the system is the users' view: it specifies what the user wants the system to do; the other 4 views describe how to achieve this. The use case view describes the external behaviour of the system and is captured in the use case model, discussed in Chapter 3. In the early stages of development, the software architecture is driven by the use cases; we model the system's software in terms of collaborations of the classes[2] required by each use case. *The design view* (sometimes called the logical view) describes the logical structures required to provide the functionality specified in the use case view. The design view describes the classes (including attributes and operations) of the system and their interactions. This view is captured in the class diagram, discussed in Chapter 5 and the interaction diagrams, discussed in Chapters 6 and 10. *The process view* is concerned with describing concurrency in the system; concurrency is beyond the scope of this book, but we briefly discuss how sequence diagrams can be used to achieve it. *The implementation view* describes the physical software components of the system, such as executable files, class libraries and databases. This view of the system can be modelled using component diagrams (see Chapter 9). *The deployment view* of the system describes the hardware components of the system such as PCs, mainframes, printers and the way they are connected. This view can also be used to show where software components are physically installed on the hardware elements. Deployment diagrams describe this view of the system. Not all of these views will be required for every system. For instance, if your system does not use concurrency, you will not need a process view. If your system runs on a single machine, you will not need the deployment view or the component view, as all of the software components will be installed on one machine.

The 4 + 1 view gives us five different ways of viewing a system and is supported in UML by modelling techniques to capture each view. Just as a photograph, family tree and description give us different views of a person, the UML views show us a system from different points of view. Each one offers a different perspective, no one gives us the whole picture. To gain complete understanding of the system all of the views must be considered.

UML and CASE tools. One of the advantages of a standardized language for producing diagrams during system development is that a number of CASE tools have been developed to provide automated support for developers.

2. *A collaboration of classes is the set of classes required to provide the functionality of a specific use case (see Chapter 3).*