USE CASES

REQUIREMENTS IN CONTEXT

Daryl Kulak

and

Eamonn Guiney

SECOND EDITION

Text Illustrations by Erin Lavkulich

Use Cases

Requirements
in Context

Second Edition

When use cases drive the lifecycle, it also helps to assure the stakeholders that all the requirements are being addressed as development progresses. For example, at any point in the lifecycle, anyone who is creating artifacts—whether they are design deliverables or code—should be able to answer this question: Which use case (or scenario) are you elaborating?

3.8.4 Use Cases Discourage Premature Design

Use cases discourage (although nothing can prevent) premature design. When design creeps into a use case, it becomes quite obvious. When the system is performing several steps before responding to a user, it sounds an alarm that perhaps internal system design is being created and exposed.

3.9 The Role of the Business Rules Catalog

Use cases are an important part of the requirements picture, but by themselves they are not enough. Use cases capture the interactions between users and the system, but they cannot portray all the subtleties of how a business is run. For this, we need *business rules*.

A list of business rules is not the same thing as a list of requirements. Business rules are the written and unwritten rules that dictate how a company or agency conducts its business. Requirements relate to a specific application being considered or developed.

In our methodology, we use a combination of use cases and business rules. Use cases cover a great many of the requirements by specifying interactions between actors and an application. However, these interactions are governed by the business rules that set the tone and environment in which the application operates. Some businesses create lists of business rules for their businesses even though they may not be building any new computer applications. It is simply important to them to document how their business runs.

According to Ellen Gottesdiener (2002) author and consultant, there are five categories of business rules:

- *Structural facts*—Facts or conditions that must be true. Example: The first customer contact is always with a salesperson.
- *Action restricting*—Prohibiting one or more actions based on a condition. Example: Do not accept a check from a customer who does not have an acceptable credit history.
- *Action triggering*—Instigating an action when one or more conditions become true. Example: Send the shipment as soon as the pick items (items selected from inventory) have been collected.
- *Inferences*—Drawing a conclusion when one or more conditions become true. Example: Members who fly more than 100,000 miles in one calendar year become Elite members.
- *Calculations*—Calculating one value given a set of other values. Example: The sales amount is the total retail value of the line items but does not include state or federal tax.

Business rules must be *atomic*: Each business rule should be stated at the finest level of granularity possible. Methods of atomizing tend to be more art than science.

The best way to get a feel for business rules is to see lots of examples (see Table 3.3). We've provided some examples from our experiences and our imagination and have categorized them in several ways:

- Type of rule
- Likelihood of change (whether the rule is static or dynamic)
- Source

Your business rules catalog may look significantly different, depending on the type of data you record and cross-reference.

NOTE: It is easy to slip into the trap of creating a list of business rules that is actually a list of requirements. Remember that business rules always relate to how a company operates; they do not relate to the requirements of a specific application. A requirement is usually phrased, "The system shall," whereas a business rule states, "The business works like this."

Table 3.3 We-Rent-All Equipment Rentals Co. Business Rules Catalog Sample

No.	Rule Definition	Type of Rule	Static/ Dynamic	Source
001	Cash, personal check, and credit card are accepted for rental payment.	Action restricting	Dynamic	Management policy
002	Customers who rent driving equipment must possess a U.S. driver's license.	Action restricting	Static	Management policy
003	Customers who rent driving equipment must provide proof of insurance for the rental.	Action restricting	Static	Management policy
004	Rentals must be returned by the next day at the same time as the conclusion of the rental transaction, unless otherwise specified at the time of rental or in a requested extension.	Action restricting	Static	Management policy

continues

Table 3.3 *continued*

No.	Rule Definition	Type of Rule	Static/ Dynamic	Source
005	If a rental item is not available, substitute using the substitution part chart.	Action triggering	Static	Management policy
006	Customers can make reservations up to six months in advance.	Action restricting	Dynamic; advance limit may change	Management policy
007	Rental items will be serviced according to the part service chart.	Action triggering	Static	Management policy
008	If a rental item that was reserved is not available to the customer on the day of the rental for any reason, the customer service representative will arrange for the customer to rent it from a competitor at our cost.	Action triggering	Dynamic; customer service level may change	Management policy
009	If customers are renting firearms, a background check must be initiated and a 30-day waiting period is required.	Action triggering	Dynamic; law may change	Federal law
010	A customer may return an item only if it has not yet been taken out of the store.	Structural facts	Static	Management policy

3.10 Managing Success

Adopting a use-case-driven approach can help you organize and systematize your software development efforts. This chapter describes a series of steps you can take to produce a set of documents that will help you carefully define your system's requirements. The next three chapters discuss the three iterations that we recommend, explaining how the use cases we've introduced are used to drive and guide the process overall.

What is to give light must

endure burning.

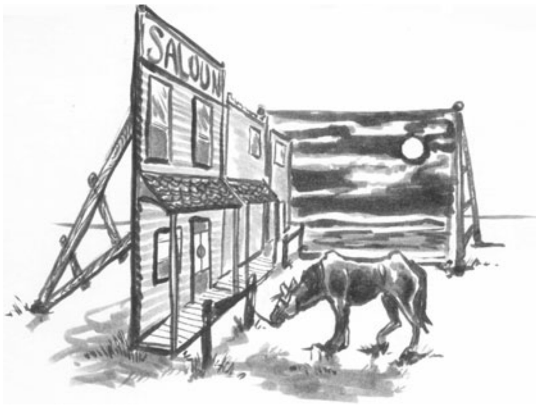
—Viktor Frankl

4 The Facade Iteration

4.1 Objectives

The first iteration in the requirements lifecycle is the Facade iteration. Its purpose is to create placeholders for each major interaction you expect the actors to have with the application.

A Facade use case contains only the minimum information needed as a placeholder, including a name and short description of the interaction. It also identifies the initiating actor and other actors. Executing this iteration is difficult because you may not yet have a



The Facade Iteration

concept of the application. For this reason, the team will do its best work if the environment encourages openness and creativity.

As you develop a definition of the proposed system, you can call on the following sources for ideas and opinions:

- Users
- Project team
- Industry experts
- IT management
- User management
- Owners of the data

4.1.1 Users

The users have the major influence on the definition of proposed system interactions. They are the focus of the new system, and their input and buy-in are critical.

However, as anyone knows who has been through requirements definition even once, the users cannot tell you the whole story. They are not equipped to define fully what the new system should do. Why? First, the new system is probably not being created merely to automate an existing system. Many new business processes, perhaps not yet built, will dictate the system interactions. In addition, the users often know their domain so well that they assume that most of what they do is terribly obvious. Alternatively, they've been put into their current role recently as a result of organizational restructuring, and they haven't had time to become familiar with their environment. One department in an oil and gas company for which we worked had a 110 percent annual staff attrition rate, so relying solely on the users was not possible for that development effort. In any case, the requirements-gathering team has its work cut out for it.

There's another issue. If you don't have a *subject matter expert* (SME) on your project team who knows this domain as well as a user, your team will not have the opportunity to "read between the lines" of what the users tell you. You need to make these kinds of inferences to identify better, faster, cheaper processes and to recognize the vital pieces of the puzzle that the users have left out—omissions that will come back to bite you in three months' time. SMEs are often ex-users themselves, or they may be IT people who have specialized in an industry and have a number of system implementations under their belts.

4.1.2 Project Team

The project team, too, has input into how the user-system interactions should occur. After all, the team is responsible for the work! The project team should have a focus on setting standards for interaction, maintaining scope, making inferences from user input, and documenting, storing, and indexing the requirements. The major guiding force behind these interactions should be the users and the SMEs on your project team. The task of the rest of the team is to transmute these models into use cases. The users provide the information on how they do things now and what they would like to see changed, and the SMEs help to shape the system into more refined, elegant, and profitable processes.

4.1.3 Industry Experts

The challenge of the project team members is to take *user-centric* information offered by computer industry experts and luminaries and to determine whether it applies to this situation. Industry experts can provide only rules of thumb, and rules are meant to be broken. Determine your position on the industry "advice-of-the-day," and if you decide to deviate from it be prepared to defend your position. Whatever you do, don't simply follow the experts blindly. Unless you believe in your direction, your project won't work.

4.1.4 IT Management Group

The IT management group always has opinions on how the interactions of the system should be described. If you are a member of the in-house IT group, you must balance these opinions with the user needs and wants. If you are an outside consultant the same balancing act is required, but sometimes it is easier being a third party. The IT group can provide valuable input regarding systems that have been developed previously, interfaces between user departments, previous incarnations of this functionality, and other contextual information.

As with user input, you should be cautious in weighing IT viewpoints. IT staff (yes, this means us!) have a strong tendency to push requirements gathering into something that is technology-centered. Technology-centric solutions have no place in this early activity of system development. It is dangerous to commit an application to a specific technology or

Requirements analysts need to wear blinders to avoid thinking about the various technology options.