.

The claim verification step is started once the claim data entry is completed. The following business policies are extracted from interactions with the different claim processors of the company (Table 4.1):

For the purpose of this sample, we are not developing a complete application. Claim processing represents one of the most difficult applications in the insurance industry. The rules above should help to support the analysis and development of our first ruleset and not support a real-life claim processing application. It is also

Table 4.1 Claim verification rule description

Process step: Validate claim or medical invoice	Called after the claim or medical invoice is entered in the system.	
	Business motivation: Any violation of the following rules will be a rejection of the claim or the medical bill. The claimant needs to provide accurate data. The sooner we can extract data inconsistency the lower will be the cost of processing.	
Rule id	Raw description of the rule	Comment – rule classification
VC01	The Claim should be initiated within 30 working days after the accident	Guidance. We may need to specify a range of possible days when the Claim must be rejected. We can propose 45 days and never after 1 year. Also the number of days could come from the policy
VC02	We need to verify that the accident location is one supported by the policy. For a given product, MyWebInsurance defines different states where the policy applies. But as the customer may want to change this coverage, the list of possible states is attached to the policy	Constraint. The list of supported states is variable Rejection generates issue
VC03	Verify that the person state of residency is one supported by the policy	Constraint. Rejection generates issue.
VC04	The customer insurance policy has a set of coverage with coverage code, which needs to be different from 05 and a business code not equal to 45	Very specific – Constraint. Rejection generates issue
VC05	The claim must be issued before the expiration date of the policy.	Constraint Rejection generates issue
VC06	The date of loss should be before the expiration date of the policy and after the effective date	Guidance. We may need to specify a range of possible days when the Claim must be rejected according to the federal and local laws
VC07	The claim applies on a property covered by the policy (a car, a bicycle)	Constraint
VC08	The billing invoice is always linked to a claim. It includes a reference to a patient and a list of billing item. Each billing item has a procedure code, a quantity, a service data, total charges, and noncovered charges. If any procedure code is unknown raise an issue	Constraint
VC09	When the medical invoice is the first received for a given claim, we need to verify that the date of the earliest service has to be within 1 year after the date of loss, and the invoice should be received before this 1 year delay	
VC10	The first medical treatment should be within 90 days after the date of loss	
VC11	A policy applies to one or more listed drivers. Listed drivers mean the first name and last name provided by the insured person. A claim must come from one of the listed drivers	

Table 4.2 Adjudication rule description

Business Activity: adjudication of medical invoice
Decision:
Policies:

Owner	Adjudication department – Adjudicator director	
Candidate rule project	adjudicateClaimRules	
History		
Rule name	Rule description	Comment
Verify Treatment needs independent medical evaluation	Evaluate if one of the treatments in the medical bill needs an IME by looking at the medical procedure code in table "IMEevaluationNeeded." We want to add some criteria on the invoice amount and later on the service provider	This may be implemented with a decision table to look for each line item the procedure code and the action is to set some IME needed or not with a type of IME request. This resolution will include information such as: Need peer review, need potential peer review, and need claim processor's review The invoice is put on hold, and the action is wait for IME results
Review IME results	If the invoice needs IME, verify we have all the IME results. If not continue to keep invoice on hold and create issue for each missing IME result	
Missed Medical Evaluation Appointments	When there are two or more missing Independent Medical Evaluation appointments, the invoice is set to "grounds for non-cooperation." This should force denial of the entire claim Create an issue for the claim processor to contact the person with the number of appointments missed	
Identify Medical Procedure excluded from Expert Treatment evaluation	Deny any invoice with medical procedure (s) not supported by our expert evaluation	The IME result has Treatments. So we need to verify for each line item of the invoice; the procedure code is the same as the treatment code *This can be accomplished in java or SQL with a join between two collections*
ReviewByLicensed chiropractic	The chiropractic claim must be reviewed by a licensed chiropractor. Procedure code start by "CHIR"	
GoodStanding Chiropractic	The chiropractic consultant must be in good standing and have a current license in the state in which the review is performed with no current license term violations	

(*continued*)

Table 4.2 (continued)

Business Activity: adjudication of medical invoice
Decision:
Policies:

Owner	Adjudication department – Adjudicator director	
Candidate rule project	adjudicateClaimRules	
History		
Rule name	Rule description	Comment
Emergency Treatment after date of loss	Create an audit review if there is a medical treatment given in the emergency room later than 5 days after the accident	ER Treatment after DOL
Ambulance after date of loss	Create an audit review if there is a ambulance transport not on the same day as the accident	Ambulance Treatment not on DOL
Too late medical treatment	If the earliest medical service date of any treatment invoiced is after 1 year of the accident deny the entire invoice and report an issue	Reason is: "First Date of Service is one year past Date of Loss"
Gap in treatments	If there is at least more than 100 days between the earliest medical service date of any treatment invoiced and the last date of medical service on previous invoice, then deny the entire invoice	
Late treatment	Create an issue when the earliest medical service is given 90 days after the date of loss	Reason is: "First Date of Service is a late treatment"
Late invoice	Create an issue when the received date of the medical invoice is 1 year after the date of loss	Reason is "Receipt date of invoice one year past Date of Loss"
Bill not timely	We are rejecting the invoice if there is any line item with a date of service older than 90 days from the date of invoice	
Outpatient reimbursement	The covered outpatient services include the following services, emergency room, ambulatory room, medically necessary outpatient hospital and clinic, radiology, and medical imaging When the invoice is from an hospital and related to an outpatient service, the revenue code needs to be 490 and bill type 83X; any surgical procedure listed in CPT code will be reimbursed accordingly Reimburse the service at the outpatient OMB rate Otherwise claims are reimbursed by multiplying covered charges by the statewide outpatient cost-to-charge ratio	

important to note that those rules are not in their final state, we will transform them using a more formal representation during the analysis phase.

In insurance, claim adjudication refers to the determination of an insured person's payment, or financial responsibility, after a medical claim is applied to the insured's insurance benefits. Most of the time the insurance company will initiate some *expert audit*, called *Independent Medical Evaluation*, to complete the diagnostic of the patient and evaluate the appropriateness of the given treatment. An *Adjustment* is the calculation of the amounts to be made paid by the insurance company. For the "adjudicate claim" decision point, the rule discovery aims to develop the business rules as in Table 4.2 which presents a second type of template.

4.4 Rule Analysis

The goal of the rule analysis activity is to understand the meaning of the rule as stated by the business person and subject matter experts and to remove any ambiguity and semantic issue. The objective is to prepare the rules for the future implementation. As mentioned in the Chap. 3, rule analysis can start as soon as the team has some rule descriptions which are agreed upon by the subject matter experts. The rule analysis phase includes activities such as "analyze the rule description and fact models", "transform the rule", "build test scenario", "design the data model used by the rules", and "synchronize with current logical or physical data models." The flow may look like in Fig. 4.4.

4.4.1 Analyze Rule Descriptions and Fact Models

The first activity focuses on analyzing the rule descriptions to extract the business entities and terms used. During the elicitation activity, the raw description of the

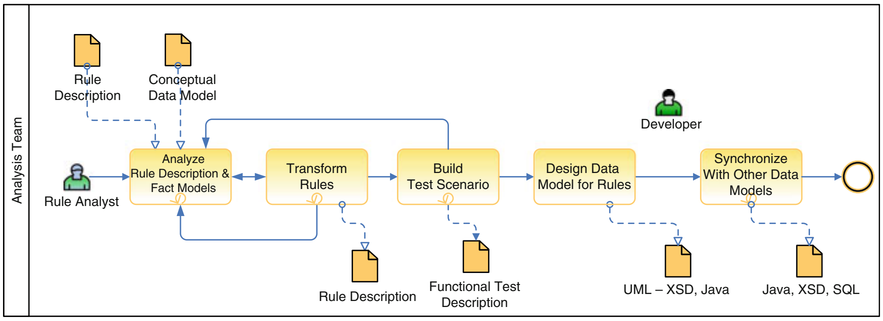

Fig. 4.4 Rule analysis activities

rule uses business terms as used in a common language, used by the people. We are at the expression level used to communicate between humans. At this level, terms have lot of ambiguities. To be able to remove those ambiguities, we need to define the meaning of the concepts used and link them by formal propositions.

W3C has produced important specifications to define semantic models which could be used to define the data model used by the rules. As mentioned earlier, OWL (Ontology Web Language) helps to define the enterprise domain model or ontology. The ontology describes the concepts in the domain and the relationships between them. It can vary from taxonomy to conceptual model. OWL leverages W3C-RDF, the Resource Description Framework, to persist the semantics of the things to be described in XML format. A reasoning engine can check the internal consistency of the statements and definitions in the ontology. It can also *classify* concepts by finding the definitions/categories under which they fit. Many companies interested in business rules are also considering developing an enterprise ontology with OWL-RDF. The adoption of such standards and emerging tools will help to develop the complete semantics needed for enterprise data models.

Using a more traditional approach, Object-Oriented Analysis (OOA) describes *what* the system is by using a set of models of the system as a group of interacting objects. Each object represents some entity of interest and is defined with a Class. A class includes attributes and behaviors as methods. OOA models are typically represented with UML use case diagrams, UML class diagrams, and a number of UML interaction diagrams. Using an OOA approach, we can model the concepts used by the business in static class diagrams to use a formal notation. At this stage of the methodology, we may have the following possible data models in our hands:

- A conceptual data model
- A logical data model
- A physical data model
- Some reference data which describes static list, enumerated value, classifications, and the like

Those models are not finalized, and they are enhanced and transformed during future iterations of the harvesting and prototyping phases. Most projects have already some data models, and database analysts contribute to explain how some business concepts are mapped to physical data model elements. When using a rule approach, it makes sense to start by the rule description and define the data model from there and *not* to start from an existing physical data model. We need a view of the current data model and a clear definition of the terms used by the rules; we do not need a complex data model to express the rules. Rule analysts have to extract the business terms from the rule statement and build a business glossary. We did observe that this glossary brings a lot of value to the line of business as terms are defined without ambiguity and interpretation. Business terms and their relationships can be represented in a conceptual data model (CDM) or entities diagram. The steps to perform the analysis are:

- Highlight the nouns used in each rule description. We are talking about Terms, as referring to a business concept used in daily business operations. It can be one or more words and nouns. They are often differences found in between departments, and each department may refer to the same business concept but defines it using different perspective and hence different words. These are actually synonyms. Examples of term are: a taxpayer, a taxpayer obligation, a loan, a claim, a legal entity, an application, a customer, a product, etc.
- Analyze facts: A fact is a statement that connects terms into a business-relevant relationship. Some examples of facts: A taxpayer files a tax return form; the customer could have only one purchase order at a time; a medical invoice is linked to one claim. The fact has to be analyzed to understand how the application will support it. It can be through use case implementation, business rule, or the relation between objects.
- Build the facts declaration to define the used terms. It is possible at this level to represent entities in a model using a diagram such as a UML class diagram (see Figs. 4.5 and 4.6).
- Map it within a conceptual data model diagrams.

A term may describe a business concept which will be mapped to a Class, a characteristic of a business entity which will be mapped to an attribute of a class, and sometimes a term may describe the way a business object behaves. In that last case, it will be mapped within a method of a finite state machine. As an example we can take the following rule description:

Adjusters reject the invoice if any line item has a date of service older than 90 days from the date of invoice.

From this rule statement, an analyst can build the following facts:

- Adjuster is an employee of the insurance company.
- Invoice is a medical invoice.
- A medical invoice has a date of invoice.
- A medical invoice has at least one line item.
- A line item describes a medical treatment.
- A medical treatment has a date of service.
- A medical treatment must have one unique procedure code, a quantity which is at least one (and most likely a price but we do not know yet).

The creation of facts may generate new rules. Here we can add the following business rules into the scope:

- If the claim is related to a loss, the date of loss has to be provided.
- The insured person must have a residency in the USA.
- The patient name, address, and status must be on the medical invoice.

It is clear that we can add a lot of facts to link terms in our model. And we can spend months of documentation doing so. In Agile Business Rule Development, we prefer having working rules and light documentation. Such facts can be presented by a set of diagrams, which will help us communicate with business users. Those diagrams evolve later to class diagrams from which we can generate code. Diagrams are always a good vehicle for communication. It is also important to make different diagrams for different audiences, but to also maintain them in synchronous manner. We propose to keep the conceptual model as a set of diagrams to communicate to the business user. These diagrams evolve with the rule harvesting phase.

It is important to note that not all the rules can be implemented and deployed into a software component. Some rules may end in a procedure manual delivered to the worker to enforce a good business practice as defined by corporate policies.

4.4.2 Transforming Rules

The activity "Transform Rules" leads to modifying rule declarations so that they become formal, atomic, and standalone elements. This is needed for understandability and ease of implementation and maintenance. This activity also includes understanding the rule patterns, eventually removing redundant rules, or resolving overlaps among rules. This activity is also conducted during the implementation of the rules, but it is started during the analysis, so we are detailing the approach in this context. The key concept is to transform rules to an atomic level as much as possible.

***Concept*: Atomic Rule**
A rule is atomic if it cannot be further decomposed without losing meaning. Atomicity is desired for understandability, ease of maintenance and execution efficiency.
The following rule statement can be decomposed into two rules. From

> *The insurance does not reimburse medical expenses incurred abroad if the claim is presented more than one year after the expenses had been incurred, or if the claimant has spent more than 182 days abroad within the past year.*

to

- *When the date of creation of the claim is more than one year after the date of treatment of the medical expense then reject the medical expense.*
- *When the claimant spend more than 182 days abroad within the past year then reject the claim.*

Rule conditions are true or false and should lead to one result. The rule analyst has to clearly understand the Boolean logic.

***Concept*: Boolean Logic Summary**[6]

AND/Conjunction

The conjunction of two propositions is true when both propositions are true. The truth table is:

AND	A	True	False
B			
True		*True*	*False*
False		*False*	*False*

Another notation is using the dot operator for AND so A.B is equivalent to A AND B.

OR/Disjunction

Disjunction of two propositions is false when both propositions are false.

OR	A	True	False
B			
True		*True*	*True*
False		*True*	*False*

Another notation for disjunction is using the operator + for A OR B like A + B.

NOT/Negation

A	NOT A
True	*False*
False	*True*

Implication

A→B, implication is a binary operation which is false when A is true and B is false. A→B can be expressed as NOT A OR B.

A→B	A	True	False
B			
True		*True*	*True*
False		*False*	*False*

XOR or exclusive OR

Exclusive OR of two propositions is true just when exactly one of the propositions is true:

XOR	A	True	False
B			
True		*False*	*True*
False		*True*	*False*

[6]See also http://en.wikipedia.org/wiki/Introduction_to_Boolean_algebra; http://www.internettutorials.net/boolean.asp.

De Morgan's Law

De Morgan's law represents rules in formal logic relating pairs of dual logical operators in a systematic manner expressed in terms of negation:

NOT (A AND B) = NOT A OR NOT B

NOT (A OR B) = NOT A AND NOT B

De Morgan's law can be used to improve rules during the rule transformation activity.

To refine rules to the atomic level, the rule analyst has to apply some transformation patterns. For example, when a rule is an inference or an action enabler, it may be important to consider separating expressions linked with ANDs within the action part of the rule (also named the right hand side).

A pattern of criteria organized such as:

IF condition_A THEN do (B) AND do(C) may be rewritten as two rules to make them atomic

IF A THEN do (B)
IF A THEN do (C).

This is due to the fact that a change in the data used as part of the conditions of a rule may force the reevaluation of all rules using such data. We will detail the rule engine's RETE algorithm in Chap. 6. In the first schema nothing happens before the end of the action. So if the action B makes the condition A false, C is executed when it really should not.

When expressing an inference rule or an action enabler, do not allow ORs on the left hand side of the rule (the condition part); break the rule.

A pattern like:

IF A OR B THEN do(C) can be rewritten as two rules to make them atomic

IF A THEN do(C)
IF B THEN do(C).

This is a good practice when the conditions A and B are complex. A simple condition like the age is 18 or 21 does not need to be separated into two rules. Also the semantic of the OR has always to be assessed. It could be that the subject matter expert means an exclusive OR. In that last case the rules are:

If A and Not B then do(C)
If B and Not A then do (C)

When expressing constraints (must, have to) and guidelines (should), try to remove ANDs between conditions and clearly separate them in different rules.

A business policy like:

A driver must be 25 years old or older AND must have good credit rating

May be split into two constraints like:

Rule 1: A driver must be at least 25 years old
Rule 2: A driver must have good credit rating

The goal is to clearly separate the constraints. The action part of the rule will most likely raise an issue. It may be more efficient to have all the issues the business transaction is violating. Here we want to see the issues reported about a bad credit and a young driver.

Make sure that each rule contains only necessary conditions; do not over-constrain the rule applicability. The rule analyst has also to look for redundant rules and try to remove them. Redundant rules are duplicated rules, duplicated through some transformations (renaming, inversion of conditions, etc.), and redundancies among rules that create a common data value or a common truth value, or initiate a common action.

Removing redundancy is simpler if rules are atomic, otherwise analyst may get lost in the equivalence of complex logical formulas (e.g., If NOT (A AND B) is equivalent to IF (NOT A) OR (NOT B)). There are subtle forms of redundancy: IF A AND B THEN C is equivalent to IF (NOT C) THEN (NOT A) OR (NOT B). Sometimes changing the order of conditions can help highlight identical rules: IF A AND B THEN C is the same as IF B AND A THEN C. This looks obvious as written like a mathematical expression, but depending on the rule language it may be difficult to see at first reading.

Another step of the analysis is to remove inconsistent rules. Overlapping rules are partially redundant because they are not semantically equivalent but they point to problems: one rule may say IF A AND B THEN C, the other says IF A THEN C. The question will be: is B really needed to infer C? One of the two rules should be eliminated or modified to fix the inconsistency.

It is also possible to get semantically equivalent conditions with contradictory conclusions: two rules like IF A THEN B; and IF A THEN NOT(B) are two conflicting rules, probably due to two different sources of information for documenting the decisions. Typically, this is symptomatic of the fact that we are missing some necessary conditions in either rule (or both, e.g., IF A AND C THEN B; IF A AND D THEN NOT(B)).

Another pitfall are rules that lead to the same conclusion based on contradictory conditions: rules like IF A THEN B and IF NOT (A) THEN B. Logically, this means that the conclusions should always be true. This is symptomatic of the fact that the condition is not really relevant to the conclusion.

The analysis has to ensure the completeness of the rules. We may consider three kinds of completeness:

- Make sure that all the possibilities are covered for a given rule pattern. If you have a rule that says "loans for value greater than $250,000 should be approved by the branch manager", it does not tell us who must/can approve loans of value less than $250,000.
- Make sure that all derived data in the object model has corresponding computation or inference rules. This involves computed attributes, qualifications (e.g., customer status, account type, etc.).
- Make sure that integrity and cardinality constraints are somehow represented. Either in the object model or in rules.

The analysis phase is a good time to ask the business user how often the rule will change, we call this rule volatility. Rules about risk computation, eligibility, underwriting, or configuration may change over time. We notice that when a user does not anticipate the rule changing, rules unplanned at the beginning are added over time. Some rules may not change often but other rules in the same ruleset may. Moving the "non changing" rules outside of the ruleset may have bad impacts on the ruleset integrity. When looking at rule volatility, it is important to assess which factors trigger rule changes and how new rules are defined for a given decision point.

Lastly, the rule analyst needs to understand the rule dependencies and rule sharing goals. A rule R1 depends on a rule R2 if the enforcement of R2 results into a situation where R1 is relevant (or needs to be enforced). A simple example is a rule R2 which is creating new data or is modifying existing data that is tested by R1.

Rule sharing is a more complex concept to implement and may be linked to the BRMS capability. The goal is to avoid to copy and paste the same logic across rulesets. One ruleset can include a set of rules that are common to multiple ruleset. For example, testing the age of a customer can be put in a common rulesets. The other rulesets are referencing the common one, and rules are shared. A possible side effect of rule sharing is rule overriding: a specific rule in one ruleset takes precedence over another rule in a common ruleset. The overriding enforcement is most likely done using some meta-properties attached to the rule.

Understanding dependencies help determine the likely "execution" sequence of rules. The execution sequence is useful for rule analysis to detect undesirable dependencies. For the implementation, the execution sequence is useful to understand what the results will look like: some rule engine determines that sequence automatically and on the fly (chaining). If we implement business rules in a procedural fashion, we need to understand the execution sequence to enforce it. Some of the undesirable dependencies include circular dependencies leading to infinite loops.

4.4.3 Building Test Scenarios

Developing software without testing makes no sense in today's world (we hope!). Developing rules deployed in a rule engine helps developers to efficiently support a Test Driven Development (TDD) approach. Writing tests before authoring the rule makes testing part of a validation feedback loop. During the harvesting phase, the analysis team needs to develop test scenarios and data elements to support the future rule writing and testing. The development of concrete scenarios leads to the clarification of ambiguities, finds holes in the decision processing, enhances rules decision coverage, and the overall quality. Implemented rules are software elements like methods and classes in object-oriented development: we may define tests for each rule. Concrete scenarios may be written as a story board. Start by a simple case and then add more data elements to cover specific rule.

Here is an example of user story: *Jack Bee living in California and customer of WebInsurance for 3 years as a good driver. He filed a claim for a minor car accident where his friend Mark, located on the right seat, was slightly injured at the neck. Mark went to the hospital one day after the accident and he follows up with his medical provider. The hospital and the medical provider are sending invoices to Webinsurance. One medical invoice includes neck massage with a date of treatment six months after the date of the accident. The invoice should be rejected.*

With Test Driven Development (TDD) we write a single test before writing the rule which fulfills that test. Basically, the rule writer executes the following steps:

- Add test by specifying the data and expected results
- Run the tests to ensure that the new test does in fact fail
- Create or update the rule or rules so that they pass the new test
- Run the test suite again to verify the test now succeeds

The advantage of TDD is to write rules by small increments, which is safer than writing a complete ruleset without testing. Another advantage is that it helps design the code, the rules, and how exceptions are reported.

4.4.4 Verify Rules Against the Data Models

The rule analyst needs to continuously verify that business terms used in rule statements are part of the logical data model as classes or attributes. The model exposed to the rules needs to get data from data sources. If a concept is not in the data, it has to be quickly handled and managed by the application architect. So this activity of synchronizing the work done at the model level with the different existing data models is a very important task of any business rule project. Most of the time a concept has different names, but sometimes a new concept may force adding a new column in a table.

4.5 Case Study: Rule Analysis

Back to the WebInsurance claim processing application, the rules in Tables 4.1 and 4.2 are analyzed and completed after discussions with the SMEs. The following facts are added:

- The claim must reference one insurance policy.
- The insurance policy has at least one insured person.
- A patient is also called an involved party.
- An involved party is a legal entity involved in a loss; he could be the insured person.
- An accident is a loss.
- A policy is an insurance policy.
- An insurance policy lists a set of coverage.
- A coverage has a unique coverage code.
- Coverage is the amount of protection against loss.
- A deductible is the amount the insured must pay when a loss occurs.
- An insurance policy has one effective date and one expiration date.
- Claim has a date of creation.
- The insured person has one or more properties covered by one policy.
- A medical invoice includes code to define the type of bill.
- A medical bill is synonymous as a medical invoice.
- A medical bill includes the patient information.
- A medical bill includes a control number (required number assigned by the provider).
- A medical invoice is issued by one medical provider.
- A medical invoice includes a cover period.
- The cover period has beginning and ending dates of the period included in the bill.

From these facts, we can build the following conceptual data model. This model is closed to a UML class diagram, but is used as a tool to communicate the business concepts with the SMEs and the IT team. This is important to use this artifact to present the data model used by the rules (Fig. 4.5).

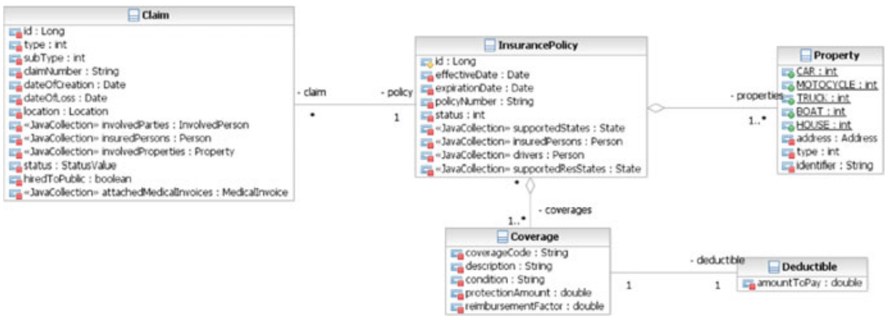

Fig. 4.5 Claim conceptual data model (CDM)

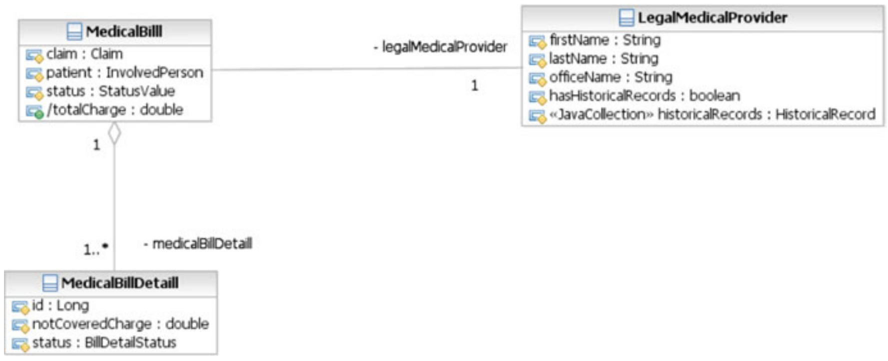

Fig. 4.6 Invoice conceptual data model (CDM)

For the medical invoice or medical bill, the model looks like as shown in Fig. 4.6.

These diagrams are not complete, but we have enough elements to prototype some rules. All these model artifacts help us build a common vocabulary with a structure and syntax we can quickly leverage for our implementation.

4.6 Summary

Rule discovery should not be performed in one long session running for weeks to produce only documentation. Rule harvesting starts at the beginning of the project but is supported with the rule analysis and rule authoring, so that the work produces executable rules and nonexecutable rules. There are cases where the business rules have to be coded in the core application, in the data model structure, or in components other than a rule engine. The rule documentation can indicate where the rule is implemented or enforced. The rule description uses the language of the business and not a technical language.

Rule Analysis is a very important activity in the ruleset development life cycle as it prepares the rules for a successful implementation. Focusing on the data model, the rule semantics and the process flow can help to determine where to implement the business rules. Analysis should not be limited to paper work, but should also use UML tools and even the rule IDE. This should not be a long activity as we are proposing to quickly move to the next phase of rule prototyping. It is easier to find issues related to rule expressiveness or the data model by implementing rules, not by writing extensive documentation.

4.7 Further Reading

Barbara von Halle's STEP methodology, presented in her first book "*Business Rules Applied: Building Better Systems Using the Business Rules Approach*" (2001), does a great job with rule discovery and analysis, both in terms of identifying the different discovery and analysis activities and in proposing effective techniques for performing them. The techniques presented here are *largely* based on STEP.

Tony Morgan's book "Business Rules and Information Systems: Aligning IT with Business Goals" proposes three rule discovery roadmap families depending on rule sources (SMEs, documents, and code), and much of our discussion of those (sections 2.2.7, 2.2.8, and 2.2.9) is inspired from that book.

Two of the main contributors on decision management and decision service approach are James Taylor and Neil Raden with their book "*Smart Enough Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions*" – Prentice Hall (2007).

The Object Management Group (http://www.omg.org) has defined the Semantic of Business Vocabulary and Rules specification, which can be read at http://www.omg.org/spec/SBVR/1.0/.

The OMG also specifies an important framework to define a business motivation model, where the specification can be read at http://www.omg.org/spec/BMM/1.1/.

Detailed about the W3 "OWL Web ontology Language" (OWL) and Resource Description Framework (RDF) can be found at http://www.w3.org/TR/owl-features/ and at http://www.w3.org/TR/2004/REC-owl-features-20040210/#ref-rdf-schema.

Conceptual data model is introduced at en.wikipedia.org/wiki/Conceptual_schema and at http://www.agiledata.org/essays/dataModeling101.html.