An Introduction to Use Cases

Geri Schneider and Jason P. Winters
Wyyzzk, Inc.
Santa Clara, CA 95051

1 Abstract

Use cases, also referred to as user stories, enable the functional requirements of a software system or business process to be written from the software or processes' point of view. This paper is a brief introduction to use cases. The basics of writing a good use case are described.

2 Introduction

For many decades, the software industry has depended on a functional requirements specification document to define the software to be developed. Requirements in the document usually take a form such as "The system shall do something." Some examples might be: "The system shall display all numbers with 2 decimal places." "The system shall calculate all numbers to 4 decimal places of accuracy." These requirements are explicit and easy to test.

How the system works is difficult to determine. If I am sitting in front of a computer, what is my primary objective, secondary goal, etc? Use cases were developed to write functional requirements in a way that emphasizes how the system is to be used.

Ivar Jacobson began the preliminary work that led to the development of use cases. In 1967, he developed a new set of modeling concepts for the development of large telecommunications switching systems. Mr. Jacobson continued working on his methodology during the 1970s and 1980s. Objectory Process, his methodology of modeling concepts, was released. Use cases were an integral part of that methodology. Use cases were included in the Unified Modeling Language standard, developed during 1995-97.

Use cases have continued to increase in popularity. They are used to write requirements in a wide variety of applications. Compatible with any software development process, they are prominent in Unified Process, Objectory Process, and eXtreme Programming, where they are called user stories.

The remainder of this paper focuses on the writing of a good, basic use case. Only the text form of the use case is considered. Use case diagramming will not be shown. The text form can be done with a number of diagrams. The interested reader may refer to the list of resource books at the end of the paper for more information about use cases and diagramming use cases.

3 Definitions

Use cases describe the functional requirements of the system from the point of view of system users.

A use case is a complete sequence of steps that provides an actor with a result of value.

An actor is either a human or non-human system user.

A system is the object under development. This could be software or a business process.

4 Determining your Audience

First, consider the objective of the use case before beginning the writing stage. Determine who needs the use case, how it will be used, and what will be done with the case. This will help to write a use case with the appropriate level of detail.

Different audiences have different needs for use cases. Consider if the use cases are being written for managers, users, or developers. Placing each differing viewpoint in one document is not the most feasible approach. Doing so creates a very large document; smaller documents are easier to work with. Secondly, numerous differing viewpoints will be confusing to some readers and users.

Use cases are used for a variety of purposes after being written. Consider various purposes when determining the level of detail in the use case. Is the use case being written to describe the basic requirements of the system as part of a contract with a customer? Will the use case be used to create white-box test plans, black-box test plans, or both? Will user manuals for software be created from use cases? Will the use case be used to document new corporate processes? Will software be developed from use cases? If use cases will be serving these purposes, more than one version of each use case is necessary.

The audience should be kept in mind as the use case is written. If you are uncertain whether particular information should be included in the use case, consider who will be reading the use case. Then decide if those users need that information. If they do, include the information. If they do not, exclude certain information. However, all information, whether included or excluded, should be saved in a separate document or diagram.

5 The Basic Structure of a Use Case

Each use case must include details outlining what must be done to accomplish functionality. Basic functionality, alternatives, error conditions, anything that must be true before starting the use case or exiting the use case must be considered.

A complete use case description may become quite complicated. This is not something written at the very beginning, but a description that evolves over time. The use case is documented in several sections for easier reading. Sections can be written one at a time until the use case is complete. This paper illustrates the most basic form of the use case. Flow of events, preconditions, and postconditions are included. Let's look at the parts that make up a use case in a little more detail.

5.1 Flow of Events

The primary part of the use case is the flow of events. The flow of events is divided into two sections: the basic path and the alternative path. We will start writing the basic path by choosing the most common sequence of steps for the use case. After writing the basic path, alternatives and exceptions to the use case can be added. These are the alternative paths of the use case.

5.2 The Basic Path

The basic path is written on the assumption that everything goes right. Neither bugs, nor errors exist; it is a perfect world often called the happy day scenario. One basic path is required for each use case.

The basic path is a series of simple declarative statements listing the steps of a use case from the actor's point of view. A statement such as "The use case begins when...," designates the beginning. Similarly, a phrase such as "The use case ends," signifies the end. At each step, assume everything is correct. Pick the most common way of doing each step.

Exhibit 5-1 illustrates an example of the basic path for a use case in which a customer is placing an order for products. Your objective may be to provide software to be used by your customers to place an order, and you want to include that information in the use case. In this scenario, the use case would resemble Exhibit 5-2.

Exhibit 5-1 Basic Path Example
Place Order Use Case
Flow of Events
Basic Path
1. The use case begins when the customer contacts the company to place an order.
2. The customer supplies his or her name and address.
3. The customer supplies product codes for the products he or she wishes to order.
4. The company gives the customer a total amount due.
5. The customer supplies credit card payment information.
6. The company gives the customer an order identifier and the use case ends.

Exhibit 5-2 Basic Path Software Example
Place Order Use Case
Flow of Events
Basic Path
1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. The customer enters a product code.
4. The system supplies a description and price.
5. The system adds the item price to the total.
6.The customer enters credit card payment information.
7. The customer selects Submit.
8. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system.
9. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends.

The first use case allows the customer to order multiple products. The second use case involves the customer entering one product code. We need to indicate repetition in this use case because the user can enter more than one product on a single order.

Use repetition to repeat a step or a set of steps multiple times. Indicate clearly where the repetition starts and ends. Also clearly indicate how it will be ended. It may end at the end of a set, or a condition may cause the repetition to stop.

Repetition is shown with either a for statement or a while statement. Either one will work; choose the statement that is the easiest to read. Exhibit 5-3

shows repetition with the for statement. Exhibit 5-4 shows repetition with the while statement.

Exhibit 5-3 Repetition Example with "for"
Place Order Use Case
Flow of Events
Basic Path
1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. The customer enters product codes for products to be ordered.
4. *For each product code entered*
   a) the system supplies a description and price
   b) the system adds the item price to the total.
end loop
5. The customer enters credit card payment information.
6. The customer selects Submit.
7. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system.
8. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends.

Exhibit 5-4 Repetition Example with "while"
Place Order Use Case
Flow of Events
Basic Path
1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. While the customer enters product codes
   a) the system supplies a description and price
   b) the system adds the item price to the total.
end loop
4. The customer enters credit card payment information.
5. The customer selects Submit.
6. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system
7. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends.

5.3 Alternative Paths

The basic path only handles the case in which everything is correct. Alternatives and error conditions must still be indicated within the use case. Alternative paths are used for this purpose.

An alternative path is one that allows a different sequence of events than that used for the basic path. Alternative paths are used to show different choices a user can make, error conditions, and things that can happen at any time.

When a user is given a choice of one of several options, the basic path is selected as the most likely choice. The remaining choices are documented as alternative paths.

Since the basic path assumes everything is correct, alternative paths are used to document error conditions. These alternative paths answer questions such as: What could go wrong and what will we do about it? What if a transaction is cancelled midway? What is done in that situation?

Alternative paths are particularly useful for illustrating potential occurrences at any time, such as, a cancelled transaction, or accessing context-specific help. For example, during the Place Order use case, a customer may be allowed to cancel the order at any time prior to submission.

One method for finding alternative paths is by thumbing through the basic path line by line while asking questions:

- Can another action be taken at this point?
- Could something go wrong at this point?
- Could specific behavior present itself at any time?

Categories are another method used to discover alternatives. Ask if your particular use case needs alternative paths of these types:

- An actor exits the application
- An actor cancels a particular operation
- An actor requests help
- An actor provides bad data
- An actor provides incomplete data
- An actor chooses an alternative way of performing the use case
- The system crashes
- The system is unavailable

Once alternative paths for the use case have been found, put the list of alternatives in the alternative paths section, located in a separate section of the document. This section of the document follows the basic path (see Exhibit 5-5). To begin this section, simply list each alternative and exception thought of.

Complex or important alternative paths also require a sequence of steps detailing their behavior. These can be written in the same way the basic path was written. Select a readable style, check for completeness and correctness, and apply a writing style consistent with the primary scenarios.

Alternative paths can be written in a paragraph format. Refer to the third alternative in which the payment is not confirmed in example Exhibit 5-5.

Detailed alternatives may also be written using a numbered list. (see Exhibit 5-6).

Simpler use cases can be documented with alternatives in the basic path. More complex use cases are easier to read if alternatives are written separately. Approaches are frequently combined. These include the placement of simple alternatives in the basic path and more complex alternatives in the alternative paths section. Step 3 of Exhibit 5-7 shows an alternative in the basic path.

Exhibit 5-5 Place Order Use Case with an Alternative Paths Section

Place Order Use Case
Flow of Events
Basic Path
1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. The customer enters product codes for products to be ordered.
4. For each product code entered
a) the system supplies a description and price
b) the system adds the item price to the total.
end loop
5. The customer enters credit card payment information.
6. The customer selects Submit.
7. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system.
8. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends.

Alternative Paths
• If any information in step 7 is incorrect, the system will prompt the customer to correct the information.
• The customer can cancel the order at any time before selecting Submit and the use case ends.
• If payment is not confirmed in step 8, the system prompts the customer to either correct payment information or cancel. If the customer chooses to correct the information, return to step 5 in the Basic Path. If the customer chooses to cancel, the use case ends.
• Customer unable to login due to bad password or username
• Product code does not match actual products
• Product is no longer available
• Customer pays by check
• Customer sends order by mail
• Customer phones in order
• The system crashes midway through placing the order
• Customer unable to login due to system not responding
• Order gets lost

Exhibit 5-6 Place Order Use Case Detailed Alternative Paths Section

Alternative Paths
Alternative 1: Incorrect data
1. This alternative begins in step 7 of the basic path once the system detects incorrect information.
2. The system prompts the customer to correct the information.
3. The basic path continues with step 7.
Alternative 2: Cancel
1. At any time during the Place Order use case, the customer may select cancel.
2. The system prompts the customer to verify the cancel.
3. The customer selects OK and the use case ends.

Exhibit 5-7 Place Order Use Case with an Alternative Paths Section

Place Order Use Case
Flow of Events
Basic Path
1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. If the customer enters only a zip code, the system supplies the city and state.
4. The customer enters product code(s) for product(s) to be ordered.
5. For each product code entered
a) the system supplies a description and price
b) the system adds the item price to the total.
end loop
6. The customer enters credit card payment information.
7. The customer selects Submit.
8. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system.
9. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends.

Alternative Paths
Alternative 1: Incorrect data
1. This alternative begins in step 7 of the basic path once the system detects incorrect information.
2. The system prompts the customer to correct the information.
3. The basic path continues with step 7.
Alternative 2: Cancel
1. The customer may select cancel at any time during the Place Order use case.

2. The system prompts the customer to verify the cancel.
3. The customer selects OK and the use case ends.

How detailed should the alternatives be? A complete sequence of steps could be written for each alternative path; however, this is unnecessarily time-consuming. In many cases, alternative paths will vary from the basic path and from one another incrementally. Instead of writing an entire sequence of steps, note the variation in the alternative brief description. Writing a complete set of detailed descriptions requires time that could be used toward building a system. There is no point in building your whole system in a natural language, such as English. There are no automatic English-to-Java translators!

5.4 Pre- and Postconditions

Now the use case flow of events has been written, two sections remain to be completed. These sections are the precondition and the postcondition.

Pre- and postconditions indicate what comes before and after the use case. They tell what state the system must be in at the start of the use case (precondition), or what state the system must be in at the end of the use case (postcondition). The postcondition must be true regardless of

which branch or alternative is followed for the use case. Exhibit 5-8 provides an example of a precondition and a postcondition for the Place Order use case. Notice that the postcondition is not a simple expression. Since the postcondition must be true regardless of what happens, compound conditions are frequently used for the use case postcondition.

Exhibit 5-8 Pre- and Post Conditions
Place Order Use Case
Precondition: A valid user has logged into the system.

Flow of Events
Basic Path
1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. If the customer enters only the zip code, the system supplies the city and state.
4. The customer enters product codes for the desired products.
5. For each product code entered
a) the system supplies a description and price
b) the system adds the item price to the total.
end loop
6. The customer enters credit card payment information.
7. The customer selects Submit.
8. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system. If any information is incorrect, the system prompts the customer to correct it.
9. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends. If payment is not confirmed, the system will prompt the customer to correct payment information or cancel. If the customer chooses to correct the information, go back to step 6 in the Basic Path. If the customer chooses to cancel, the use case ends.

Alternative Paths
Alternative 1: Cancel
1. At any time in the Place Order use case, the customer may select cancel.
2. The system prompts the customer to verify the cancel.
3. The customer selects OK and the use case ends.

Postcondition: If the order was not cancelled, it is saved in the system and marked confirmed.

5.5 Who Initiates the Use Case

The use case initiator is usually - an actor or the system. The Place Order use case is clearly started by the customer actor. If we created a use case Get Status on Order, it is less clear where it should start. We could either have the customer always request status, have the system send a message to the customer when status changes, or both. Each of these three are correct. It is important to be explicit as to what is allowed in the use case.

6 Level of Detail

Many ask how much detail should be included in the use case. This depends on the audience. We have used up to three versions of a use case at different levels of detail.

One level of use case that is quite useful is the Business Process use case. Business process use cases describe the processes a company uses to satisfy the requests of the customers. A business process use case can include the use of manual processes, physical entities such as paper forms, and software. The business process use case may also indicate those inside the company who perform the business processes.

A business process use case describes a complete process from the point of view of a customer of the

company. It frequently looks like a sequence of lower level use cases. It starts with a request from a customer and ends with the fulfillment of the request. Exhibit 6-1 is an example of this kind of use case. The focus is on the order the things are done and what department is responsible. Do not worry how each department does its job. From the customer's point of view, this describes the entire use case from the time an order is placed until the product arrives.

Exhibit 6-1 Order Products Use Case - Business View

Flow of Events

Basic Path

1. The use case begins when the customer places an order for products with the customer service department.
2. The customer service department sends the payment information for the order to the accounting department.
3. The accounting department updates National Widgets accounts and deposits the payments in the bank.
4. The customer service department sends the order to the warehouse department.
5. The warehouse department collects the items for the order and sends them with the shipping address to the shipping department.
6. The shipping department packages items with the shipping address and sends the package through a shipping company for delivery to the customer. The use case ends.

This use case describes the complete process to the customer. Interactions between different parts of the company are also described. It is also good at describing how the different parts of the company interact. Let's now assume part of this use case will be automated. More detail about software use is required. Instead of putting the detail in this use case, making it large and complex, select the steps required for automation and place them into new software use cases.

Software use cases describe how a user will interact with specific software. A software level use case only describes the use of software. Most of the examples in this paper are software use cases that describe the software from the point of view of the actor who is a system user. This kind of use case is the one most familiar to people, and is the most common kind of use case written for a project. Business process use cases show how all software use cases work together to accomplish a larger task.

Exhibit 6-2 is an example of a software use case. It is the same place order use case we have been working with. Comparing it to the business process use case in Exhibit 6-1, we see that it is steps 1 and 2 of exhibit 6-1 with more detail included. It is labeled as a user view because it only includes information known to the user.

Exhibit 6-2 Place Order Use Case - User View

Flow of Events

Basic Path

1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. If the customer enters only the zip code, the system will supply the city and state.
4. The customer enters product codes for products to be ordered.
5. *For each product code entered*
   a) the system supplies a description and price
   b) the system adds the price of the item to the total.

end loop

6. The customer enters credit card payment information.
7. The customer selects Submit.
8. The system verifies the information, saves the order as pending, and forwards payment information to the accounting system.
9. When payment is confirmed, the order is marked confirmed, an order ID is returned to the customer, and the use case ends.

Another level of detail of a software use case is for a developer, who requires more information to develop the system. The developer must write code system requirements. The user view of the use case leaves many questions unanswered. For example, where will the system obtain city, and state information when a zip code is supplied in Step 3 of the Place Order Use Case? Will this information appear in a table created by the developer, does it exist in a company database, must software containing this information be purchased, or will this information be provided by the U.S. Post Office? It is presumed in Step 5a, that the inventory system will supply this information, but that should be explicitly stated. What does verify the information mean in Step 8? What is supposed to happen there? What does payment is confirmed mean in Step 9? What is supposed to happen there? Lastly, when and how are the tax and shipping information calculated?" Exhibit 6-3 is another version of Place Order that answers these questions.

Much of the information contained here is of no relevance to the customer. However, this use case need not be shown to a customer. This is used by the developers to write code. Most of the time we do not write developer level use cases. Instead, this

information is documented in diagrams, such as sequence diagrams.

Exhibit 6-3 Place Order Use Case - Developer View

Flow of Events

Basic Path

1. The use case begins when the customer selects Place Order.
2. The customer enters his or her name and address.
3. If the customer enters only the zip code, the system will use the zip code to query the U.S. Post Office online repository to get the city and state. The system will add the city and state to the order.
4. The customer enters product codes for products to be ordered.
5. *For each product code entered*

a) The system uses the product code to query the inventory system software for a product description and price. The system adds the description and price to the order. The system queries the customer for the quantity of the product. The customer enters a quantity for the product.

b) The system adds the price of the item to the subtotal of the order.

end loop

6. The customer enters credit card payment information.
7. The customer selects Submit.
8. The system ensures all necessary data is entered, which must include a complete shipping address, credit card payment information, and at least one product. The system saves the order as pending, and forwards the payment information and subtotal to the accounting system.
9. The accounting system calculates the tax and shipping amounts, and returns a total for the order along with an indication of success in accepting the payment. The system marks the order confirmed, returns the total and an order ID to the customer, and the use case ends.

Whether or not a use case is complete depends on the developer's point of view. The Order Products use case at the Business Process Level answered questions regarding how parts of the company work together, and in which order objectives must be completed. From the point of view of managing the process of ordering products, the Order Products use case is complete. From the point of view of a customer, the Order Products use case is missing information with regards to placing an order. But a correct level of detail is used in the Place Order use case, describing the actor view of the software. From the point of view of a developer, a version of Place Order describing the developer view of the software is preferential.

7 A Quick Review of the Use Case

This is a good point in time to review the written use cases. This section includes some simple things to consider when first reviewing use cases.

Following is a very quick review used each time while reviewing use cases. It is both simple and catches a lot of errors.

1. How does the use case begin and who initiates it? If the use case is for software, how does the software know when the use case begins? If the use case describes a business process, when does that process start?
2. How does the use case end? What is the final thing that happens?
3. If the use case produces data, does that data need to be stored? Where?
4. If the use case uses data, where did the data come from?
5. Have all actors in the use case text been accounted for?

Here are some other things to look for. Each step of the use case should be a simple, declarative statement. By default, the steps will be in order by time. What if the steps can occur in any order? If this is the case, make it clear in the description. This could be a simple statement at the beginning of the use case that the steps can run concurrently. Or you might state that some of the steps can happen in any order.

Resist the temptation to become too detailed. More detail can be added over time. At this point in the process, we are collecting requirements, not doing detailed analysis or design. On the other hand, the use case needs to be complete. Be very clear on the start and end points, and make sure the list of steps generally cover everything needed to accomplish the functionality of the use case.

You will find a large percentage of use cases start and end with an actor. From our order-processing system, we see that Place Order starts and ends with the customer. A smaller number of use cases start with an actor and end internally or start inside the system and end with an actor. We have found this convenient when dealing with time. For example, if our order-processing system is automated to check and place back-ordered items from a supplier once each week, this would start internally and end with the supplier actor.

By definition, use cases are written from the actor's point of view. Therefore, each step in the use

case should be visible to or easily surmised by the actor.

Use cases are a communication tool. They are effective only to communicate information about how the system works to the reader. It is important to consider who will be reading the use cases. Will it be end users, marketing specialists, developers, or management? Whoever it is, they have to be able to understand the use cases. If they don't, the use cases need to be rewritten.

Another correctness check is to look at each step of the basic path, one by one. For each step, ask yourself, "What is the most likely thing to occur here?" That is what should be written for that particular step.

Don't worry about getting the use cases perfect. The nature of the process is to be iterative; keep looking back over completed work and refine it to reflect knowledge learned. The use cases will improve as your understanding of the system improves.

On the other hand, enough information must be included in each use case to be able to determine whether a particular use case handles a particular functionality.

8 A Use Case Template

Only the most basic parts of a use case have been covered in this paper. There are other kinds of information that may be included in a use case. Below is a sample detail template for a use case. Every section does not need to be included. If your use cases are not this complex, not all sections will be used. Additional sections may be helpful. That is fine. This is a sample given as a starting point. If it works as is, use it; otherwise, modify as needed.

Use Case Name

Brief Description

<Usually a paragraph or less. May include the priority and status of this use case.>

Context Diagram

<A small use case diagram showing this use case and all of its relationships.>

Preconditions

<A list of conditions that must be true before the use case starts.>

Flow of Events

<A section for the basic path and each alternative path.>

Postconditions

<A list of conditions that must be true when the use case ends, regardless of which scenario are executed.>

Subordinate Use Cases Diagram

<A small use case diagram showing the subordinate use cases of this use case.>

Subordinate Use Cases

<A section for each subordinate use case with its flow of events.>

Activity Diagram

<An activity diagram of the flow of events, or some significant or complex part of the flow of events.>

View of Participating Classes

<A class diagram showing the classes that collaborate to implement this use case.>

Sequence Diagrams

<One or more sequence diagrams for the basic path and alternatives.>

User Interface

<Sketches or screen shots showing the user interface. Possibly storyboards.>

Business Rules

<A list of the business rules implemented by this use case.>

Special Requirements

<A list of the special requirements that pertain to this one use case. For example - timing, sizing, or usability.>

Other Artifacts

<This can include references to the subsystem the use case belongs to, an analysis model, a design model, code, or test plans.>

Outstanding Issues

<A list of questions pertaining to this use case that needs to be answered.>

9 Other Documents Required

Other documents are frequently developed along with the use cases. These include non-functional requirements, a glossary of terms, a data definition document describing the format and validation rules for data elements, and guidelines for the user interface. You may also want to maintain a document that lists outstanding issues or questions that need to be addressed.

Sections of the use case document are frequently moved into other documents. For example, you may want all the user interface screen shots together in a document separate from the use cases. Or you may want to collect all the special requirements together in one document, rather than scattering them throughout the use case documents. It is good to have a template for these other documents as well, so everyone knows what kind of information to include in the document.

Requirements such as timing, sizing, performance, and security are frequently called non-functional requirements to distinguish them from the

use cases that are the functional requirements. It is particularly important to have a template for a non-functional requirements document for the team to understand what is meant by non-functional requirements. The template will also help you find all the non-functional requirements in your system.

Exhibit 9-1 Non-functional Requirements Template

Usability

<What is known about the users of this system? Are they computers, power users, others, or a combination of these? How easy must this system be to use?>

System

<What kind of system will this software operate on? Is it necessary to port to multiple platforms? Must the software support multiple simultaneous users?>

Security

<What are the needs for secure login or secure transmission of data?>

Persistence

<Do we have any persistent data? Are there any requirements on the database to use?>

Integration with Other Systems

<Does this software have to integrate with other software or hardware?>

Error Detection/ Handling/ Reporting

<Are errors allowed? What is a reasonable error rate? What are our requirements for prevention, detection, handling, and reporting of errors?>

Redundancy

<Are there any needs for redundant data, subsystems, processes, or hardware?>

Performance

<Are there any restrictions on how slow or fast the system or any part of it will run?>

Size

<Are there any restrictions on the size of the system or any part of it?>

Internationalization

<Does the software have to support any character set worldwide or only some of them? What information must be translated?>

Another document we recommend is a glossary of terms. Don't assume every term is known, understood, or that each term is defined the same. Write a glossary accessible to all team members.

Exhibit 9-2 Sample Glossary of Terms

Accounting System

A software system that tracks customer accounts, processes accounts receivable and processes accounts payable.

Manager

Anyone who gets reports from the Order Processing system.

Security

The need to control whom has access to our software and databases.

Another very useful document is the data definition document. This is a place to record information about the format of the data. For example, assume that a name is defined to be 50 characters. If that information is placed in the use cases, and the name is changed to 75 characters, every use case will have to be reviewed to look for a name with that number of characters to be changed. Instead, put the size of a name in the data definition document. If someone reading the use case wants to know the size of a name, they can look up the information in the data definition document. If the size of the name changes, you only have to change the information in one place, the data definition document.

Exhibit 9-3 Sample Data Definitions

City

An alphabetic string of no more than thirty characters. Strings are allowed to include a period and an apostrophe.

Product Price

A currency type field with two decimal places.

Tax

A currency type field calculated to four decimal places, but rounded to two decimal places.

Finally, a user interface document may be needed. One kind of user interface document is guidelines and standards. This can include requirements such as every screen must include an exit button, or the standard font for applications is 12 point Arial. Also included may be a user interface design document including screen shots and navigation information.

10 Conclusion

This paper has provided a brief introduction to use cases. Some history and rationale for use cases have been considered, followed by a demonstration of how to write a very basic form of a use case. A very basic use case consists of preconditions, a flow of events including the basic path and any number of alternative paths, and postconditions. A template was then provided to show other parts of a use case and

templates for other documents that should be included along with use cases. The ability of use cases to be documented with diagrams was mentioned next. The interested reader is referred to the Reference section for books with more information about use cases.

10.1 Acknowledgment

This paper is based on Chapters 3, 4, 6, and 7 from *Applying Use Cases [1]*.

10.2 References

1. Schneider, Geri, and Jason P. Winters, *Applying Use Cases, Second Edition: A Practical Guide*, Addison-Wesley, Boston, MA, 2001. ISBN 0-201-70853-1.
2. Jacobson, Ivar, Magnus Christerson, Patrik Jonsson, Gunnar Övergaard, *Object-Oriented Software Engineering*, ACM Press, NY 1992. ISBN 0-201-54435-0.
3. Fowler, Martin with Kendall Scott, *UML Distilled Second Edition*, Addison-Wesley, Menlo Park, CA, 2000. ISBN 0-201-65783-X.
4. Booch, Grady, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Menlo Park, CA, 1999. ISBN 0-201-57168-4.
5. *OMG Unified Modeling Language Specification Version 2.0,*, Object Management Group, 2004.