Chapter 5

Playing by the rules

Every organization operates according to an extensive set of policies, laws, and industry standards. Industries such as banking, aviation, and medical device manufacture must comply with volumes of government regulations. Such controlling principles are known collectively as business rules or business logic. Business rules often are enforced through manual implementation of policies and procedures. In many cases, though, software applications also need to enforce these rules.

Most business rules originate outside the context of any specic software application. The corporate policy requiring annual training in handling hazardous chemicals applies even if all chemical purchasing and dispensing is done manually. Standard accounting practices were in use long before the digital computer was invented. Because business rules are a property of the business,they are not in themselves software requirements. However, business rules are a rich source of requirements because they dictate properties the system must possess to conform to the rules. As Figure 1.1 in Chapter 1, The essential software requirement showed, business rules can be the origin of several types of requirements. Table 5.1 illustrates and provides examples of how business rules inuence several types of requirements.

Table 5.1: How business rules can inuence various types of software requirements.

Requirement type	Illustration of business rules' influence	Example
Business requirement	Government regulations can lead to necessary business objectives for a project.	*The Chemical Tracking System must enable compliance with all federal and state chemical usage and disposal reporting regulations within five months.*
User requirement	Privacy policies dictate which users can and cannot perform certain tasks with the system.	*Only laboratory managers are allowed to generate chemical exposure reports for anyone other than themselves.*
Functional requirement	Company policy is that all vendors must be registered and approved before an invoice will be paid.	*If an invoice is received from an unregistered vendor, the Supplier System shall email the vendor editable PDF versions of the supplier intake form and the W-9 form.*
Quality attribute	Regulations from government agencies, such as OSHA and EPA, can dictate safety requirements, which must be enforced through system functionality.	*The system must maintain safety training records, which it must check to ensure that users are properly trained before they can request a hazardous chemical.*

People sometimes confuse business rules with business processes or business requirements. As you saw in Chapter 3, Establishing the business requirements, a business requirement states a desirable outcome or a high-level objective of the organization that builds or procures a software solution.
Business requirements serve as the justication for undertaking a project. A business process describes a series of activities that transform inputs into outputs to achieve a specic result. Information systems frequently automate business processes, which could lead to eciencies and other benets that achieve stated business requirements. Business rules inuence business processes by establishing vocabulary, imposing restrictions, triggering actions, and governing how computations are carried out. The same business rule could apply to multiple manual or automated processes, which is one reason why it's best to treat business rules as a separate set of information.

As an example, your organization likely has security policies that control access to information systems. Such policies might state the minimum and maximum length and the allowed characters in passwords, dictate the frequency of required password changes, state how many failed login attempts a user gets before his account is locked, and the like. Applications that the organization develops should apply these policiesthese business rulesconsistently. Tracing each rule into the code that implements it makes it easier to update systems to comply with changes in the rules, such as altering the required frequency of password changes. It also facilitates code reuse across projects.

5.1 A business rules taxonomy

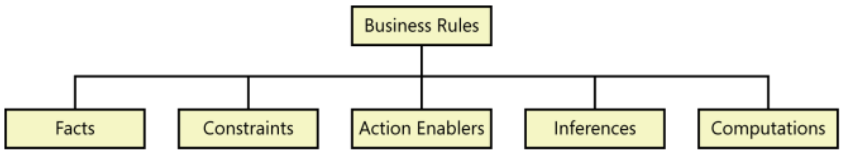
The Business Rules Group [21] provides denitions for business rules from the perspectives of both the business and its information systems:

- From the business perspective: A business rule is guidance that there is an obligation concerning conduct, action, practice, or procedure within a particular activity or sphere. (There ought to be an explicit motivation for the rule, as well as enforcement methods and an understanding of what the consequences would be if the rule were broken.)
- From the information system perspective: A business rule is a statement that denes or constrains some aspect of the business. It is intended to assert business structure or to control or inuence the behavior of the business.
-

Whole methodologies have been developed based on the discovery and documentation of business rules and their implementation in automated business rules systems ([22]; [23]; [24]). Unless you're building a system that is heavily rules-driven, you don't need an elaborate methodology. Simply identify and document the rules that pertain to your system and link them to the specic requirements that implement them.

Numerous classication schemes have been proposed for organizing business rules ([25]; [26]; [22]; [27]). The simple taxonomy shown in Figure 5.1, with ve types of rules, will work for most situations. A sixth category is terms, dened words, phrases, and abbreviations that are important to the business. You could group terms with factual business rules. A glossary is another convenient place to dene terms.

Figure 5.1: A simple business rule taxonomy.



Recording the business rules in a consistent way is more important than having heated arguments about precisely how to classify each one. However, a taxonomy is helpful to identify business rules you might not have thought of otherwise. Classifying the rules also gives you an idea of how you might apply them in a software application. For instance, constraints often lead to system functionality that enforces the restrictions, and action enablers lead to functionality to make something happen under certain conditions.

5.1.1 Facts

Facts are simply statements that are true about the business at a specied point in time. A fact describes associations or relationships between important business terms. Facts about data entities that are important to the system might appear in data models. (See Chapter 8, Specifying data requirements, for more about data modeling.) Examples of facts include the following:

- Every chemical container has a unique bar code identier.
- Every order has a shipping charge.
- Sales tax is not computed on shipping charges.
- Nonrefundable airline tickets incur a fee when the purchaser changes the itinerary.
- Books taller than 16 inches are shelved in the library's Oversize section.

Of course, there are countless facts oating around about businessesCollecting irrelevant facts can bog down business analysis. Even if they're true, it might not be obvious how the development team is to use the information. Focus on facts that are in scope for the project, rather than trying to amass a complete collection of business knowledge.Try to connect each fact to the context diagram's inputs and outputs, to system events, to known data objects, or to specic user requirements.

5.1.2 Constraints

A constraint is a statement that restricts the actions that the system or its users are allowed to perform. Someone describing a constraining business rule might say that certain actions must or must not or may not be performed, or that only certain people or roles can perform particular actions.
Following are some examples of constraints with various origins.

5.1.2.1 Organizational policies

- A loan applicant who is less than 18 years old must have a parent or a legal guardian as cosigner on the loan.
- A library patron may have a maximum of 10 items on hold at any time.
- Insurance correspondence may not display more than four digits of the policyholder's Social Security number.

5.1.2.2 Government regulations

- All software applications must comply with government regulations for usage by visually impaired persons.
- Airline pilots must receive at least 8 continuous hours of rest in every 24-hour period.
- Individual federal income tax returns must be postmarked by midnight on the first business day after April 14 unless an extension has been granted.

5.1.2.3 Industry standards

- Mortgage loan applicants must satisfy the Federal Housing Authority qualification standards.
- Web applications may not contain any HTML tags or attributes that are deprecated according to the HTML 5 standard.

Constraining business rules can convey implications for software development even if they don't translate directly into functionality. Consider a retail store's policy that only supervisors and managers are allowed to issue cash refunds larger than $50. If you're developing a point-of-sale application for use by store employees, this rule implies that each user must have a privilege level. The software must check to see if the current user is of sufficiently high privilege level to perform certain actions, such as opening the cash register drawer so a cashier can issue a refund to a customer.

Because many constraint-type business rules deal with which types of users can perform which functions, a concise way to document such rules is with a roles and permissions matrix [16]. Figure 5.2 illustrates such a matrix for various users of a public library's information system. The roles have been separated into employees and non-employees. The system functions are grouped into system operations, operations dealing with patron records, and operations involving individual library items. An X in a cell indicates that the role named in the column has permission to perform the operation shown in the row.

Figure 5.2: Constraining business rules sometimes can be represented in a roles and permissions matrix.

Roles and Permissions Matrix	Employee	Administrator	Circulation Staff	Library Aide	Non-Employee	Volunteer	Patron
System Operations							
Log in to library system		X	X	X			
Set up new staff members		X					
Print hold pick list		X	X	X			
Patron Records							
View a patron record		X	X				
Edit a patron record		X	X				
View your own patron record		X	X	X		X	X
Issue a library card		X	X				
Accept a fine payment		X	X				
Item Operations							
Search the library catalog		X	X	X		X	X
Check out an item		X	X				
Check in an item		X	X	X		X	
Route an item to another branch		X	X	X		X	
Put an item on hold		X	X	X		X	X

5.1.3 Action enablers

A rule that triggers some activity if specic conditions are true is an action enabler. A person could perform the activity in a manual process. Alternatively, the rule might lead to specifying software functionality that makes an application exhibit the correct behavior when the system detects the triggering event. The conditions that lead to the action could be a complex combination of true and false values for multiple individual conditions. A decision table (described in Chapter 7, A picture is worth 1024 words) provides a concise way to document action-enabling business rules that involve extensive logic. A statement in the form If <some condition is true or some event takes place>, then <something happens> is a clue that someone might be describing an action enabler.
Following are some examples of action-enabling business rules for the Chemical Tracking System:

- If the chemical stockroom has containers of a requested chemical in stock, then oer existing containers to the requester.

- On the last day of a calendar quarter, generate the mandated OSHA and EPA reports on chemical handling and disposal for that quarter.
- If the expiration date for a chemical container has been reached, then notify the person who currently possesses that container.

Businesses often develop policies that are intended to enhance their commercial success. Consider how an online bookstore might use the following business rules to try to stimulate impulse purchases after a customer has asked to buy a specic product:

- If the customer ordered a book by an author who has written multiple books, then oer the author's other books to the customer before completing the order.
- After a customer places a book into the shopping cart, display related books that other customers also bought when they bought this one.

5.1.4 Inferences

Sometimes called inferred knowledge or a derived fact, an inference creates a new fact from other facts. Inferences are often written in the if/then pattern also found in action-enabling business rules, but the then clause of an inference simply provides a piece of knowledge, not an action to be taken. Some examples of inferences are:

- If a payment is not received within 30 calendar days after it is due, then the account is delinquent.
- If the vendor cannot ship an ordered item within ve days of receiving the order, then the item is considered back-ordered.
- Chemicals with an LD50 toxicity lower than 5 mg/kg in mice are considered hazardous.

5.1.5 Computations

The fth class of business rules denes computations that transform existing data into new data by using specic mathematical formulas or algorithms. Many computations follow rules that are external to the enterprise, such as income tax withholding formulas. Following are a few examples of computational business rules written in text form.

- The domestic ground shipping charge for an order that weighs more than two pounds is $4.75 plus 12 cents per ounce or fraction thereof.
- The total price for an order is the sum of the price of the items ordered, less any volume discounts, plus state and county sales taxes for the location to which the order is being shipped, plus the shipping charge, plus an optional insurance charge.

- The unit price is reduced by 10 percent for orders of 6 to 10 units, by 20 percent for orders of 11 to 20 units, and by 30 percent for orders of more than 20 units.

Representing the details of computations in natural language like this can be wordy and confusing.
As an alternative, you could represent these in some symbolic form, such as a mathematical expression or in a table of rules that is clearer and easier to maintain. Table 5.2 represents the previous unit-price discount computation rule in a clearer fashion.

Table 5.2: Using a table to represent computational business rules

ID	Number of units purchased	Percent discount
DISC-1	1 through 5	0
DISC-2	6 through 10	10
DISC-3	11 through 20	20
DISC-4	More than 20	30

5.1.6 Atomic business rules

A better strategy is to write your business rules at the atomic level, rather than combining multiple details into a single rule. This keeps your rules short and simple. It also facilitates reusing the rules, modifying them, and combining them in various ways. To write inferred knowledge and action-enabling business rules in an atomic way, don't use or logic on the left-hand side of an if/then construct, and avoid and logic on the right-hand side [22]. You might break that complex library rule down into several atomic business rules, as shown in Table 5.3.
These business rules are called atomic because they can't be decomposed further. You will likely end up with many atomic business rules, and your functional requirements will depend on various combinations of them.

Table 5.3: Some atomic business rules for a library

ID	Rule
Video.Media.Types	DVD discs and Blu-ray Discs are video items.
Video.Checkout.Duration	Video items may be checked out for one week at a time.
Renewal.Video.Times	Video items may be renewed up to two times.
Renewal.Video.Duration	Renewing a checked-out video item extends the due date by three days.
Renewal.HeldItem	A patron may not renew an item that another patron has on hold.

To illustrate how using atomic business rules facilitates maintenance, when the next generation video technology comes along,or the library purges all of its DVD discs, the library could just update the Video.Media.Types rule and none of the others are aected.

5.2 Documenting business rules

Because business rules can inuence multiple applications, organizations should manage their rules as enterprise-level assets. A simple business rules catalog will suce initially. If you're using a requirements management tool, you can store business rules as a requirement type, provided they are accessible to all of your software projects. Large organizations or those whose operations and information systems are heavily business-rule driven should establish a database of business rules.// Commercial rule-management tools become valuable if your rules catalog outgrows a solution using a word processor, spreadsheet, Wiki, or other collaboration tool. Some business-rule management systems contain rules engines,which can automate the implementation of the rules in your applications. The Business Rules Group [21] maintains a list of products for managing business rules.As you identify new rules while working on an application, add them to the catalog rather than embedding them in the documentation for that specic application orworseonly in its code.
Rules related to safety, security, nance, or regulatory compliance pose the greatest risk if they are not managed and enforced appropriately.

As you gain experience with identifying and documenting business rules, you can apply structured templates for dening rules of dierent types ([23]; [22]). These templates describe patterns of keywords and clauses that structure the rules in a consistent fashion.They also facilitate storing the rules in a database, a commercial business-rule management tool,or a business rules engine. Sets of related rules can also be represented by using tools such as decision trees and decision tables (particularly when complex logic is involved) and roles and permissions matrices. To begin, though, try the simple format illustrated in Table 5.4 [6].

Table 5.4: Some sample business rules catalog entries

ID	Rule definition	Type of rule	Static or dynamic	Source
ORDER-5	If the customer ordered a book by an author who has written multiple books, then offer the author's other books to the customer before completing the order.	Action enabler	Static	Marketing policy XX
ACCESS-8	All website images must include alternative text to be used by electronic reading devices to meet accessibility requirements for visually impaired users.	Constraint	Static	ADA Standards for Accessible Design
DISCOUNT-13	A discount is calculated based on the size of the current order, as defined in Table BR-060.	Computation	Dynamic	Corporate pricing policy XX

Giving each business rule a unique identier lets you link requirements back to a specic rule. For instance, some templates for use cases contain a eld for business rules that inuence the use case.
Instead of including the rule denition in the use case description, simply enter the identiers for the relevant rules. Each ID serves as a pointer to the master instance of the business rule. This way you don't have to worry about the use case specication becoming obsolete if the rule changes.

The Type of rule column identies each business rule as being a fact, constraint, action enabler, inference, or computation.
The Static or dynamic' column indicates how likely the rule is to change over time. This information is helpful to developers. If they know that certain rules are subject to periodic change, they can structure the software to make the aected functionality or data easy to update. Income tax calculations change at least every year. If the developer structures the income tax information into tables or a database, rather than hard-coding it into the software, it's a lot easier to update those values when necessary. It's safe to hard-code laws of nature, such as calculations based on the laws of thermodynamics; laws of humans are much more volatile.
The nal column in Table 5.4 identies the source of each rule. Sources of business rules include corporate and management policies, subject matter experts and other individuals, and documents such as government laws and regulations. Knowing the source helps people know where to go if they need more information about the rule or need to learn about changes.

5.3 Discovering business rules

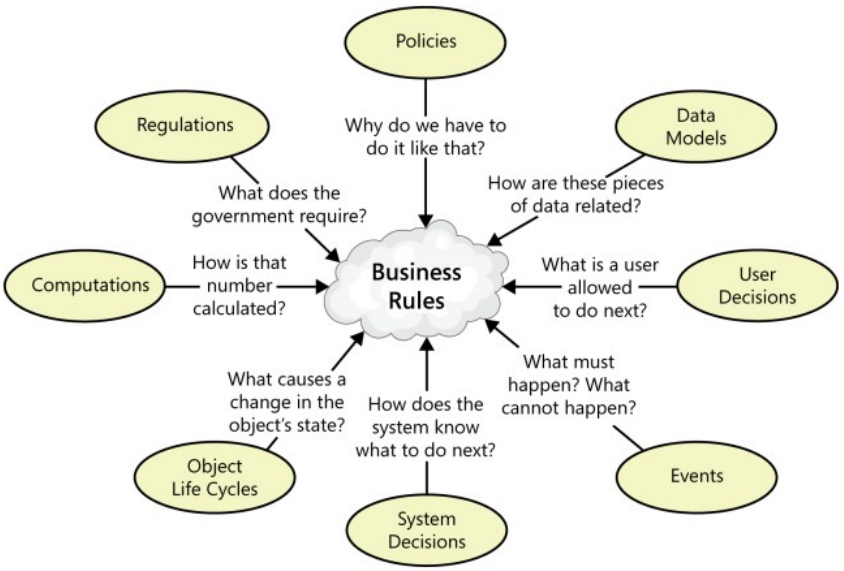
Sometimes you invent business rules as you go along, sometimes they come up during requirements discussions, and sometimes you need to hunt for them. Barbara von Halle [22] describes a comprehensive process for discovering business rules. Following are several common places and ways to look for rules [28]:

- Common knowledge from the organization, often collected from individuals who have worked with the business for a long time and know the details of how it operates.
- Legacy systems that embed business rules in their requirements and code. This requires reverse-engineering the rationale behind the requirements or code to understand the pertinent rules. This sometimes yields incomplete knowledge about the business rules.
- Business process modeling, which leads the analyst to look for rules that can aect each process step: constraints, triggering events, computational rules, and relevant facts.
- Analysis of existing documentation, including requirements specications from earlier projects, regulations, industry standards, corporate policy documents, contracts, and business plans.
- Analysis of data, such as the various states that a data object can have and the conditions under which a user or a system event can change the object's state. These authorizations could also be represented as a roles and permissions matrix like the one shown earlier in Figure 5.2 to provide information about rules regarding user privilege levels and security.
- Compliance departments in companies building systems subject to regulation.

Just because you found some business rules in these various sources doesn't mean they necessarily apply to your current project or that they are even still valid. Computational formulas implemented in the code of legacy applications could be obsolete. Be sure to conrm whether rules gleaned from older documents and applications need to be updated. Assess the scope of applicability of rules you discover. Are they local to the project, or do they span a business domain or the entire enterprise?

Often, project stakeholders already know about business rules that will inuence the application.
Certain employees sometimes dealwith particular types or classes of rules. If that's the case in your environment, nd out who those people are and bring them into the discussion. The BA can glean business rules during elicitation activities that also dene other requirements artifacts and models.
During interviews and workshops, the BA can ask questions to probe around the rationale for the requirements and constraints that users present. These discussions frequently surface business rules as the underlying rationale. Figure 5.3 shows severalpotential origins of rules. It also suggests some questions a BA can ask when discussing various requirements issues with users.

Figure 5.3: Discovering business rules by asking questions from dierent perspectives.



5.4 Business rules and requirements

After identifying and documenting business rules, determine which ones must be implemented in the software. Business rules and their corresponding functional requirements sometimes look a lot alike. However, the rules are external statements of policy that must be enforced in software, thereby driving system functionality. Every BA must decide which rules pertain to his application, which ones must be enforced in the software, and how to enforce them.
Recall the constraint rule from the Chemical Tracking System requiring that training records be current before a user can request a hazardous chemical. The analyst would derive dierent functional requirements to comply with this rule depending on whether the training records database is accessible to the Chemical Tracking System. If it is, the system can look up the user's training record and decide whether to accept or reject the request. If the records aren't available online, though, the system might store the chemical request temporarily and send a message to the training coordinator, who could approve or reject the request. The rule is the same in either situation, but the software functionality—the actions to take when the business rule is encountered during execution—varies depending on the system's environment.

As another illustration, consider the following rules:

- Rule #1 (action enabler): If the expiration date for a chemical container has been reached, then notify the person who currently possesses that container.
- Rule #2 (fact): A container of a chemical that can form explosive decomposition products expires one year after its manufacture date.

Rule #1 serves as the origin for a system feature called Notify chemical owner of expiration." Additional rules like #2 would help the system determine which containers will have expiration dates and thus require notifying their owners at the right time. For instance, an opened can of ether becomes unsafe because it can form explosive byproducts in the presence of oxygen.Based on such rules, it's clear that the Chemical Tracking System must monitor the status of chemical containers that have expiration dates and inform the right people to return the containers for safe disposal.The BA might derive some functional requirements for that feature such as the following:

Expired.Notify.Before If the status of a chemical container that has an expiration date is not Disposed, the system shall notify the container's current owner one week before the date the container expires.

Expired.Notify.Date If the status of a chemical container that has an expiration date is not Disposed, the system shall notify the container's current owner on the date the container expires.

Expired.Notify.After If the status of a chemical container that has an expiration date is not Disposed, the system shall notify the container's current owner one week after the date the container expires.

Expired.Notify.Manager If the status of a chemical container that has an expiration date is not Disposed, the system shall notify the manager of the container's current owner two weeks after the date the container expires.

Whenever you encounter a set of very similar requirements like these, consider laying them out in the form of a table instead of a list [11].This is more compact and easier to review, understand, and modify. It also provides a more concise way to label the requirements, because the table has to show just the suxes to append to the parent requirement's label. Here's an alternative representation for the preceding four functional requirements:

Expired.Notify If the status of a chemical container that has an expiration date is not Disposed, the system shall notify the individuals shown in the following table at the times indicated.

Requirement ID	Who to notify	When to notify
.Before	Container's current owner	One week before expiration date
.Date	Container's current owner	On expiration date
.After	Container's current owner	One week after expiration date
.Manager	Manager of container's current owner	Two weeks after expiration date

5.5 Tying everything together

To prevent redundancy, don't duplicate rules from your business rules catalog in the requirements documentation. Instead, refer back to specic rules as being the source of certain functionality or algorithms. You can dene the links between a functional requirement and its parent business rules in several ways; following are three possibilities.

- If you are using a requirements management tool, create a requirement attribute called Origin and indicate the rules as being the origin of derived functional requirements.
- Dene traceability links between functional requirements and the connected business rules in a requirements traceability matrix or a requirements mapping matrix [16].
  This is easiest when the business rules are stored in the same repository as the requirements.
- If the business rules and requirements are stored in word processing or spreadsheet les, dene hyperlinks from business rule ID references in the requirements back to the descriptions of the business rules stored elsewhere. Be aware that hyperlinks are prone to breaking if the location of the rules collection changes.

These links keep the requirements current with rule changes because the requirements simply point to the master instance of the rule. If the rule changes, you can search for the linked rule ID to nd requirementsor implemented functionalityyou might need to change. Using links like this facilitates reusing the same rule in multiple places and projects, because the rules are not buried in the documentation for any single application. However, a developer reading the SRS will need to follow the cross-referenced link to access the rule details. This is the trade-o that results when you elect not to duplicate information [11].

As with so many aspects of requirements engineering, there is no simple, perfect solution to managing business rules that works in all situations. But after you begin actively looking for, recording, and applying business rules, the rationale behind your application development choices will become clearer to all stakeholders.