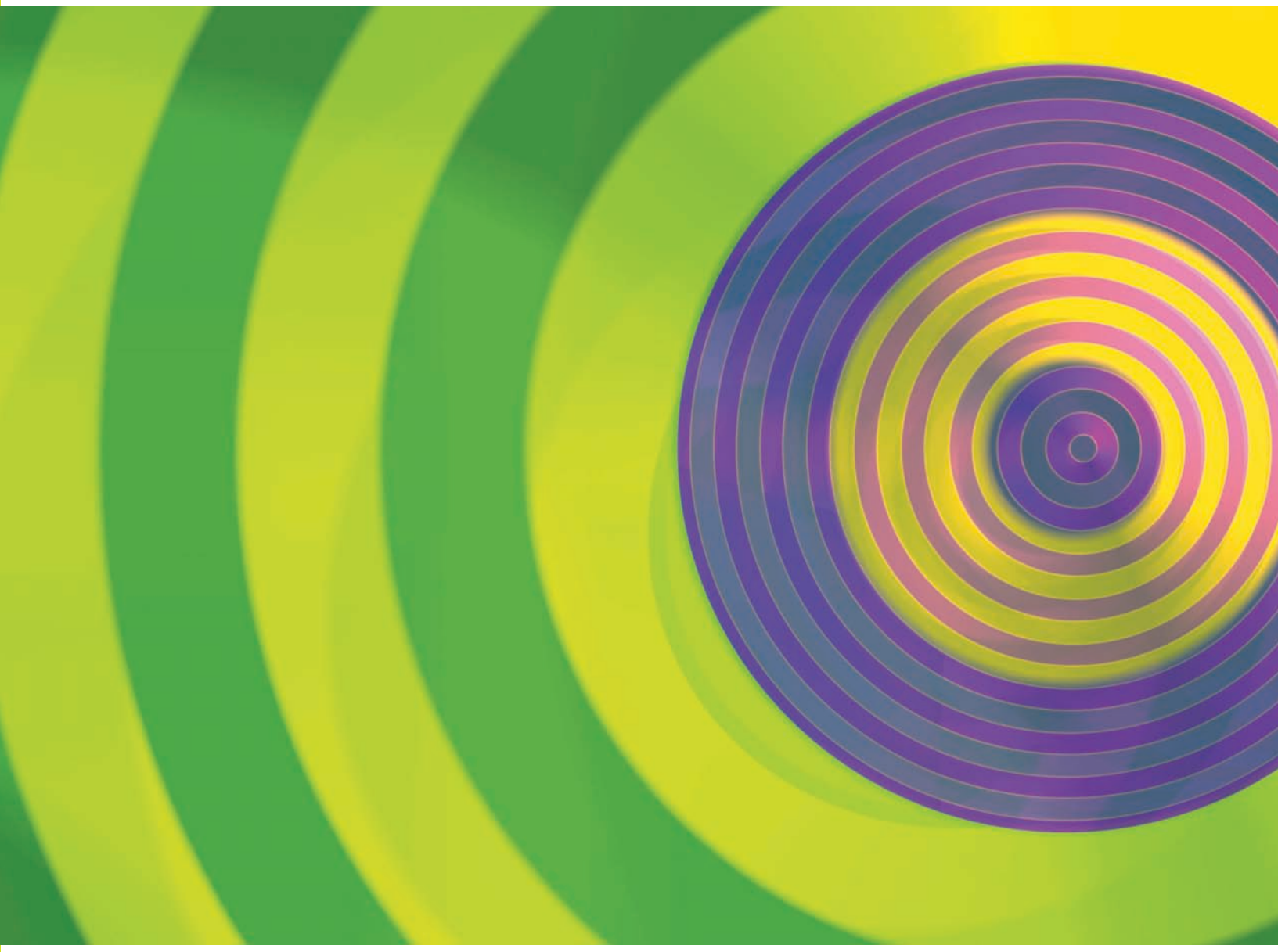
Pearson New International Edition



Object Oriented Systems Analysis and Design

Noushin Ashrafi Hessam Ashrafi

First Edition

ALWAYS LEARNING™

PEARSON®

Pearson New International Edition

Object Oriented Systems Analysis
and Design
Noushin Ashrafi Hessam Ashrafi
First Edition

PEARSON®

8. Key Concepts

Business Rule. A guideline, procedure, policy, law, standard, or constraint under which an enterprise operates. Business rules are independent of information technology and its applications.

Business Rules Dictionary. A document that defines and categorizes business rules. The dictionary is updated through questionnaires that allow stakeholders to verify individual rules.

Domain. A territory with shared concepts and rules. A ***business domain*** is an organized, goal-oriented domain. The term may be applied to an entire enterprise, its sub-domains, or the context in which the enterprise or the product or the service operates.

Domain, Derived. A domain that results from discovering a set of common concepts across various domains or sub-domains.

Domain Analysis. An iterative activity to discover and define domain concepts within the scope of an enterprise and/or its sub-domains.

Domain Catalog. A directory of documents (or references) that support domain concepts and rules. The documents may include business policies, guidelines, manuals, interviews, etc.

Domain Concept. An abstraction of a process, function, object, or role within a domain.

Domain Definition. Identifying the scope and the boundaries of a domain or sub-domain.

Domain Dictionary. An ordered repository for domain concepts, concept definitions, and concept types.

Domain Expert. A person with a high degree of knowledge about a domain or sub-domain.

Domain Scope. ❶ Definition of boundaries that separate shared activities, rules, and concepts within a domain from those outside. ❷ A document that establishes domain boundaries by describing its identifying features.

Problem Space. The context from which the problem originates and in which the solution must operate. Also called ***problem domain*** (usually shortened to just "domain").

Product. A product (or service) is the realization of features specified by requirements. A product is also the solution as answer to the problem that the requirements seek to solve or the opportunity of which they hope to take advantage.

Solution. ❶ Answer to a problem. (See **Product**.) ❷ Method or process to build a product or service that aims to solve a problem.

Solution Space. Where concrete decisions about the design and the implementation of the solution as a product or service are made. Solution space defines the scope of *how* a product is made, not *what* features it must have. Also called ***solution domain*** or ***system domain***.

Sub-Domain. A domain within a larger domain. (See **Domain**.)

Subsystem, Conceptual. A unit within an information system that correlates to a business domain; integrates with its processes, rules, and its goals; and is used by the people in that domain (actors).

9. Confusing Terms

Domain. The term "domain" has more than one application, even in the context of information systems. For example, in database design, it determines the range of acceptable values. In networking, it is a collection of computers that share the same management and security database. And more. Even though all meanings are related, they should not be confused.

Requirements. Strictly speaking, requirements should only identify the features that a product must have. In practice, however, descriptions of the problem, guidelines for solving the problem, and product specifications are also thrown into the mix. While we have emphasized that you must discover the "true" requirements, do not disregard the non-pure elements. Also, "requirements" often are used in a general sense: what is necessary to do something or a prerequisite. We must rely on the context to distinguish between the two meanings.

Solution. The method to solve a problem and the answer to a problem are both called "solution." Sometimes the distinction is not important. Most of the time the context will guide us. When the context does *not* help, we must make the distinction clear.

10. Topic Discussions

Emergence of Domain Analysis

The idea of domain analysis was introduced in the 1980s in response to the "software crisis." But it has not been long since it has shown signs of becoming popular and mainstream. The perception of "crisis," of course, has not changed. What has changed is that **[Neighbors 1981]**

- first, business is learning, very slowly and in its own good time, that reinventing the wheel every time is expensive and wasteful; and
- second, the building blocks of an enterprise-oriented approach to software development are falling into place, in terms of both technology (powerful computers, storage, networks, etc.) and methodology (object-oriented tools, languages, and modeling).

Still, you can read many chapters and papers on domain analysis and not recognize that the authors are talking about the same concept (like many other subjects in system analysis and design). Evidently, a lot more needs to be done and, fortunately, is being done.

Domain as a Concept

Domain is a *concept*, which means it is a perception of reality, not the reality itself—even though the perception might come very close to reality. Nevertheless, domains do display an inherent logical coherence—once we discover the rules or, perhaps, once we select and organize them based on our preferences and our aims. Animal Kingdom is a recognizable domain: We do distinguish between the rules that apply to a bear and those that bear upon a rose which belongs to the Plant Kingdom.

Domains, however, are *not* mutually exclusive. Within one domain, we can distinguish other domains and sub-domains: birds and mammals within the Animal Kingdom, for example. They may also overlap: We can take elements from one domain, combine it with elements from another one, and come up with a new domain that is equally valid. Take a fly from the insects domain, a finch from the birds, and a bat from the mammals and we are likely to "discover" a domain in which the rules of aerodynamics apply because they all fly. The new domain has an internal coherence and an objective reality, but we arrive at it by an act of selection.

☞ In building an information system, the point that domain is a perception is important to remember since we have a wide latitude in how we interpret the reality of a business.

11. Review Questions

1. What is the difference between domain definition and domain dictionary? Give an example for each.
2. Define business roles and give three examples for a bank.
3. Write requirements for the following products and services:
   a. Computer.
   b. Airplane.
   c. Food Processor.
   d. Online Banking
   e. Life Insurance.
   f. Party Planning.
4. What are the characteristics of business domains? Explain them in the context of a specific business.
5. Define "derived" domain. Could an information system be a derived domain? Explain.
6. Distinguish between logical and physical models.
   a. Which one comes first?
   b. Where do conceptual models fit?
7. Create a domain dictionary for an ATM system.
8. Identify three business rules for an ATM system.
9. Identify three objects for an ATM system.
10. Identify three functions for an ATM system.

12. Exercises

The following exercises apply to each of the four narratives introduced in Gathering Requirements *The* Sportz *Magazine*, *The Car Dealership*, *The Pizza Shop*, and *The Real Estate Agency*.

❶ What are the business concepts in the scenario?
❷ Categorize the business concepts into processes, functions, roles, objects, and business roles.
❸ What would you name the system and its subsystems?
❹ Write domain definitions and, from there, proceed to establish the domain scope for the system and each subsystem.
❺ Create a domain dictionary for the proposed information system.

13. Research Projects & Teamwork

❶ Refer to the narrative for Walden Medical College in Gathering Requirements. All team members should read this chapter thoroughly before meeting to discuss how the following questions should be answered.

- Create a domain dictionary for the online registration subsystem. Identify concepts, their types, and descriptions. Use Table 4 as a guide.
- Identify business rules for the subsystem and create a "rules dictionary," complete with definition, types, and sources. Use Table 5 as a guide.
- Create a business rules questionnaire similar to Table 6. Each definition should be formulated in a way that can be verified by a simple **`True`** or **`False`**.
- Select a group member as group leader and let the leader prepare a report addressing the questions above with an appendix that indicates the contribution of each member as the percentage of total work.

❷ Refer to the *automated research system* (the "reference engine") in Gathering Requirements and select one team member as the leader for this project. The team leader should decide how the team proceeds to answer the following questions. (One way is that each team member answers the two questions individually and hands them over to the leader, who then combines them to arrive at the most comprehensive answers and submits them to the instructor.) The team leader should also evaluate each member's contribution and write a short memo describing how the team worked together.

- Create a domain scope for the reference engine. Make sure to define domain boundaries and functions clearly. Use Table 3 as a guide.
- Create a domain dictionary for the application. identify concepts and their types and provide descriptions. Use Table 4 as a guide.

❸ Refer to the *online shopping system* in Gathering Requirements and answer the following questions. (Follow the same process described for the previous project, but with a different project leader.)

- Create a domain scope for the online shopping system. Make sure to define domain boundaries and functions thoroughly.
- Identify business rules for the system and create a rules dictionary complete with definitions, types, and sources. Use Table 5 as a guide.
- Create a business rules questionnaire similar to Table 6. Each definition should be formulated in a way that can be verified by a simple **`True`** or **`False`**.

❹ Alberto is a new graduate and has been hired by a doctor to automate his patient scheduling system. He is your best friend and has no clue where to start. Use all you have learned from this chapter and help him by conducting a domain analysis. (If you do not have access to a real doctor's office, there is always the Web.) Identify domain boundaries, functions, tasks in each function, and business rules.

14. Suggested Readings

Despite the growing acceptance of domain analysis, the volume of writings on the subject in the context of object-oriented software development is still rather limited. Many valuable contributions are published as academic papers or articles, not as books. The good news is that more than a few are available on university Web sites.

Some of the books cited in our References discuss domains (if not domain analysis) at some length. But for a basic and clear introduction to domain analysis, see "Domain Analysis: An Introduction" by **Rubén Prieto-Diaz** in *Software Engineering Notes, 15–2* (April 1990, http://www.cs.jmu.edu).

If you are interested in the origins of the idea and its relationship to components, read *Software Construction Using Components* by **J. Neighbors** (Ph.D. Thesis, Department of Information and Computer Science, University of California, 1981).

For an advanced discussion of problems within the context of software development, consult *Problem Frames: Analyzing and Structuring Software* by **Michael Jackson** (Addison-Wesley, 20001).

Our discussion of business rules barely scratches the surface. There are a number of comprehensive Web sites dedicated to business rules, including:

- The Business Rules Community (http://www.brcommunity.com).
- The Business Rules Group (http://www.businessrulesgroup.org).

Behavioral Modeling I

Use Cases: The Basics

1. Overview

Use case modeling represents the behavior of a system. A use case details the interaction of entities outside a system, called actors, with the system to achieve a specific goal by following a set of steps called a scenario.

- ☑ Inception
- ☑ Requirements Discovery
- ☐ Development
  - ☐ Analysis
    - ☑ **Domain Analysis**
    - ☐ Behavioral Modeling
      - ☐ **Use Cases: The Basics** ⬅
      - ☐ Developing Use Cases
    - ☐ Structural Modeling
    - ☐ Dynamic Modeling
  - ☐ Design
  - ☐ Implementation

Chapter Topics

- What use case modeling is and is not.
- The four components of a use case.
- The basic elements of use case diagram.

- Various flows in the narrative of a use case.
- How to transform concepts from domain analysis into use cases.
- Identifying prominent actors.
- Identifying major use cases.
- The context diagram.

Introduction

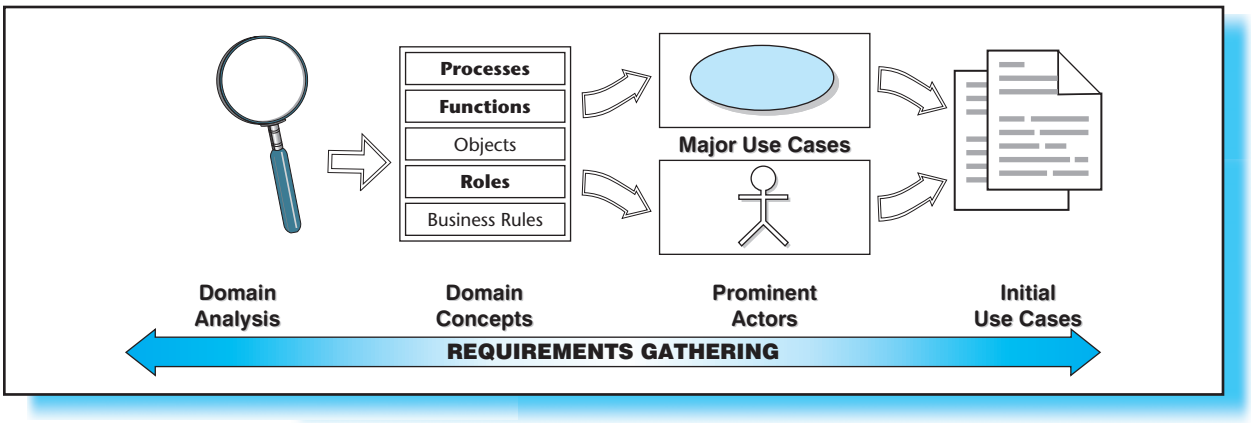
In the chapter on domain definition and domain analysis, we argued that to build a solution we must first understand the problem and, second, distinguish between the method for solving a problem and the answer to the problem. We also identified two spaces: ***problem space*** where the problems arise and solutions must operate in, and ***solution space*** where issues relating to the product itself are encountered.

The task of ***domain analysis*** is to put requirements back into context and discover and define ***business concepts*** that are shared between the problem space and the solution space. ***Domain definition*** is needed to establish the ***domain scope*** for analysis and provide the conceptual framework for subsystems.

Domain analysis, however, does not produce a model but the concepts that we must use for modeling. The main gateway between domain analysis and the modeling of the information system—be it dynamic or structural, conceptual or logical—is **use case** modeling (Figure 1), which represents a ***behavioral model*** of the system.

Use case modeling is the foremost tool in taming the complexity of developing systems. It is also the most sensitive stage in gathering requirements. Gathering requirements, of course, does not start with use cases: The project starts with the initial requirements, and domain analysis must carry a heavy load of it. Nor does it end with use cases: We have a mass of detail to work out before the structure of an information system really takes shape.

Figure 1 **Discovery of Use Cases: Transforming Domain Concepts into Behavioral Models**



Use case are discovered by analyzing and expanding domain concepts, discovered through domain analysis.

The significance of use case modeling is that it provides the ***framework*** for the most important building blocks of modeling and beyond. Use cases are the indispensable guideposts from one end of system development to the other: from gathering requirements and communicating upstream with stakeholders to exchanging information downstream with designers and programmers to testing the product and training the users.

As we shall show, the essence of a use case is simple. However, use case modeling extends beyond use cases themselves. It is not limited to the components of the use cases but also originates a diverse set of documents that shapes both the means and the goal of system development.

The formal definition of use case is straightforward. Grasping its significance and depth, however, takes some effort. In this chapter we first present a ***definition*** of use case modeling and the ***four components*** of a use case. Next, we draw a map for arriving at ***major use cases*** from domain concepts. Later, we discuss how the findings of this ***initial stage*** of use case modeling should be organized and presented.

2. Introducing Use Cases

A use case is a compound entity. Understanding it requires a good understanding of its components and how they affect each other. It also requires a clear understanding of what use case modeling is *not.* Without this negative definition, use cases are liable to miss their mission and turn into a grab bag of irrelevant and misguiding details.

What Use Case Modeling Is

☞ A ***unit*** is "a group regarded as a distinct entity within a larger group." [American Heritage 1996]

> Use cases model the behavior of a system.

A use case is a unit of system behavior. This definition needs a description to clarify it.

☝ By ***behavior*** we mean exactly that—and no more: A use case describes *what* a system does as viewed from outside, not *how* it does it inside it own boundaries.

> A use case details the interaction of an actor with a system to accomplish a goal of value to the actor.

The description, in turn, needs a story to illustrate what a use case actually is.

A customer enters the supermarket. The customer takes a shopping cart or basket and strolls through the supermarket. The customer selects items from the shelves and puts them in the shopping cart or the basket. When finished, the customer brings the items to the cash register. The cashier calculates the total price of the merchandise. The customer pays for the merchandise. The cashier bags the items, issues a receipt to the customer, and, if necessary, returns the change. The customer picks up the bags and leaves the supermarket.