Unlike a photograph, the family tree in Figure 1.4(a) tells us nothing about Jane's appearance, but it does provide information about her family. We can see that she has four brothers and is the youngest child; we also discover the first names of Jane's parents and siblings. The family tree is a graphical, diagrammatic model. Figure 1.4(b), on the other hand, is a textual model; it tells us some details about Jane, her birth date, address, school and which class she is in. Each model (the photograph, the family tree and the piece of text) tells us something about Jane, but we can only get all the information by looking at all the models (in fact we would need many more detailed models to get a complete picture of her).

The characteristic of a model to provide some but not all the information about the person or thing being modelled is known as *abstraction*. We can say that each of the models in Figure 1.4 is an abstraction of the real-life Jane, focusing on certain aspects of her and ignoring other details. In the same way, each of the modelling techniques in the Unified Modelling Language provides a particular view of the system as it develops; each UML model is an abstraction of the complete system.

Abstraction is a very important tool in modelling any sort of software system because typically the problems developers have to deal with are much too complex for one developer to hold all the details in his head at once. Abstraction provides a means of concentrating on only those aspects of the system that are currently of interest, and putting other details to the side for the time being.

Another, equally important tool for modelling software systems is *decomposition*. This is the breaking down of a large, complex problem or system into successively smaller parts, until each part is a 'brain-size' chunk and can be worked on as an independent unit. Traditionally software systems used to be decomposed according to their functions – the tasks that the system had to carry out. As we shall see later, however, object-oriented systems are decomposed according to the data that they have to store, access and manipulate.

Initially, the models that are constructed using the techniques provided by the UML help the developer to impose a coherent structure on the information gathered from clients and users. One of the principal uses of the models is as a basis for discussions, since talking about the system as represented helps to identify gaps and inconsistencies in the information. As the project progresses, the original models are enhanced with details relating to design and implementation issues, so that eventually they become a blueprint for the development of the system.