and do some of them in parallel. The stages listed below were originally proposed by Rumbaugh *et al.*, (1991) and have been widely used with minor adaptations ever since:

- Identify the objects and derive classes
- Identify attributes
- Identify relationships between the classes
- Write a data dictionary to support the class diagram
- Identify class responsibilities using CRC cards
- Separate responsibilities into operations and attributes
- Write process specifications to describe the operations.

In this chapter we will concentrate on the first four stages in this list; responsibilities and operations are covered in Chapter 6.

Identify the objects and derive classes

Class diagrams are a very useful way of modelling the structure of the objects in a system and the links between them. However, when we are trying to design a new software application, we don't know what the classes are in advance and we will often start by looking for objects before making decisions about what their class will be.

Objects come in different categories; being aware of the categories gives us an idea of the sorts of things to look for. Objects at the analysis stage will be things that have meaning in the application domain. They can be:

- People (such as customers, employees, students, librarians)
- Organizations (such as companies, universities, libraries)
- Physical things (such as books, bikes, products)
- Conceptual things (such as book loans and returns, customer orders, seating plans).

Various techniques can be used for object identification, none of them are foolproof, none can be guaranteed to produce a definitive list of objects and classes; they are just guidelines that might help. A good starting point is to look at the documentation about the system produced so far and to search for nouns in any description of the system requirements, preferably one that is complete and concise. It is a good idea to discuss your choice of objects with users who understand the application domain, and with colleagues who have experience in object analysis.