Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

First Let's look at a class diagram--Analysis class diagram-new.png.

The elements of a class-based model include classes or objects, attributes, operations, class-responsibility-collaborator (CRC) models, collaboration diagrams, and packages.

10.1 IDENTIFYING ANALYSIS CLASSES

We can begin to identify classes by examining the usage scenarios developed as part of the requirements model (Chapter 9) and performing a " grammatical parse " on the use cases developed for the system to be built.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Classes are determined by each noun or noun phrase and entering it into a simple table. Synonyms should be noted. If the class (noun) is required to implement a solution, then it is part of the solution space; otherwise, if a class is necessary only to describe a solution, it is part of the problem space.（e.g., add items to shopping cart）

Analysis classes manifest themselves in one of the following ways:

- External entities (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.
- Things (e.g., reports, displays, letters, signals) that are part of the information domain for the problem.
- Occurrences or events (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

- Roles (e.g., manager, engineer, salesperson) played by people who interact with the system.
- Organizational units (e.g., division, group, team) that are relevant to an application.(e.g., the group of project management system)
- Places (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.(e.g., the sensor position in SafeHome system)
- Structures (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects.(e.g.,control panel)

To illustrate how analysis classes might be defined during the early stages of modeling, consider a grammatical parse (nouns are underlined, verbs italicized) for a processing narrative for the SafeHome security function.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

The SafeHome security function *enables* the homeowner to *configure* the security system when it is *installed*, *monitors* all sensors *connected* to the security system, and *interacts* with the homeowner through the Internet , a PC or a control panel.

During installation , the SafeHome PC is used to *program and configure* the system. Each sensor is assigned a number and type , a master password is programmed for *arming and disarming* the system, and telephone number(s) are *input for dialing* when a sensor event occurs.

When a sensor event is *recognized*, the software *invokes* an audible alarm attached to the system. After a delay time that is *specified* by the homeowner during system configuration activities, the software *dials* a telephone number of a monitoring service , *provides* information about the location, *reporting* the nature of the event that has been detected. The telephone number will be *redialed* every 20 seconds until telephone connection is *obtained*.

The homeowner *receives* security information via a control panel, the PC, or a browser, collectively called an interface. The interface *displays* prompting messages and system status information on the control panel, the PC, or the browser window. Homeowner interaction takes the following form . . .

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Extracting the nouns, we can propose a number of potential classes:

Potential Class	General Classification
homeowner	role or external entity
sensor	external entity
control panel	external entity
installation	occurrence
system (alias security system)	thing
number, type	not objects, attributes of sensor
master password	thing
telephone number	thing
sensor event	occurrence
audible alarm	external entity
monitoring service	organizational unit or external entity

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

The list would be continued until all nouns in the processing narrative have been considered. Note that we call each entry in the list a "potential" class. We must consider each further before a final decision is made.

Coad and Yourdon suggest six selection characteristics that should be used as you consider each potential class for inclusion in the analysis model:

① Retained information. The potential class will be useful during analysis only if information about it must be remembered so that the system can function.

② Needed services. The potential class must have a set of identifiable operations that can change the value of its attributes in some way.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

③ **Multiple attributes**. During requirement analysis, the focus should be on "major" information; a class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.

④ **Common attributes**. A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.

⑤ **Common operations**. A set of operations can be defined for the potential class and these operations apply to all instances of the class.

⑥ **Essential requirements**. External entities that appear in the problem space and produce or consume information is essential to the operation of any solution for the system will almost always be defined as classes in the requirements model.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

To be considered a legitimate class for inclusion in the requirements model, a potential class should satisfy all (or almost all) of these characteristics. The decision for inclusion of potential classes in the analysis model is somewhat subjective,and later evaluation may cause an object to be discarded or reinstated.

However, the first step of class-based modeling is the definition of classes, and decisions (even subjective ones) must be made. With this in mind, you should apply the selection characteristics to the list of potential SafeHome classes:

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Potential Class	Characteristic Number That Applies
homeowner	rejected: 1, 2 fail even though 6 applies
sensor	accepted: all apply
control panel	accepted: all apply
installation	rejected
system (alias security function)	accepted: all apply
number, type	rejected: 3 fails, attributes of sensor
master password	rejected: 3 fails
telephone number	rejected: 3 fails
sensor event	accepted: all apply
audible alarm	accepted: 2, 3, 4, 5, 6 apply
monitoring service	rejected: 1, 2 fail even though 6 applies

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

It should be noted that (1) the preceding list is not all inclusive, additional classes would have to be added to complete the model; (2) some of the rejected potential classes will become attributes for those classes that were accepted (e.g., number and type are attributes of Sensor , and master password and telephone number may become attributes of System); (3) different statements of the problem might cause different "accept or reject" decisions to be made (e.g., if each homeowner had an individual password or was identified by voice print, the Homeowner class would satisfy characteristics 1 and 2 and would have been accepted).

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

10.2 SPECIFYING ATTRIBUTES

To develop a meaningful set of attributes for an analysis class, you should study each use case and select those "things" that reasonably "belong" to the class. In addition, the following question should be answered for each class: What data items (composite and/or elementary) fully define this class in the context of the problem at hand?

To illustrate, we consider the System class defined for SafeHome. A homeowner can configure the security function to reflect sensor information, alarm response information, activation/deactivation information, identification information, and so forth. We can represent these composite data items in the following manner:

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

identification information = system ID + verification phone number + system status

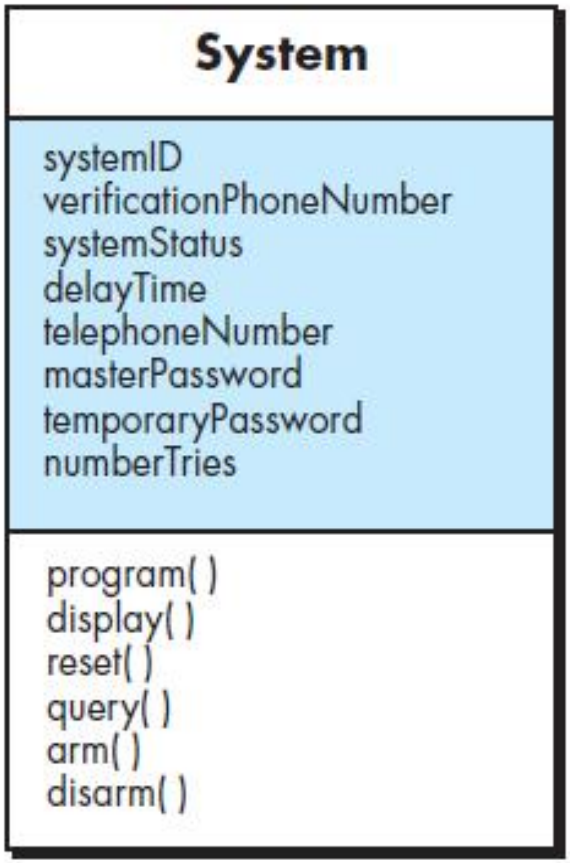
alarm response information = delay time + telephone number

activation/deactivation information = master password + number of allowable tries + temporary password

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

FIGURE 10.1

Class diagram for the system class



Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

10.3 DEFINING OPERATIONS

Operations define the behavior of an object. Although many different types of operations exist, they can generally be divided into four broad categories: (1) operations that manipulate data in some way (e.g., adding, deleting, reformatting,selecting), (2) operations that perform a computation, (3) operations that inquire about the state of an object, and (4) operations that monitor an object for the occurrence of a controlling event. These functions are accomplished by operating on attributes and/or associations (Section 10.5). Therefore, an operation must have "knowledge" of the nature of the class attributes and associations.

As a first iteration at deriving a set of operations for an analysis class, you can again study a processing narrative (or use case) and select those operations that reasonably belong to the class.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

To accomplish this, the grammatical parse is again studied and verbs are isolated. Some of these verbs will be legitimate operations and can be easily connected to a specific class.

For example, from the SafeHome processing narrative presented earlier in this chapter, we see that "sensor is assigned a number and type" or "a master password is programmed for arming and disarming the system." These phrases indicate a number of things:

- That an assign() operation is relevant for the Sensor class.
- That a program() operation will be applied to the System class.
- That arm() and disarm() are operations that apply to System class.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Upon further investigation, it is likely that the operation program() will be divided into a number of more specific suboperations required to configure the system. For example, program() implies specifying phone numbers, configuring system characteristics (e.g., creating the sensor table, entering alarm characteristics), and entering password(s). But for now, we specify program() as a single operation. In addition to the grammatical parse, you can gain additional insight into other operations by considering the communication that occurs between objects. Objects communicate by passing messages to one another.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

10.4 CLASS-RESPONSIBILITY-COLLABORATOR(CRC) MODELING

In reality, the CRC model may make use of actual or virtual index cards. The intent is to develop an organized representation of classes:

- ***Responsibilities*** **are the attributes and operations that are relevant for the class.**
- ***Collaborators*** **are those classes that are required to provide a class with the information needed to complete a responsibility. In general, a collaboration implies either a request for information or a request for some action.**

See Analysis class diagram.png again.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

A. **Classes：** **Basic guidelines for identifying classes and objects were presented earlier in this chapter. The taxonomy of class types presented in Section 10.1 can be extended by considering the following categories:**

- **Entity classes, also called model or business classes, are extracted directly from the statement of the problem (e.g., FloorPlan and Sensor).These classes typically represent things that are to be stored in a database and persist throughout the duration of the application (unless they are specifically deleted).**
- **Boundary classes are used to create the interface (e.g., interactive screen or printed reports) that the user sees and interacts with as the software is used. Entity objects contain information that is important to users, but they do not display**

themselves. Boundary classes are designed with the responsibility of managing the way entity objects are represented to users.

- Controller(interface) classes manage a "unit of work" from start to finish. That is,controller classes can be designed to manage (1) the creation or update of entity objects, (2) the instantiation of boundary objects as they obtain information from entity objects, (3) complex communication between sets of objects, (4) validation of data communicated between objects or between the user and the application. In general, controller classes are not considered until the design activity has begun.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

B. **Responsibilities:** identifying responsibilities (attributes and operations) , Wirfs-Brock and her colleagues suggest five guidelines for allocating responsibilities to classes:

① System intelligence should be distributed across classes to best address the needs of the problem. If system intelligence is more evenly distributed across the classes in an application, each object knows about and does only a few things (that are generally well focused), the cohesiveness of the system is improved. This enhances the maintainability of the software and reduces the impact of side effects due to change.To determine whether system intelligence is properly distributed, the responsibilities noted on each CRC model index card should be evaluated to determine if any class has an extraordinarily long list of responsibilities. This indicates a concentration of intelligence. In addition,the responsibilities for each class should exhibit the same level of abstraction.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

② **Each responsibility should be stated as generally as possible.** This guideline implies that **general responsibilities** (both attributes and operations) should reside high in the class hierarchy (because they are generic, they will apply to all subclasses).

③ **Information and the behavior related to it should reside within the same class.** This achieves the object-oriented principle called **encapsulation**. Data and the processes that manipulate the data should be packaged as a cohesive unit.

④ Information about one thing should be localized with a single class, not distributed across multiple classes. **A single class should take on the responsibility for storing and manipulating a specific type of information.** This responsibility should not, in general, be shared across a number of classes. If information is distributed, software becomes more difficult to maintain and more challenging to test.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

⑤ Responsibilities should be shared among related classes, when appropriate.There are many cases in which a variety of related objects must all exhibit the same behavior at the same time. As an example, consider a video game that must display the following classes: Player, PlayerBody,PlayerArms, PlayerLegs, PlayerHead. Each of these classes has its own attributes (e.g., position, orientation, color, speed) and all must be updated and displayed as the user manipulates a joystick. The responsibilities update and display must therefore be shared by each of the objects noted.

C. Collaborations： Classes fulfill their responsibilities in one of two ways:(1) A class can use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility, or (2) a class can collaborate with other classes.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Collaborations are identified by determining whether a class can fulfill each responsibility itself. If it cannot, then it needs to interact with another class. Hence, a collaboration.

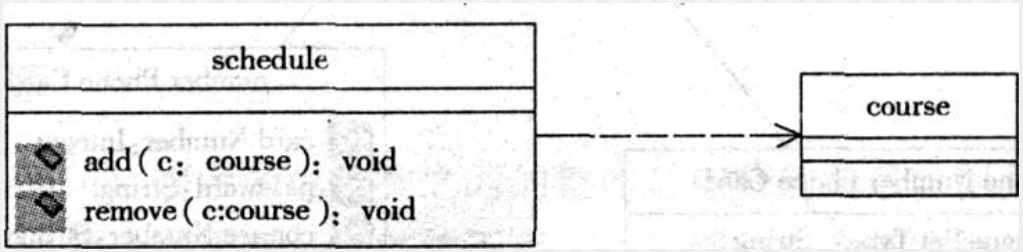
To help in the identification of collaborators, you can examine three different generic relationships between classes: (1) the is-part-of relationship, (2) the has-knowledge-of relationship, and (3) the depends-upon relationship:

- ***All classes that are part of an aggregate class are connected to the aggregate class via an is-part-of relationship. Consider the classes defined for the video game noted earlier, the class PlayerBody is-part-of Player , as are PlayerArms,PlayerLegs, and PlayerHead.***
- ***When one class must acquire information from another class, the has-knowledge-of relationship is established. The determine-sensor-status() responsibility noted earlier is an example of a has-knowledge-of relationship.***

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

- ***The depends-upon relationship implies that two classes have a dependency that is not achieved by has-knowledge-of or is-part-of. For example, PlayerHead must always be connected to PlayerBody. An attribute of* the PlayerHead object called center-position is determined from the center position of PlayerBody.**

  Another example:



When a complete CRC model has been developed, the representatives from the stakeholders can review the model using the following approach:

① **All participants in the review (of the CRC model) are given a subset of the CRC model index cards. Cards that collaborate should be separated (i.e.,no reviewer should have two cards that collaborate).**

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

② All use-case scenarios (and corresponding use-case diagrams) should be organized into categories.

③ The review leader reads the use case deliberately. As the review leader comes to a named object, she passes a token to the person holding the corresponding class index card.

④ When the token is passed, the holder of the class card is asked to describe the responsibilities noted on the card. The group determines whether one (or more) of the responsibilities satisfies the use-case requirement.

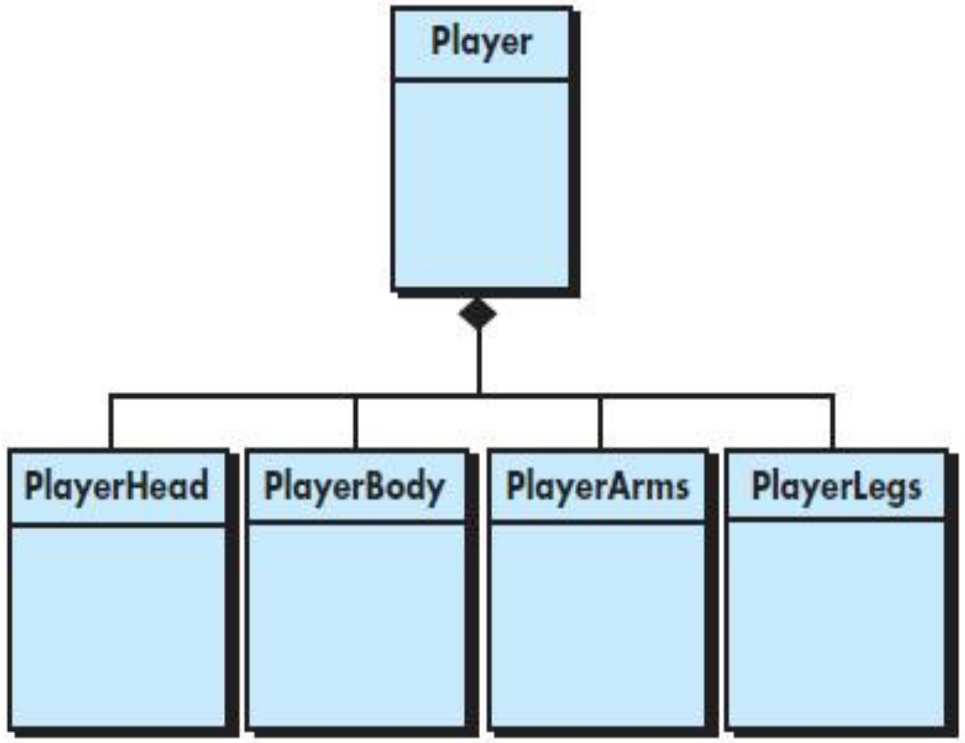
⑤ If the responsibilities and collaborations noted on the index cards cannot accommodate the use case, modifications are made to the cards. This may include the definition of new classes (and corresponding CRC index cards) or the specification of new or revised responsibilities or collaborations on existing cards.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

This modus operandi continues until the use case is finished.

FIGURE 10.4

A composite aggregate class



Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

10.5 ASSOCIATIONS AND DEPENDENCIES

In many instances, two analysis classes are related to one another in some fashion.

In UML these relationships are called *associations*. In some cases, an association may be further defined by indicating *multiplicity*.
See Analysis class diagram.png again.

.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

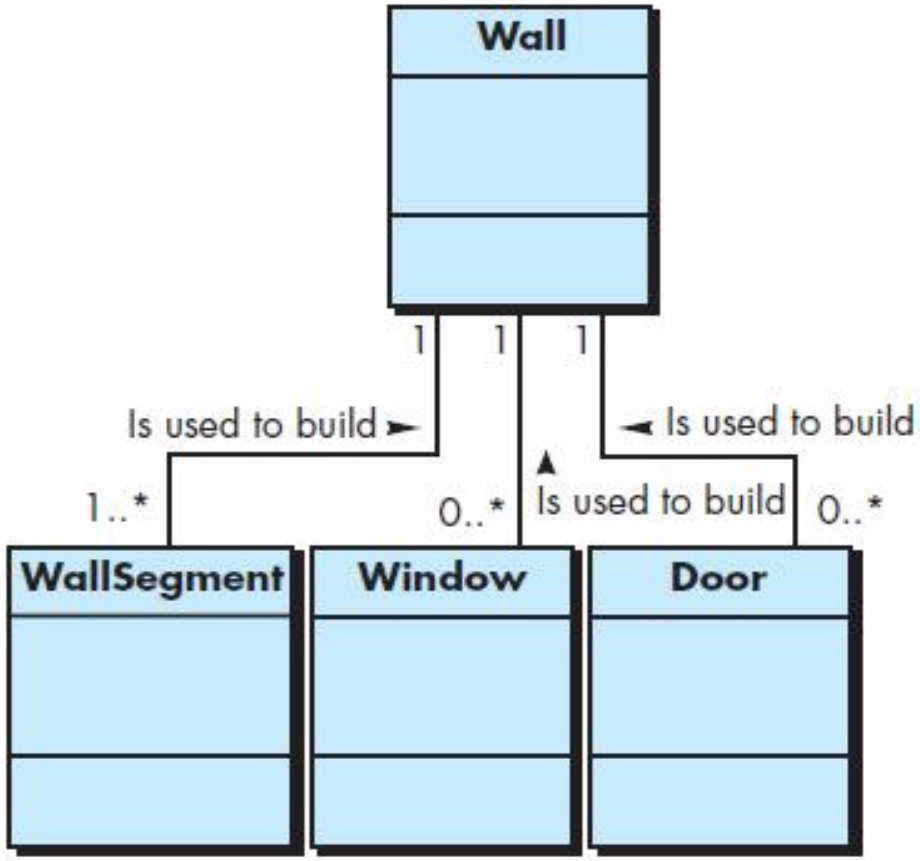
In Figure 10.5 , where “one or more” is represented using 1..*, and “0 or more” by 0..*. In UML, the asterisk indicates an unlimited upper bound on the range.

In many instances, a client-server relationship exists between two analysis classes. In such cases, a client class depends on the server class and *a dependency relationship* is established.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

FIGURE 10.5

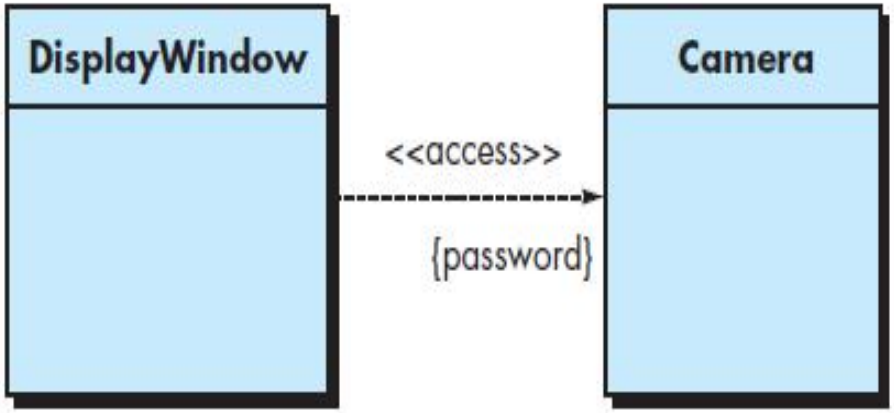
Multiplicity



Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

FIGURE 10.6

Dependencies



10.6 ANALYSIS PACKAGES

An important part of analysis modeling is categorization. That is, various elements of the requirements model (e.g., use cases, analysis classes) are categorized in a manner that packages them as a grouping—called an analysis package —that is given a representative name.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Analysis packages are used to categorize and group classes in a manner that makes them more manageable for large systems.

For a example:

the analysis model for the video game is developed, a large number of classes are derived:

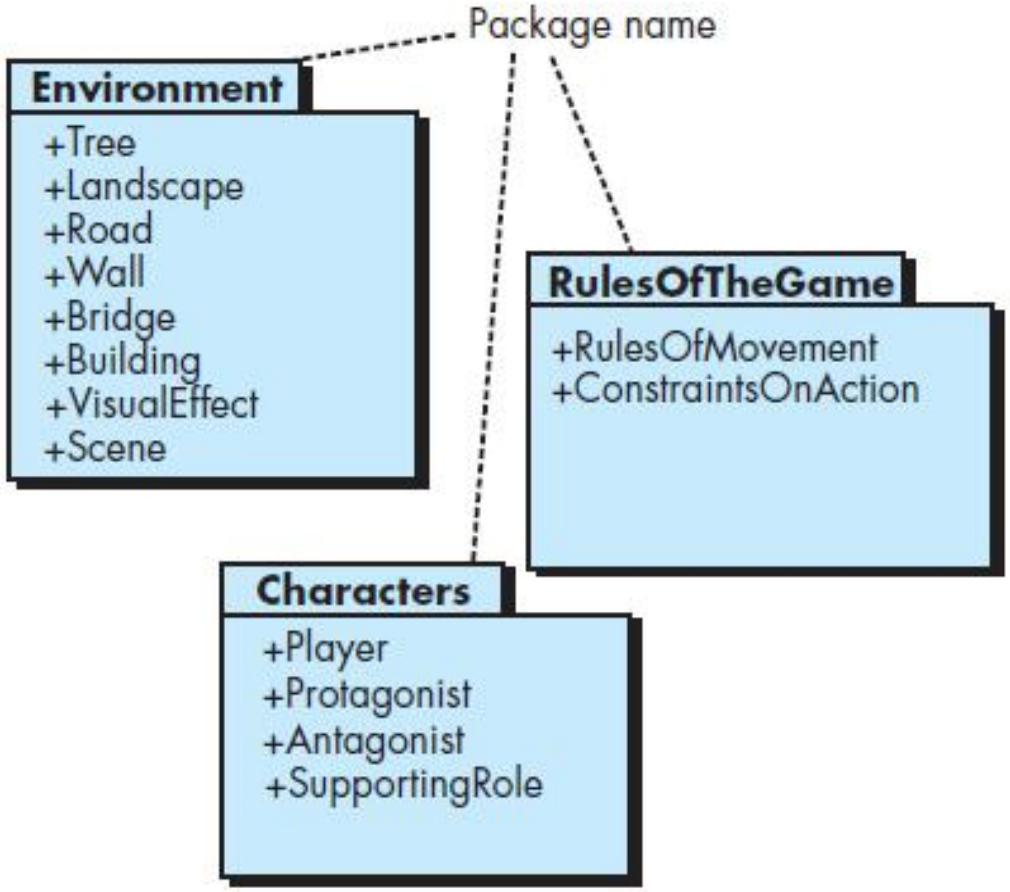
- **Some focus on the game environment.**
- **Others focus on the characters within the game.**
- **Still others describe the rules of the game.**

Analysis packages are shown in Figure 10.7.

Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

FIGURE 10.7

Packages



Chapter 10 REQUIREMENTS MODELING: CLASS-BASED METHODS

Summary:

Class-based modeling uses information derived from use cases and other written application descriptions to identify analysis classes. A grammatical parse may be used to extract candidate classes, attributes, and operations from text-based narratives. Criteria for the definition of a class are defined.

A set of class-responsibility-collaborator index cards can be used to define relationships between classes. In addition, a variety of UML modeling notation can be applied to define hierarchies, relationships(associations, aggregations,inheritance and dependencies)among classes. Analysis packages are used to categorize and group classes in a manner that makes them more manageable for large systems.