Introduction to Ridgeline Mountain Outfitters (RMO)

Ridgeline Mountain Outfitters (RMO) is a large retail company that specializes in clothing and related accessories for all types of outdoor and sporting activities. By the 2010s, the Rocky Mountain and Western states had seen tremendous growth in recreation activities, and with the increased interest in outdoor sports, the market for winter and summer sports clothes had exploded. Skiing, snowboarding, mountain biking, water skiing, jet skiing, river running, jogging, hiking, ATV biking, camping, mountain climbing, rappelling—all had seen a tremendous increase in interest in these states. People needed appropriate sports clothes for these activities, so RMO expanded its line of sportswear to respond to this market. It also added a line of high-fashion active-wear and accessories to round out its offerings to the expanding market of active people.

The company's growth charted an interesting history of mail-order, brick-and-mortar, and online sales. RMO got its start by selling to local clothing stores in the Park City, Utah, area. In the late-1980s and early-1990s, it began selling directly to customers by using catalogs with mail-in and telephone orders. It opened its first store in 1994 and soon expanded to 10 retail outlets throughout the West. Last year, retail store revenue was $67 million, telephone and mail-order revenues were $10 million, and Web-based sales were $200 million. Most sales continue to be in the West, although the market in several areas of the eastern United States and Canada is growing.

RMO produces its own line of outdoor and sportswear clothing. However, in order to offer a complete range of outdoor clothing in its retail outlets, it also sells other brands of outdoor and sportswear clothing. In addition, other types of clothing and accessories, such as footwear, leather apparel, and specialty sportswear, are available in the retail stores and through the online store.

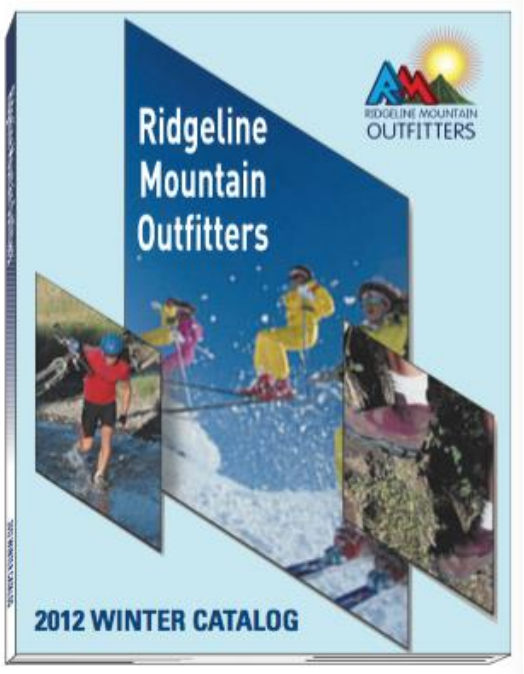

Figure 1-2 shows a sample of the catalogs that RMO mails out. Although mail-order and telephone sales are small, receiving the catalog encourages customers to go online to make purchases, so RMO continues to produce and mail abbreviated versions of its catalogs.

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

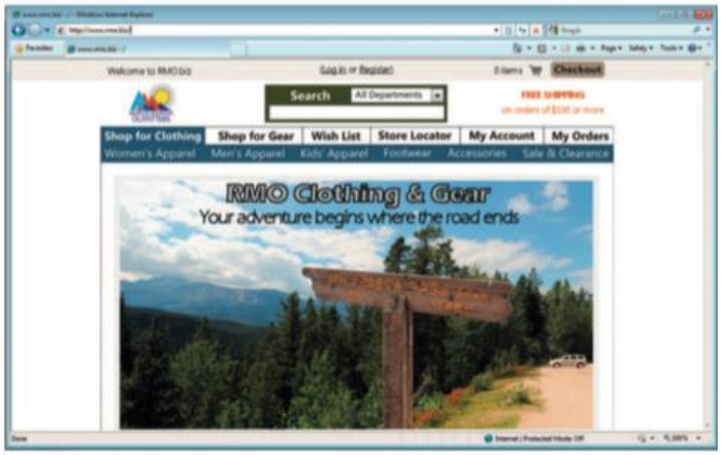

Figure 1-3 illustrates a typical order page from the online system.

Trade Shows

In order to keep its product line current and popular, RMO's purchasing agents attend apparel and fabric trade shows around the world. RMO purchasers have a good track record of predicting what products will be good sellers. In addition, RMO is always watching for new products and accessories that will enable it to expand its product line appropriately.

When purchasing agents attend a trade show, they frequently find various products that they want to add to the spring, summer, or winter apparel offering.

In the past, when RMO buyers wanted to place an order, they would exchange contact information with the seller at the trade show and upon returning to the home office would then follow up via e-mails and phone calls to formulate a contract and make a purchase order. RMO initiated an information system project to develop a system for collecting and tracking information about suppliers and about products added to its merchandise offerings. The Tradeshow System needs to take advantage of the latest in wireless devices and data capturing technologies to allow purchasing agents to research and complete purchase orders on the spot at the trade shows. RMO decided to use an agile, iterative project management approach to get the small system completed as fast as possible with maximum flexibility.

Developing RMO's Tradeshow System

We will organize our sample project—the RMO Tradeshow System—into several iterations. Our plan for the first iteration is to have the iteration last six days. Our primary objective is to introduce you to the concepts and techniques of the six core processes. Therefore, in some instances, we may go a little deeper into a core process than we might do on the first iteration of a real project. It is a little unrealistic to complete an entire iteration with all the necessary details in only six days, but it should be a good learning experience. There will not be a one-to-one correspondence with the six SDLC core processes and the six days of the project, but we will include all the SDLC core processes within the project.

Most new applications require a project with several iterations. In the first iteration, there are usually

three major objectives. The first objective is to get project approval. The second objective is to get a clear picture of the system's overall vision—all the major functions and data requirements. The third objective is to determine the detail specifications and develop a solution for one portion of the system (i.e., actually analyze, design, build, and test one part of the system). The second and third iterations would continue to work on the additional portions of the system based on the system vision

In our project, we will touch on all these objectives. We will show an example of a System Vision Document and then develop one portion of the overall system. It should be noted that the division of this project into days and daily activities is somewhat arbitrary. There are numerous ways to partition and organize the work. The following organization is quite workable, but it is not the only way to organize the project.

Pre-Project Activities

Before the project actually begins, the head of RMO's purchasing department works with a systems analyst to identify and document the specific business need as well as define a specific project objective. RMO's management reviews the primary project objective and provides budget approval. Every organization has to give budget approval before a project can start. Some organizations have a formal process to get a project approved; other organizations have a less formal process. Normally, there are two goals an organization must decide on to get a project off the ground:

- Identify the problem and document the objective of the solution system. (Core Process 1)
- Obtain approval to commence the project. (Core Process 1)

System Vision Document

As with all new projects within RMO, a System Vision Document is developed to identify the benefits to the company and the functional capabilities that will be included in the system. Frequently, this is done in two steps: developing a preliminary statement of benefits and then adding estimates of specific dollar costs and dollar benefits. Figure 1-5 is the System Vision Document for this project.

As described earlier, RMO needs a portable system that can be used by its purchasing agents as they attend various product and clothing fabric trade shows. The system needs to fulfill two major requirements. First, it has to have the functionality to capture information about suppliers and products. Second, it needs to be able to communicate with the home office systems, and because these trade shows are held in various venues around the world, various methods of connectivity are needed.

Preliminary investigation considered various equipment options, including laptop computers, iPad computing devices, and smartphones. Even though smartphones appeared to have the best connection options, the small size made viewing the details of photographs somewhat difficult; the iPad and other similar portable devices with advanced technology also appear to be viable options. However, due to the similarity of the smartphones and tablets, it seems feasible to develop an application that will execute on either device. Each purchasing agent could use his or her preferred device.

Toward the end of the pre-project activities, a meeting is held involving all the key persons, including a representative of executive management. The decision is made to move ahead with the project and budget the necessary funds.

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

Day 1 Activities

RMO—Supplier Information Subsystem

The project actually begins with Day 1, which is essentially a planning day. Usually, the first activity is the project team reviewing the System Vision Document and verifying that the preliminary work is still valid. It reviews the scope of the project to become familiar with the problem to be solved, then it plans the iterations and activities for the remainder of the project. The second SDLC core process—planning the project—includes business analysis and project management activities. All these topics will be treated in depth in later chapters. These activities are completed on Day 1:

- Determine the major components (functional areas) that are needed. (Core Process 2)
- Define the iterations and assigning each functional area to an iteration. (Core Process 2)
- Determine team members and responsibilities. (Core Process 2)

Planning the Overall Project and the Project Iterations

Myriad details need to be considered in a project plan. For our project, we will only focus on the bare essentials. We will describe project planning more elaborately in later chapters.

The project team meets with the users to review the overall business need and the objectives of the new system. The System Vision Document serves as the starting point for these discussions. As is often the case, the list of System Capabilities provides the foundation information for determining the overall project plan. The first step is to divide the system into several subsystems or components. A subsystem is simply a portion of the overall system. Based on the list of System Capabilities, the project team identifies these functional subsystems:

- Supplier Information subsystem
- Product Information subsystem

The Supplier Information subsystem will collect and maintain information about the manufacturers or wholesalers and the contract people that work for them. The Product Information subsystem will capture information about the various products, including detailed descriptions and photographs.

The next step is to identify which subsystems will be developed in which order. Many issues are considered, such as dependencies between the various tasks, sequential versus parallel development, project team availability, and project urgency. In our case, the team decides that the project will proceed in a serial fashion, with the Supplier Information subsystem scheduled as the first iteration.

Planning the First Iteration

Each iteration is like a systems development mini-project. The core processes described earlier can all be applied, with the scope limited to the component that is to be developed during the iteration. The planning process for an iteration consists of these three steps:

- Identify the tasks required for the iteration.

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

- Organize and sequence these tasks into a schedule.
- Identify required resources (especially people) and assign people to tasks.

The first step is to identify—or attempt to identify—all the individual tasks that need to be done. As these tasks are identified, they are compiled and organized. Sometimes, this organized list of tasks is called a Work Breakdown Structure. Figure 1-6 shows the Work Breakdown Structure for this iteration.

Part of the effort is trying to estimate how long each task will take. Because this project has a very limited scope (only six days), all the estimates will be in hours. These estimates do not include the time expended by those who are not on the team. However, of those on the team, the estimates include the time for the original work, the time for discussion, and the time for reviewing and checking the Work Breakdown Structure for accuracy and correctness.

The next step is to get these tasks organized into a schedule. Again, we can be very formal and use a sophisticated project-scheduling tool or we can just list the tasks in the order we think they need to be done. An important part of building the schedule is identifying any dependencies between the tasks. For example, it does not make sense to try to design the database before we have identified the information requirements. But many tasks can be done in parallel. Again, the great benefit of a single iteration is that we can make the schedule informal, and we will be able to adjust the work day by day to respond to specific complexities that occur. For our project, we will not build a complete schedule. You will learn how to do that in a later chapter. However, in order to organize our six-day project, we have taken the tasks from the Work Breakdown Structure and placed them on a day-by-day sequence that we call a work sequence draft, as shown in Figure 1-7. To develop a formal schedule, the project leader will use this diagram to assign people to the tasks as well as put the tasks on a specific schedule chart with calendar dates.

You should be aware that the sequence of activities and the dependencies of those activities are represented in this diagram with only partial accuracy. For example, we show that programming does not start until design has finished. However, in reality, there may be some overlap between the two activities.

The benefit of a work sequence draft is threefold. First, it helps the team organize its work so there is enough time set aside to think through the critical design issues before programming begins. Second, it provides a measuring rod to see if the iteration is on schedule. For example, if meetings with the purchasing agents take all day or more than a day, the team will know early on that this iteration will take longer than expected. Third, the project leader can see that programming may require more resources if the project is going to stay on this schedule. Hence, the project leader can begin lining up resources early on to help with that part of the project. It should be obvious that even this simple dependency diagram can help a project manager plan and organize the work.

Day 2 Activities

Day 1 involved planning and organizing the project. Day 2 involves systems analysis activities that help us understand and document requirements. On Day 2, we will specify the functions in more detail. These activities are included:

- Do preliminary fact-finding tasks to understand the requirements. (Core Process 3)
- Develop a preliminary list of use cases and a use case diagram. (Core Process 3)
- Develop a preliminary list of classes and a class diagram. (Core Process 3)

Fact Finding and User Involvement

Before the project commenced, a preliminary, broad definition of functions was developed. It is now time to examine the specifics of those functions and determine exactly what the user needs the system to do. The first step is to identify the key users that will help define these details. Obviously, the manager of the purchasing department will be one of the first ones to meet with. She will probably designate one or two knowledgeable purchasing agents who can work with the team on an ongoing basis to develop the specifications and to verify that the system performs as required. All successful projects depend on heavy user involvement. In Chapter 2, you will learn more about identifying key stakeholders.

There are various techniques to ensure that the fact finding is complete and thorough. These include interviewing the key users, observing existing work processes, reviewing existing documentation and existing systems, and even researching other companies and other systems.

Use Case	Description
Look up supplier	Using supplier name, find supplier information and contacts
Enter/update supplier information	Enter (new) or update (existing) supplier information
Look up contact	Using contact name, find contact information
Enter/update contact information	Enter (new) or update (existing) contact information
Look up product information	Using description or supplier name, find product information
Enter/update product information	Enter (new) or update (existing) product information
Upload product image	Upload images of the merchandise product

Figure 1-8: list of use cases

Identifying Use Cases

A use case documents a single user-triggered business event and the system's response to that event. For example, let us say a purchasing agent goes to a trade show and finds some new lightweight sports jackets that will work well for RMO's fall merchandise offerings. Maybe the first task the purchasing agent needs to do is find out if this supplier has worked with RMO before. Thus, the business event that requires the Tradeshow System might be "Look up a supplier." Activities leading up to the event of using the system are important, but we do not identify them as business events until the Tradeshow System is used—hence, the term use case—a case or situation where the system is used. One good way to help you identify use cases is to say, "The purchasing agent 'uses' the system to 'Look up a supplier.'"

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

There are multiple methods used to identify use cases, which you will learn about later in this book. Figure 1-8 is a preliminary list of use cases for the entire Tradeshow System. When the project team meets with the purchasing agents in brainstorming sessions, they together identify every business event in which a purchasing agent might use the system. However, because this first iteration is focusing only on the Supplier Information subsystem, the project team will also focus its attention on only the first four use cases on the list.

Identifying Object Classes

Object classes identify those things in the real world that the system needs to know about and keep track of. In order to find object classes, we look for all the objects, or things, that the system uses or captures. Objects come in all types and variations, from tangible items (such as merchandise products that you can see and touch) to more abstract concepts that you cannot touch (such as an order), which, though intangible, definitely exist.

Object classes are identified in the discussions with purchasing agents by looking for the nouns that describe categories of things. For example, the agents will often talk about suppliers, merchandise products, or inventory items. More details about how to identify object classes and their attributes are provided later in this book.

Object Classes	Attributes
Supplier	supplier name, address, description, comments
Contact	name, address, phone(s), email address(es), position, comments
Product	category, name, description, gender, comments
ProductPicture	ID, image

Figure 1-9: List of object classes

Figure 1-9 illustrates which nouns have been determined to be fundamental object classes for the Tradeshow System. The attributes are descriptors that help define and describe an object class.

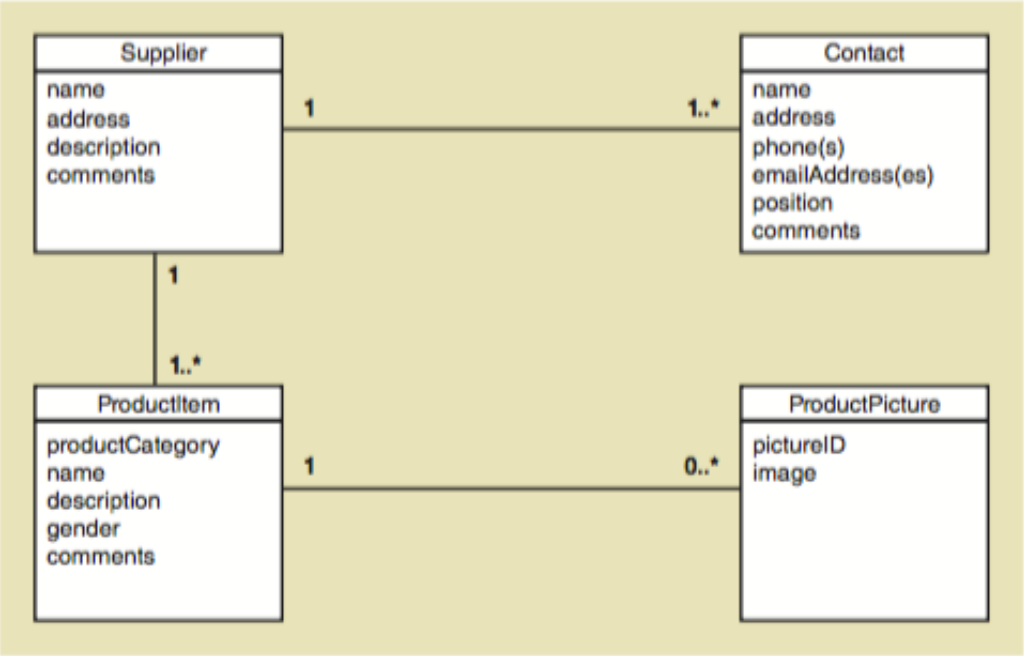

Figure 1-10: Preliminary class diagram

In addition to just providing a list of object classes, systems analysts often develop a visual diagram of the classes, their attributes, and their relationships to other classes. This diagram is called a class diagram. Figure 1-10 illustrates a class diagram for the Tradeshow System.

Each box is a class and can be thought of as a particular set of objects that are important to the system. Important attributes of each class are also included in each box. These represent the detailed information about each object that will be maintained by the system. Note that some classes have lines between them. These represent relationships between the classes that need to be captured in the system. For example, a contact is a person who works for a particular supplier. A specific example might be that Bill Williams is the contact person for the South Pacific Sportswear Company. Thus, the system needs to associate Bill Williams and the South Pacific Sportswear Company. The relationship line documents that requirement.

Class diagrams are a powerful and frequently used way to understand and document the information requirements of a system. The Tradeshow System is extremely simple, with only four classes identified—two of which belong to the Supplier Information subsystem. Most real-life systems are much larger and have dozens of classes.

Day 3 Activities

The purpose of Day 3 activities is to analyze in detail those use cases and classes that were selected to be implemented in this first iteration. During Day 3, we are still performing processes that are considered systems analysis. We are still trying to understand the requirements at a detailed level for the system. Included are these activities:

- Perform in-depth fact finding to understand details. (Core Process 3)

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

- Understand and document the detailed workflow of each use case. (Core Process 3)
- Define the user experience with screens and reports. (Core Processes 3 and 4)

It is important to note that the use cases help the project team organize its work. These drill-down activities are done for each use case. As mentioned earlier, these use cases pertain to the Supplier Information subsystem:

- Look up supplier.
- Enter/update supplier information.
- Look up contact information.
- Enter/update contact information.

The project team will develop a workflow for each use case to better understand how it works and to identify what screens and possibly what reports will need to be developed. As the team gets more into the details, it may discover that some of the initial analysis is incomplete, if not incorrect. This is a good time to make such discoveries—much better than after the programs have been written.

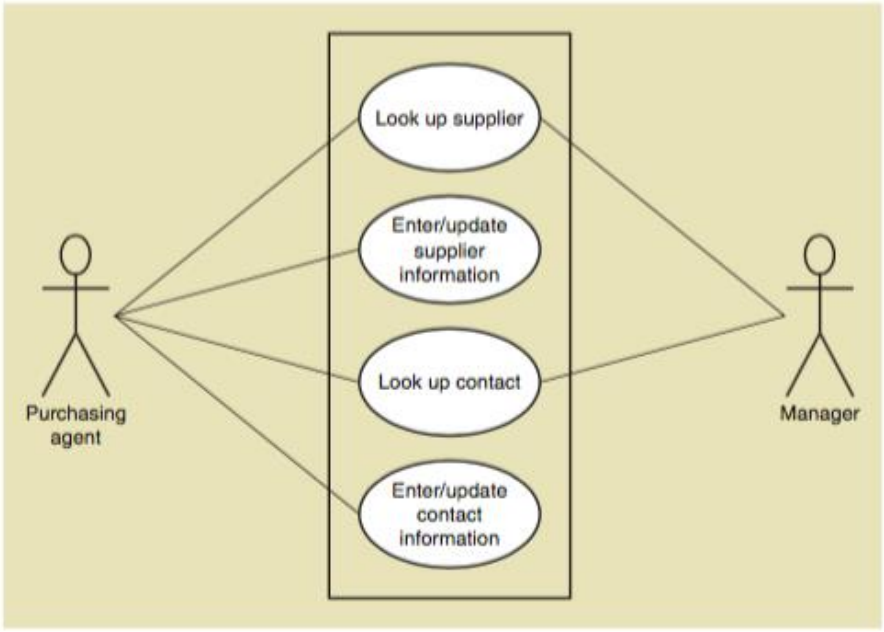

Figure 1-11: Use case diagram

Figure 1-11 illustrates a simple use case diagram. It shows the four aforementioned use cases identified and the user who will be the primary person performing that function. What the diagram means in detail and how to develop one will be discussed in a later chapter.

Developing Use Case Descriptions and Workflow Diagrams

There are various methods for documenting the details of a use case. One that you will learn later in this text is called a use case description. Another method is developing a workflow diagram, which shows all the steps within the use case. The purpose with either method is to document the interactions between the user and the system (i.e., how the user interacts and uses the system to carry out a specific

work task for a single use case).

Let us develop a workflow for one use case. To develop a workflow, we will use a simple type of diagram called an activity diagram. Figure 1-12 illustrates the workflow for the Look up supplier use case. The ovals in the diagram represent the tasks, the diamonds represent decision points, and the arrows represent the sequence of the flow. The columns represent who performs which tasks. Usually, workflow diagrams are quite easy to understand.

The arrows that cross the center line represent the interactions between the system and the user. These are critically important because the developers must provide a screen or Web page that either captures or displays information. The arrows that cross the center line identify the data elements that become part of the user interface.

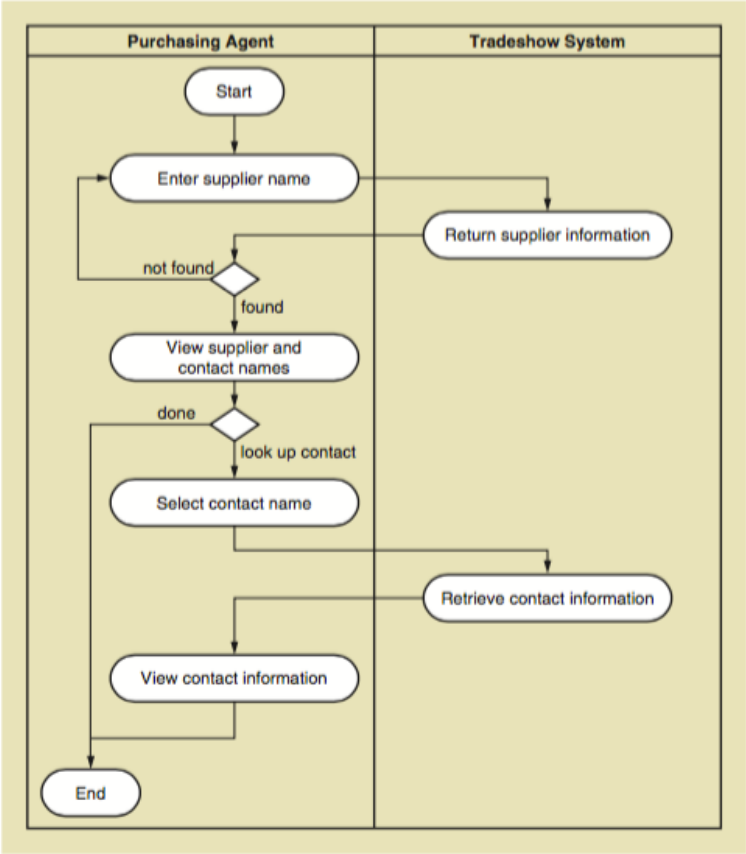

Figure 1-12: Workflow diagram for the Look Up Supplier use case

Looking at Figure 1-12, we see that the top arrow indicates that the supplier name enters into the system. Thus, we infer that the user must have an online form in order to enter the supplier name for the initial lookup. The next arrow indicates that there must be a form that displays all the details for an individual supplier, including a list of existing contacts. The user may also want to see more details about a specific contact person for this supplier, so the user may request detailed information for a particular person. Because the user can select one of the displayed results, it appears that we must design the form so each entry on the list is either a hotlink or has some mechanism to select it.

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

Defining Screen Layout

User-interface design includes all those tasks that describe the look and feel of the system to the user. Because the user interface is the window that the users work with to utilize the functionality of the system, the user interface is essentially the system. If the interface is poorly designed, users will not be able to take full advantage of the system; they may even consider the system to be less than optimal. On the other hand, a well-designed user interface—one that is intuitive and easy to use, with a full range of features to facilitate navigation, and that provides good information—will enhance the utility of the system tremendously.

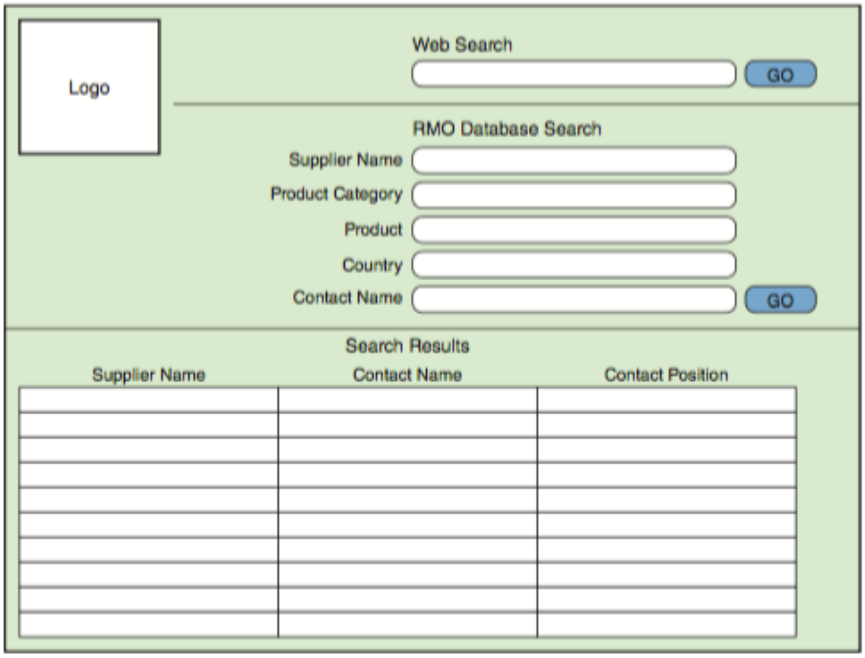

Figure 1-13: Draft of screen layout of the Look Up Supplier use case

Figure 1-13 illustrates the layout of the first screen used for the workflow in the use case Look up supplier. The top portion of the screen provides the locations for the user to enter the supplier information, and the bottom portion of the screen shows the results. When results are provided, the search box for data entry will remain visible to allow the user to enter another search. Each entry in the results will be built as a hotlink, so the user can click on any particular supplier to retrieve more detailed information. This drill-down technique is a common method used in today's systems and will be intuitively easy for the users.

Searches are conducted on the RMO database, resulting in such RMO information as name, address, and contact information. An Internet-wide search is also possible. This allows the purchasing agents to look for and view the suppliers' own Web sites, which can be helpful, as can looking at forums and discussions about the supplier. Note that this tangential activity is not captured in the workflow of Figure 1-12. It is intended to assist in screen design, not document everything the user can or will do.

Day 4 Activities

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

The primary focus of Day 4 activities is to design the various components of the solution system. Up to now, we have mostly been trying to understand the user requirements. On Day 4, we carry out design activities that direct programming efforts. In that sense, design activities can be considered somewhat of a bridge. During analysis activities, the project team's objective is to understand user needs. During programming, the objective is to produce the solution. Thus, design is the bridge between understanding and construction. It provides the outline for how the solution will be structured and how it is to be programmed. System design also tends to involve the technical people, with less need for user participation.

Design can be a complex process. In our small project, we will limit our design examples to only a few models and techniques. Later in this textbook, you will learn additional design techniques. Day 4 Activities include the following:

- Design the database structure (schema). (Core Process 4)
- Design the system's high-level structure. (Core Process 4)

Database design is a fairly straightforward activity that uses the class diagram as input and develops the detailed database schema that can be directly implemented by a database management system. Such elements as table design, key and index identification, attribute types, and other efficiency decisions are made during this activity.

Table Name	Attributes
Supplier	SupplierID: integer {key} Name: string {index} Address1: string Address1: string City: string State-province: string Postal-code: string Country: string SupplierWebURL: string Comments: string
Contact	ContactID: integer {key} SupplierID: integer {foreign key} Name: string {index} Title: string WorkAddress1: string WorkAddress2: string WorkCity: string WorkState: string WorkPostal-code: string WorkCountry: string WorkPhone: string MobilePhone: string EmailAddress1: string EmailAddress2: string Comments: string

Figure 1-14: Database schema for Supplier Information subsystem

Designing the high-level system structure and the individual programs can be an intricate and complex process. First, the overall structure of the system is designed, including identifying the subsystems and connections to other systems. Within each subsystem, decisions are made about individual programs,

such as user-interface programs, business logic programs, and database access programs. Then, at the lowest level, the login within each program is defined, including what program functions are required and what variables are used.

It is not uncommon for developers to begin writing program code as they develop portions of the design. It is a good idea to complete most of the structural design before writing code. But as the lower levels of the system are being designed, programming often begins. However, in the RMO Tradeshow System project, we will list them as separate activities.

Designing the Database

Designing the database uses the information provided by the class diagram to determine the tables, the columns in the tables, and other components. Sometimes, the database design is done for the entire system or subsystem. At other times, it is built piecemeal—use case by use case. To keep our project simple, we will just show the database design for the two classes that are required for the Supplier Information subsystem. Figure 1-14 shows the database schema for the Supplier Information subsystem. Two tables are defined: Supplier and Contact.

Approaching High-Level Systems Design

There are fundamental design principles that will guide you through systems and program design; they will be explained in detail later in this book. For now, we will describe the general approach to design.

One of the first questions encountered in systems design is how and where to start. So far, we have three types of documents that can provide specifications to help answer that question. We have use cases, with their accompanying documentation, such as use case workflow diagrams. We have a class diagram that will help us identify some of the object-oriented classes that will be needed in the system. (In the previous section, we used the class diagram as the basis for the database design. Those same classes are important in developing object-oriented program classes.) Finally, we have screens and reports that also provide specifications for program logic and display logic.

Before we jump into design, let us briefly discuss the objective of systems design and what we expect to have as the output or result. Object-oriented programs are structured as a set of interacting classes. Therefore, in order to program, we need to know what those programming classes are, what the logic is within each class (i.e., the functions), and which programming classes must interact together. That is the final objective of systems design: to define the classes, the methods within those classes, and the interactions between classes.

We perform this design by starting at the very highest level and then drilling down to the lowest level until we have defined all the functions within each class. Detailed design is the thought process of how to program each use case. Later in the text, you will learn techniques to carry out detailed design. For Day 4, we will focus only on the overall design.

Designing the Overall Architecture

Figure 1-15 shows the overall architecture or structure of the new system. Although the figure itself appears rather simple, some important decisions have been involved in the development of this design. First, note that the decision was made to build this application as a browser-based system. A different

and very popular approach would have been to build smartphone or tablet applications. Browser-based systems sometimes do not provide the same connectivity speed and control as smartphone or tablet applications, but they are more versatile in that they can be more easily deployed on different equipment, such as laptops, without modification.

These high-level design decisions will determine the detailed structure of the system. A browser-based system is structured and constructed differently than an application system that runs on a smartphone or a tablet computer.

Defining the Preliminary Design Class Diagram

Given that the Tradeshow System will be built by using object-oriented programming (OOP) techniques, an important component of the design is developing the set of object classes and functions that will be needed for the system. This process can become quite detailed, and we will not try to explain it for this project. You will learn those techniques later in this book.

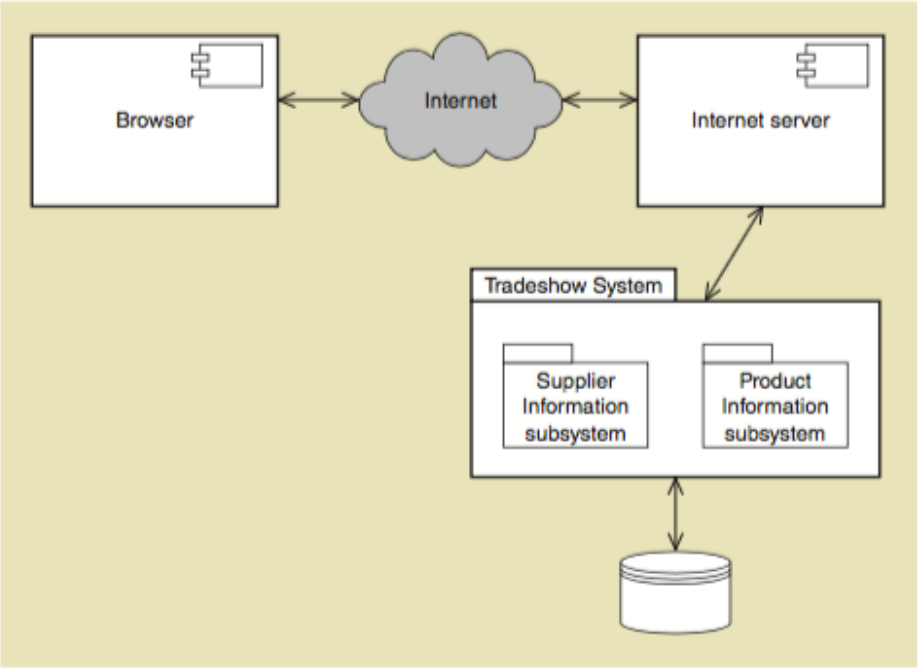

Figure 1-15: Tradeshow System architectural configuration diagram

Figure 1-16 is a preliminary design class diagram for the Tradeshow System. A design class diagram (DCD) identifies the OOP classes that will be needed for the system. The set of design classes includes problem domain classes, view layer classes, sometimes separate data access classes, and utility classes. In Figure 1-16, we show only the problem domain classes and the view layer classes. Problem domain classes are usually derived from those classes that were identified during analysis activities—hence, the name: problem (user need) domain classes. You will also notice that they very closely correspond to the database tables; in fact, in this simple project, they are almost exactly the same as the database tables. On more complex systems, they will be similar but not exactly the same. However, remember that programming classes are distinct from database tables.

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.

Other classes are required for the graphical user interface (GUI). In a dynamic Web system, such as the Tradeshow System, they are the classes that receive the input from the browser and format the output HTML files to be dis-played by the browser.

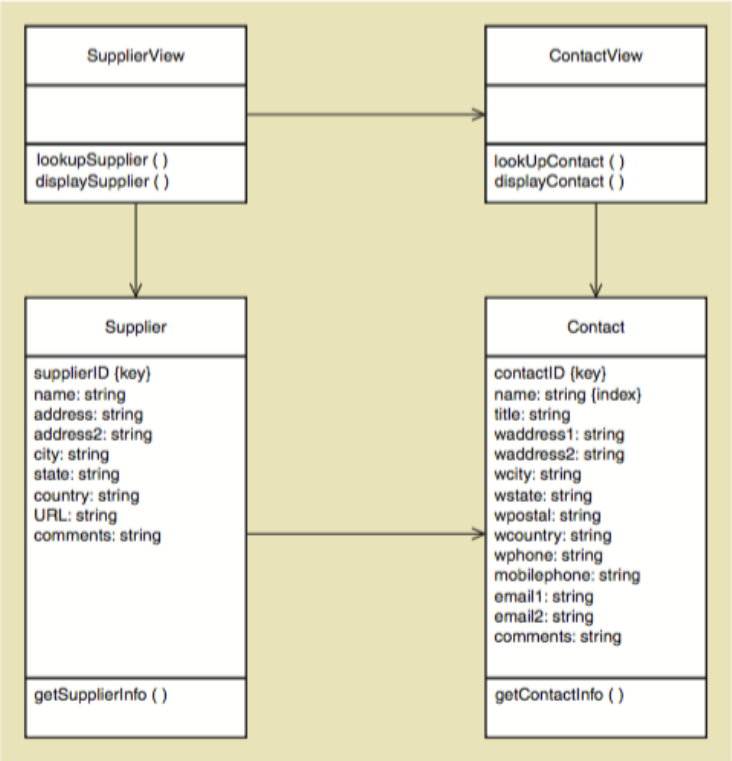

Figure 1-16: Preliminary design class diagram

The design classes in Figure 1-16 include the class-level variables that are needed for the class. These classes also show method names of the important methods within each class. These methods are identified and specified during high-level design and detailed design. One final element in the design class diagram are arrows that show which classes can access the methods of which other classes.

Designing Subsystem Architecture

Once we have an overall structure and an overall approach for implementing the new system, we begin to drill down to the subsystem design. Figure 1-17 illustrates the architectural design of the Supplier Information subsystem. Notice that this subsystem is further divided into layers: a view layer and a model layer. You will learn much more about multilevel design later in this textbook. One of the advantages of partitioning the system into layers is that the system is much easier to build and maintain with this kind of structure. For example, the system will be browser based, but different browsers require different techniques. It is better not to get these complexities mixed in with the basic program functions. Hence, they are separated out into a distinct layer.

Managing the Project

Design is a complex activity with multiple levels—from high-level structural design to low-level detailed program design. In our project, we have separated the tasks for designing the overall system

structure from detailed design of the programs themselves. However, these activities are often done concurrently. The basic high-level architectural structure is defined first, but mid-level and low-level design are often done concurrently with programming.

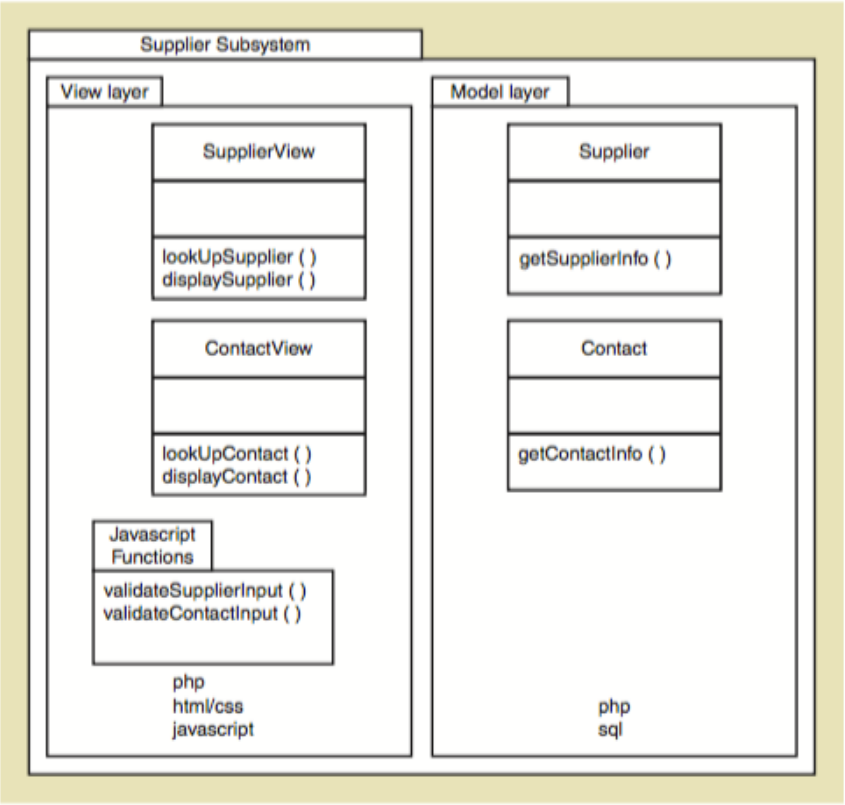

Figure 1-17: Supplier subsystem architectural design diagram

In Figure 1-17, we can see that detailed design and programming are quite time-consuming activities. A project manager must decide whether to extend the project or bring on additional programmers to help write the code. In our project, we have elected to insert a half-day of free time to bring in two additional programmers and train them. Of course, we could go ahead and begin Day 5's activities to ensure that we keep the project on schedule.

Day 5 Activities

Even though detailed design and programming may frequently begin earlier in the project, we have identified it as a separate day's activities. We do this for a couple of very important reasons. First, we want to emphasize that it is not a good practice to begin programming before critical information is obtained and decisions are made. Often, novice programmers will begin to program before the users' needs are adequately understood or even before the structure of the overall system has been determined. But a much better approach is to understand, design, and build small chunks of the system at a time. Agile Development anticipates and plans for the expected changes and refinements to the problem requirements that happen during detailed design and programming.

As the programmers write the code, they also perform individual testing on the classes and functions they program. This textbook does not focus on programming activities. However, we include an

example of program code so you can see how systems design relates to the final program code. Figure 1-18 is an example of a class that receives and processes the request for supplier information.

Day 6 Activities

The focus of Day 6 activities is to do the final testing that is required before the system is ready to be deployed. There are many types of testing that are required. In this example, we mention only two types of testing: overall system functional testing and user acceptance testing. Functional testing is usually a system-level test of all user functions and is often done by a quality assurance team. User acceptance tests are similar in nature, but they are done by the users, who test not only the correctness of the system but its "fitness" to accomplish the business requirements.

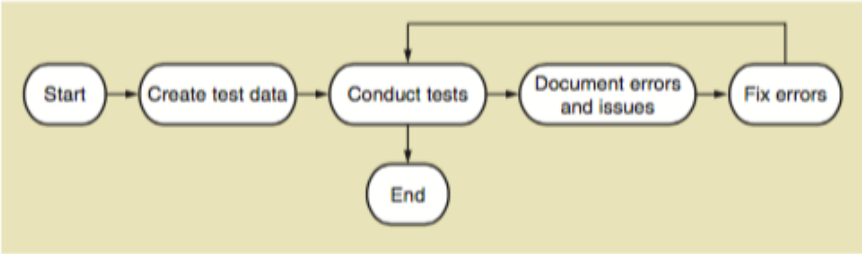

Figure 1-19: Generalized workflow of testing tasks

Each of the various testing activities in Day 6 has a somewhat similar sequence of tasks to perform. The tasks themselves highly depend on the test data and on the method for testing a particular test case. In some instances, the testing may be automated. In others, individuals may need to manually conduct the tests. Many new systems are interactive systems with user activity. There are some testing tools that automate that process somewhat, but it tends to be a rather complex task.

First Iteration Recap

Figure 1-20 is a screenshot of the browser page that is used in the Tradeshow System to enter and view suppliers.

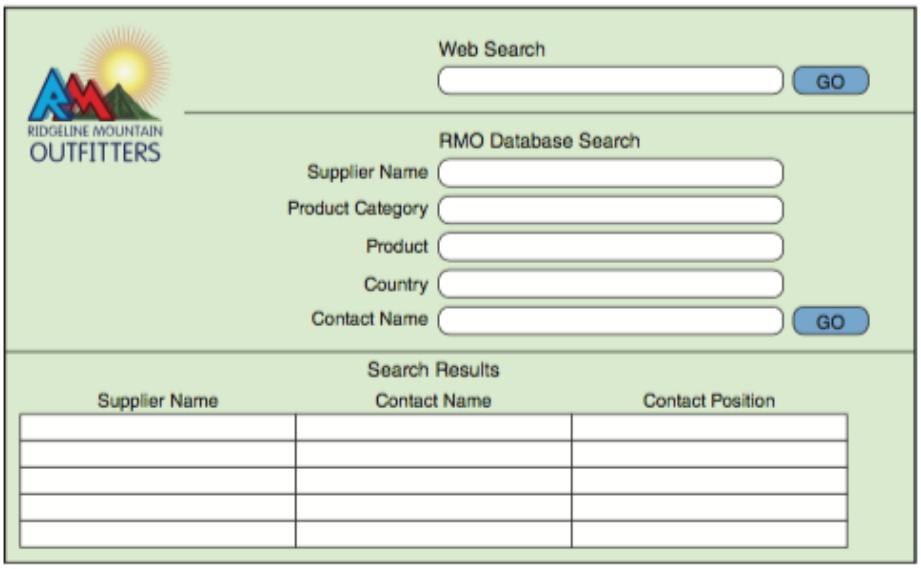

Figure: 1-20: Screen capture for Look Up Supplies use case

As stated previously, this is the first (six-day) iteration of a longer project. Using Agile techniques and iterations within an overall project allows flexibility in defining and building a new system. One of the Agile mandates is that the user should be heavily involved in the development of the new system. In this six-day project, the users have had major involvement during all days except Day 4 and Day 5.

A primary problem in developing a new system is that as the project progresses, new requirements are often identified. This happens because the users and the project team learn more about how to solve the business need. Agile, iterative projects are structured to handle these new requirements—often by adding another iteration to the overall project.

As a final step in a current iteration, or perhaps as part of the planning process for the next iteration, there should be a review of the processes and success of the current iteration. The lessons learned and issues to be carried forward create an environment of continual improvement and refinement. Iterative projects tend to improve and become more efficient during the life of the project.

From: Systems Analysis and Design in a changing world. John W. Satzinger, Robert B. Jackson, Stephen D. Burd, 2016.