

# Object Modeling &

---

# Flow Diagramming

---

# For Designers

---

Methods for designing with ease and confidence

◆◆◆ Heidi P. Adkisson

# Objects

As a designer, you're probably already familiar with the idea of objects, though you may not think about them as such. For example, typical objects in an e-commerce system include *Product*, *Bag*, *Wish List*, and *Review*.

Each object has a set of associated actions. As shown in Figure 2.1, actions for the *Product* object might include *Search For*, *Add to Bag*, *Apple Pay*, *Add to Wish List*, *Add to Store Pickup*.

Product
Search For() Add to Bag() Apple Pay() Add to Wish List() Add to Store Pickup()

**Figure 2.1**

The *Product* object with associated actions

Thus, a system can be characterized (and modeled) according to its objects and actions.

## A Short History of Objects and Actions

The centrality of objects and actions in modern systems stems from the development, in the mid-1970s, of the graphical user interface (GUI). The GUI is such a part of our lives today that it's easy to forget what a user experience revolution it was at the time. Before the GUI, users had to master either a command-line interface or, more challenging yet, punch cards (Figure 2.2).



**Figure 2.2**

Computer punch card operator in the 1960s

Thankfully, punch cards have been relegated to the history bin. However, the command-line interface (CLI) lives on and does have its advantages for some types of computing tasks. Even if you've never used a CLI, you might be familiar with it as the Windows command prompt or the macOS terminal (Figure 2.3).

```
john — -bash — 80x24
Last login: Wed Jun 2 09:29:22 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
johns-air:~ john$ sudo softwareupdate -l
[Password:
Software Update Tool

Finding available software
Software Update found the following new or updated software:
* Label: macOS Big Sur 11.4-20F71
  Title: macOS Big Sur 11.4, Version: 11.4, Size: 11206447K, Recommended:
YES, Action: restart,
johns-air:~ john$
```

**Figure 2.3**

Example command-line interface

The command line can be a highly efficient method of interacting with a computer. However, there is a steep learning curve: users must type commands exactly and, to work efficiently, memorize commands. With the advent of the GUI, computing became wholly more accessible, requiring much less specialized knowledge.

The Xerox Star system (Figure 2.4), released in 1981, was the first commercially available system with a GUI. It inspired other early GUI-based computers, including the Apple Lisa (in 1983) and the Apple Macintosh (in 1984).

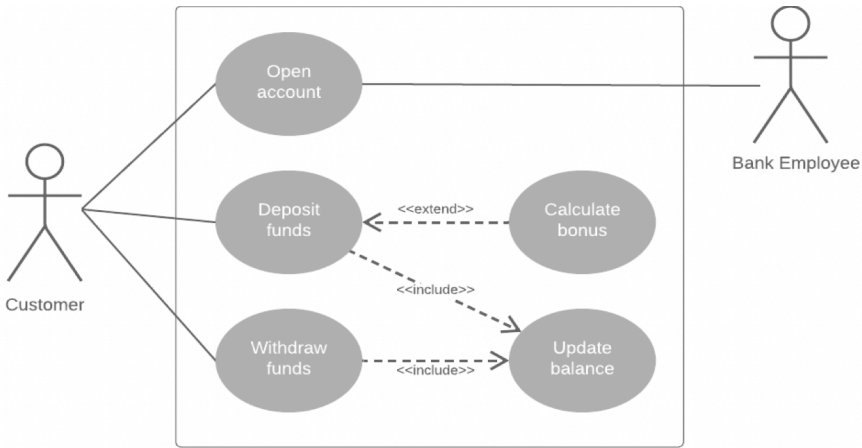


**Figure 2.4**  
The Xerox Star system (1981)

## Modeling Objects

Today objects and actions are foundational to any user experience. As we'll see in the chapters that follow, **modeling a system's objects** can help streamline the design process and improve design outcomes.

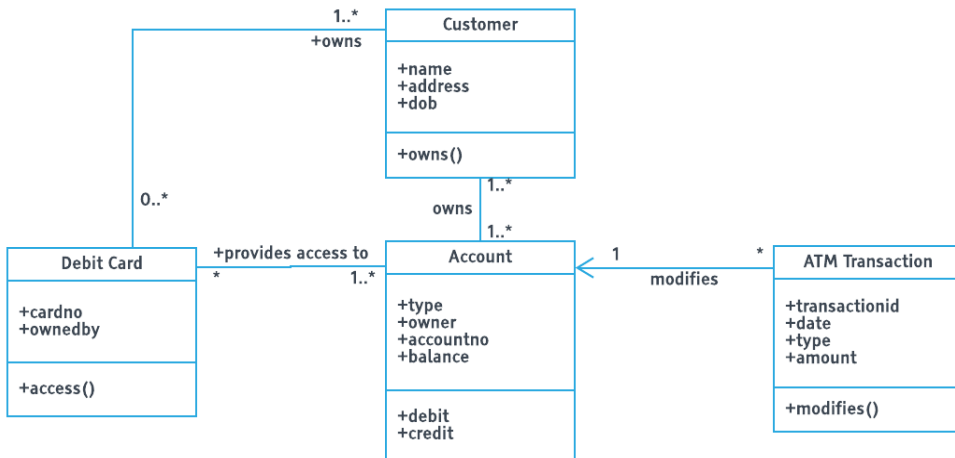
Readers familiar with the Unified Modeling Language (UML) will recognize that the practice of modeling objects is not new. UML is a notational system created in the 1990s to represent system design using a standard set of diagrams. In 1994, Rational Software developed UML as part of its suite of developer tools. Since then, the International Organization for Standardization (ISO) has managed UML as a standard. There are two main types of UML models: **behavioral** and **structural**. As a designer, you might be familiar with a use case model, a behavioral model that represents tasks supported by a system. Figure 2.5 is an example of a use case model for a banking system.



**Figure 2.5**  
Use case model for a banking system

To understand object modeling, we will focus on a structural model: the **UML class diagram**. The class diagram illustrates a system’s objects and the relationships between them.

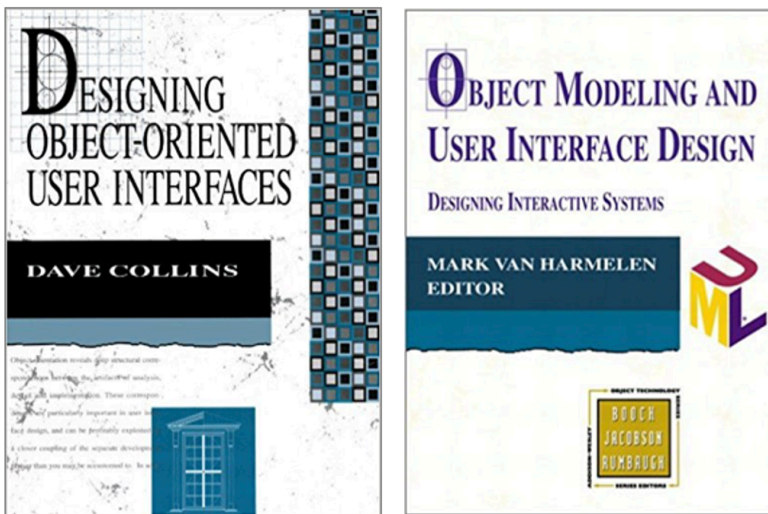
Figure 2.6 is a simplified UML class diagram that illustrates the conventions:



**Figure 2.6**  
Simplified UML class diagram for an ATM. You can also view this diagram at [objectmodelingfordesigners.com/chapter-2](http://objectmodelingfordesigners.com/chapter-2)

- The top pane of the object rectangle contains the name of the object.
- The lower pane displays each object's associated actions.
- The middle pane lists the object's attributes (associated data fields).
- Lines between the objects indicate relationships.

While the class diagram may be less familiar to designers today, its original intent was to serve as the basis of object-oriented user interface design. In fact, in the late 1990s and early 2000s, entire books were written on this topic (Figure 2.7).

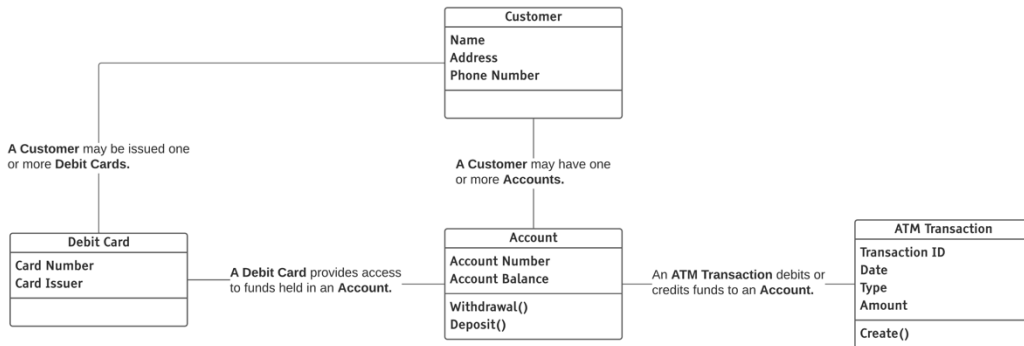


**Figure 2.7**  
Books on object-oriented interface design

### The Narrative Object Model

Creating a complete UML class diagram can be time-consuming and requires a set of skills more typical of an engineer or analyst. It's not something that a UX professional today would likely prepare. We can, however, take inspiration from the traditional UML class diagram and create a simpler, more accessible version—what I call a narrative **object model**. A narrative object model (mostly) dispenses with the UML notation and instead relies on a plain-English narrative.

Figure 2.8 is the ATM class diagram example (Figure 2.6) recreated as a narrative object model.



**Figure 2.8**

Narrative object model for an ATM. You can also view this diagram at [objectmodelingfordesigners.com/chapter-2](http://objectmodelingfordesigners.com/chapter-2)

In the chapters that follow, we'll walk through the process of creating a narrative object model, breaking the process into three main steps:

- 1) Identifying objects
- 2) Characterizing the relationships between objects
- 3) Identifying each object's actions and attributes

I advocate keeping your approach to object modeling flexible: to create the model that best serves your design needs. For example, building a complete model might not always make sense; sometimes, even just identifying the objects is enough. Later chapters contain examples of simplifying the modeling approach for different design circumstances.

### Tools You'll Need

You can use any drawing program to create an object model; however, it should support dynamic connectors. It's ideal to have a palette with the standard line treatments for UML class diagrams since we are re-purposing some of those conventions. The models you see in this book were built using Lucidchart, but any similar application will do.

Building an initial model can also be done collaboratively as a team activity. To do this in person, all you need is a set of sticky notes and a whiteboard or a large expanse of paper (such as butcher paper). Virtually, you can use a collaborative working space such as Mural. For more on creating models collaboratively, see

[objectmodelingfordesigners.com/resources](http://objectmodelingfordesigners.com/resources)

## Starting Points and Project Types

Creating any object model begins with identifying its objects. Depending on your design project type, you'll likely have a mix of resources to use as your starting point. This book will walk you through how to use two primary sources:

- User stories (or similar requirements documentation)
- A system's existing interface

The mix of sources you will rely upon depends on the type of design project at hand, as summarized in Figure 2.9.

Type of project	Existing interface	Requirements documentation
Greenfield design		◆
Redesign/ re-platforming	◆	◆
Feature release	◆	◆

**Figure 2.9**

Types of design projects and sources for object modeling

**Greenfield design** is designing a new application or system from the ground up; there is no existing starting point, and you are building the experience from scratch. In this case, you are likely only working from user stories or other types of requirements resources. Following a Lean methodology, the initial release in a greenfield design project may be a Minimal Viable Product (MVP).

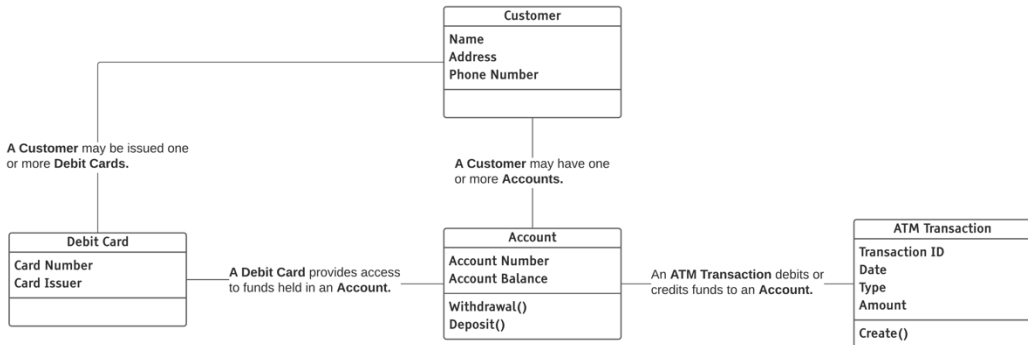
In the purest sense, a **redesign** takes an existing system and redesigns it to improve the user experience. This type of redesign commonly occurs with legacy systems and may also involve re-platforming (for example, moving it from a desktop client to the browser). In this case, you may be creating the model primarily or even exclusively by inspecting the current interface without user stories to reference.

A **feature release** is adding new features to an existing system. In this case, I advocate modeling the new features as part of modeling the system as a whole, referencing both the current interface and the requirements documentation for the new features. This approach helps ensure the new features are appropriately architected into the overall experience.



# Identifying Objects

The easiest way to think about objects is that they are the nouns in a system (rather than the verbs, which represent actions). Each object is represented by a rectangle in the model, as shown in Figure 2.10.



**Figure 2.10**  
 Narrative object model for an ATM. You can also view this diagram at [objectmodelingfordesigners.com/chapter-2](http://objectmodelingfordesigners.com/chapter-2)

It’s important to understand the difference between an object, which represents a concept, and an instance of an object. In Figure 2.11, *Customer* is the object, and the instances of that object are the specific database records for Jose E. Hamill, Jamie Averett, and Tiffany Scott.

Object	Instances							
<table border="1"> <tr><th>Customer</th></tr> <tr><td>Name</td></tr> <tr><td>Address</td></tr> <tr><td>Phone Number</td></tr> <tr><td> </td></tr> </table>	Customer	Name	Address	Phone Number		Jose E. Hamill 4310 Ben Street Sedalia, MO 65302 518-269-9354	Jamie Averett 2446 Taylor Street New York, NY 10013 914-598-6717	Tiffany Scott 934 Goodwin Avenue Omak, WA 98841 509-429-4885
Customer								
Name								
Address								
Phone Number								

**Figure 2.11**  
 The *Customer* object and example instances of that object

To identify objects, we’ll use two different resources:

- An existing interface
- User stories

## Identifying Objects from an Existing Interface

By first walking through how to identify objects from an existing interface, we'll see examples of how objects are represented in a design. Menus and button bars typically provide fruitful "hunting grounds" for objects.

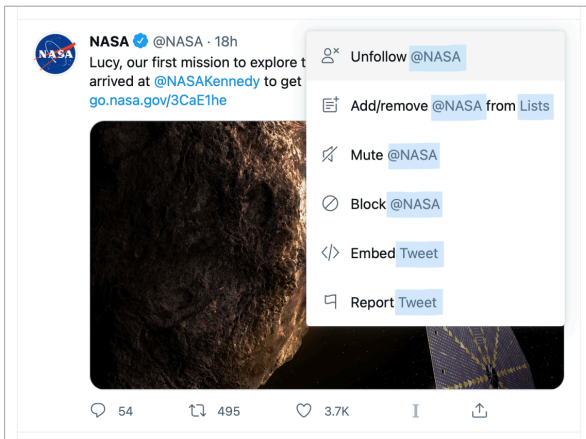
Figure 2.12 shows a tweet from Twitter.com. It has a contextual menu at the top (A), a button bar at the bottom (B), and another contextual menu at the bottom (C). We'll use these elements to identify some nouns (objects).



**Figure 2.12**

Button bar and menus for a tweet

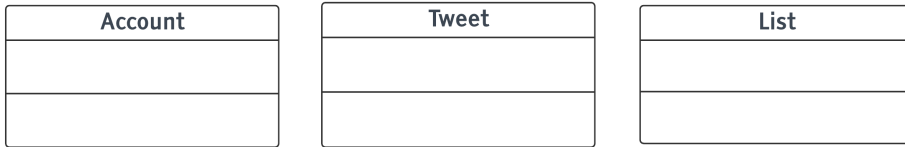
Figure 2.13 is the top contextual menu for a tweet with nouns highlighted.



**Figure 2.13**






Nouns highlighted in the contextual menu for a tweet

One object is *Account*, represented by the account name NASA (NASA being an instance of an account). *Tweet* and *List* are also objects. Figure 2.14 illustrates the objects.



**Figure 2.14**  
Objects from Twitter

The lower button bar contains options represented by unlabeled icons. Figure 2.15 shows the icons and their associated labels so that we can identify any additional objects.

-  Reply to Tweet
-  Retweet
-  Like this Tweet
-  Save Tweet to Instapaper
-  (More Actions Menu)

**Figure 2.15**  
Options represented in the button bar for a tweet

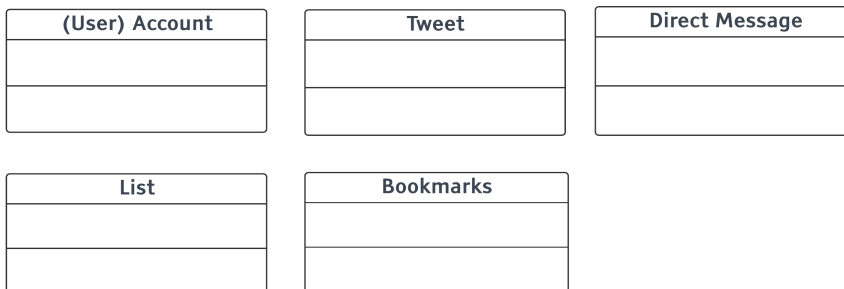
In this case, the icons represent actions that users can take against a tweet—an object we’ve already identified.

The last icon in the button bar, which looks like a share menu, is actually a “more actions” menu. Figure 2.16 shows this menu with its nouns highlighted.



**Figure 2.16**  
Nouns highlighted in the contextual menu for a tweet

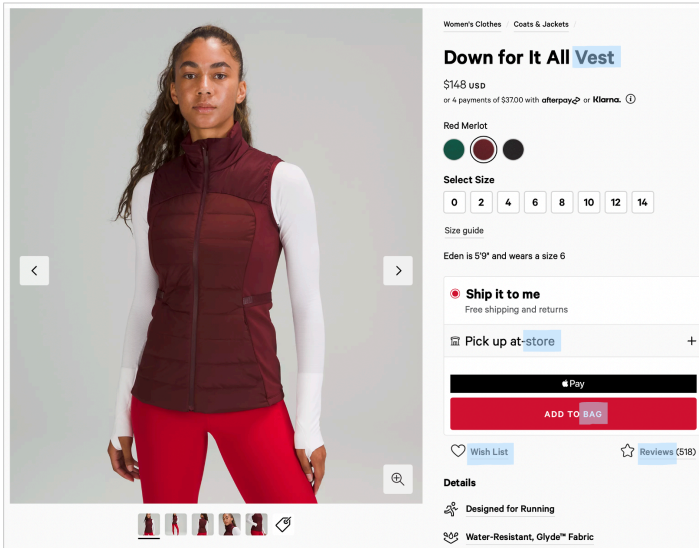
The “more actions” menu shows the *Tweet* object again, plus two new objects: *Direct Message* and *Bookmarks*. Figure 2.17 shows the consolidated list of Twitter objects we’ve identified so far.



**Figure 2.17**  
Objects identified for Twitter

Note that I renamed the Account object (*User*) *Account*. I did this to indicate the type of account (versus, for example, an administrator account). Additionally, as we’ll see once we define relationships, the (*User*) *Account* label makes the model a bit easier to read.

Next, let’s look at a different example from an e-commerce system. Figure 2.18 is a product detail page with its objects highlighted.



**Figure 2.18**  
Product detail page with its objects highlighted

As shown in Figure 2.19, the objects we identified on the above product detail page are *Product*, *Store*, *Bag*, *Wish List* and *Review*.



**Figure 2.19**  
Objects identified on the product detail page

In looking at the product detail page, you might wonder how to treat color and size. In this case, color and size are **attributes** of a *Product*, not objects in and of themselves, as is the case with the size guide and details content. Additionally, the photo of the product is not an object; it represents the *Image* attribute of a *Product*. Attributes are covered in a later chapter but think of them as data fields that would be part of a database record for an object.

## Identifying Objects from User Stories

Another way to identify objects is using requirements documentation. We are going to focus on user stories, given their widespread use.

User stories should be familiar to designers who work in Agile development environments. They are short, simple expressions of system requirements told from the user's perspective and follow a standardized syntax as shown below:

Syntax: As a <type of user>, I want <some goal> so that <some reason>.

Example: As an unregistered user, I want to create an account so that I use the service.

We can use the goal statement part of a user story, which begins with “I want...” to identify objects. This statement expresses what action the user wants to take, and what objects are involved. The example below shows the goal statement in bold text.

Syntax: As a <type of user>, **I want <some goal>** so that <some reason>.

Example: As an unregistered user, **I want to create an account** so that I can use the service.

Here are some additional goal statements highlighted in a set of sample user stories for Twitter:

- As an unregistered user, **I want to create an account** so that I can use the service.
- As a user, **I want to send a tweet** so that I can communicate to my followers.
- As a user, **I want to add a hashtag to a tweet** so that my tweet can be grouped with other tweets that share the same hashtag.
- As a user, **I want to follow another person's account** so that I can see the tweets that they post.
- As a user, **I want to see tweets from all accounts I follow in a timeline** so they are all together in a single view.
- As a user, **I want to create a list of specific accounts** so that tweets from those accounts are together in a single view.
- As a user, **I want to add a tweet to my bookmarks** so I can easily refer to it later.
- As a user, **I want to send a direct message to another person's account** so

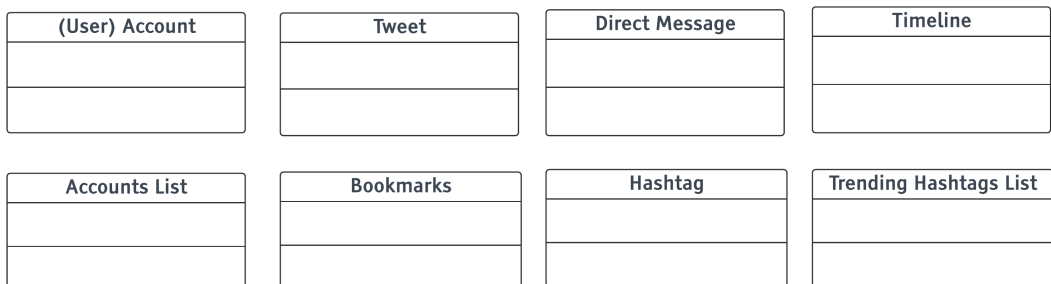
that I can communicate with them privately rather than publicly.

- As a user, I want to see a list of trending hashtags so that I can stay up-to-date on the hottest topics.

To identify the objects, we can highlight all the nouns within each goal statement as shown below:

- As an unregistered user, I want to create an **account** so that I can use the service.
- As a user, I want to send a **tweet** so that I can communicate to my followers.
- As a user, I want to add a **hashtag** to a **tweet** so that my tweet can be grouped with other tweets that share the same hashtag.
- As a user, I want to follow another person's **account** so that I can see the tweets that they post.
- As a user, I want to see **tweets** from all accounts I follow in a **timeline** so they are all together in a single view.
- As a user, I want to create a **list** of specific **accounts** so that tweets from those accounts are together in a single view.
- As a user, I want to add a **tweet** to my **bookmarks** so I can easily refer to it later.
- As a user, I want to send a **direct message** to another person's **account** so that I can communicate with them privately rather than publicly.
- As a user, I want to see a **list** of trending **hashtags** so that I can stay up-to-date on the hottest topics.

Figure 2.20 shows the Twitter objects identified from the user stories above.



**Figure 2.20**  
Objects identified from the Twitter user stories

Here's another set of example user stories with their objects highlighted; this set is from an e-commerce system:

- As a user, I want to search for **products** so that I can find products that meet my needs.
- As a user, I want to read **reviews** so I can learn what other people think about the product.
- As a user, I want to add **products** to my **wish list** so that I can save items that I don't want to purchase now but might want to later.
- As a user, I want to add **products** to my **bag** so I can purchase them when I'm done shopping.
- As a user, I want to pick up **products** I purchase at a nearby **store** so that I can receive them more quickly.
- As a user, I want to submit an **order** for **products** in my **bag** so that I can receive them.
- As a user, I want to create an **account** so that I can have access to my order information and purchase history.
- As a user, I want to review my **orders** so that I can see what I've purchased in the past.
- As a user, I want to create a **review** about the **product** I purchased so that I can share my opinion with potential buyers.

Figure 2.21 shows the objects identified from the e-commerce user stories.



**Figure 2.21**  
Objects identified in the e-commerce user stories

### Identifying Objects as a Team

Identifying objects can be a team activity, either using a “divide and conquer” strategy or working all together from the same set of sources. As you identify objects,



write each one on a sticky note, either a physical sticky note or in an online collaborative workspace. Group the duplicate sticky notes to create a consolidated list. Identifying objects as a team has the advantage of getting everyone involved in the process and creating a shared understanding of the objects.



**Figure 2.22**  
Identifying objects as a team

## Exercise A: Identifying Objects

Now it's your turn! For this exercise, you'll use a set of user stories for the online collaboration application Slack. (Don't worry if you're not familiar with Slack, you'll still be able to do this exercise.) Using the stories below, you'll do three things:

- 1) Identify the goal statements.
- 2) Highlight the nouns within each goal statement.
- 3) Create a list of objects.

The solution is in the next section.

- a) As an administrator, I want to sign up for an account so that we can use the service for our organization.
- b) As an administrator, I want to create one or more workspaces for my organization so that we will have our own dedicated space on Slack.
- c) As an administrator, I want to invite users to join a workspace so that they can communicate together.
- d) As an administrator, I want to create a channel so that posts on a particular topic can be kept all together within the channel.
- e) As a user, I want to create an account so that I can use the Slack service.
- f) As a user, I want to join a workspace so that I can communicate with others in that workspace.
- g) As a user, I want to create posts in a channel so that I can communicate with others in that channel.
- h) As a user, I want to send a direct message to another user so that we can communicate privately.
- i) As a user, I want to edit a post or direct message I've already published so that I can correct any mistakes.
- j) As a user, I want to delete a post or direct message so that people don't see posts that I've changed my mind about publishing.
- k) As a user, I want to attach a file to a post or direct message so that I can share it with others.
- l) As a user, I want to add a reaction to a post or direct message so that I can quickly show a response without having to type out a message
- m) As a user, I want to initiate a call to another user so we can quickly talk in real-time.