included the extra details as part of the original definition of Customer as follows:

Customer = customerID + title + [initial | firstName] + surname +
(houseName) + houseNumber + street + town +
county + postcode + {areacode + number}$_2$ + (email)

However, this is clumsy and difficult to read; the structured approach that we showed above is much clearer.

There is still one more refinement that we may wish to make to the definition. The expression [initial | firstName] allows the system to record a single initial or first name for a customer, but not both. However, some businesses may wish to record customer names in different ways, for example: Lee Jones, L. Jones, Lee M. Jones, L. M. Jones, etc. We can cater for this in our definition by changing [initial | firstName] to {[initial | firstName]}. This allows a choice between either an initial or a name any number of times, and will allow all the variations that we need.

It is often useful to write data dictionary definitions of documents provided by clients during the early phases of development. This allows the developer to separate the information held in the document from the way it is presented. Figure 5.16 shows a receipt form as used in the current Wheels system.

When documenting something like this, it is best to divide it up into separate sections and then describe each of these in turn. This produces a definition that is clearly structured and easy to follow. For example, we can divide the Wheels receipt into the sections: title, customer details, hire details, and total as shown below.

Receipt = title + customerDetails + {hireDetails} *a customer may
hire more than one bike at a time* + total
title = "Wheels Bikes Receipt for Hire" + receiptDate
customerDetails = customerName + customerAddress
hireDetails = bike# + bikeDescription + ratePerDay + no.OfDays
+ hireCost + deposit + totalCost
total = amountDue + "Paid with thanks"

We can decompose to further levels as needed, for example we could add:

bikeDescription = make + model + type + size

In general, the closer we get to implementation, the more details we need in the data dictionary.