CHAPTER 15

Hints on creating specifications

15.1 Introduction

The process of creating formal specifications needs more than fluency with the mathematical tools provided by the Z notation. It requires imagination and investigation and the ability to revise one's work when difficulties arise and when a new approach offers itself. It has been said that the most important aid to formal specification is a large waste-paper basket!

This chapter offers some hints on how to go about creating formal specifications.

15.2 Types

Find out what the sets (types) in the system to be specified are. Keep these sets general. The style which uses 'the set of all ...' allows you to consider a subset and to add new elements from the 'set of all' to the subset and to remove elements from the subset.

For example, if you declare

[STUDENT] the set of students at this university

you cannot enrol any new students or allow students to leave, since the type is fixed.

The following is far better:

[PERSON] the set of all persons

students: $\mathbb{P}$PERSON

Be sure that your types are truly *atomic*. For example, don't declare a type *CLASS* meaning a *group of* students. Instead, use the type *PERSON* as above and make *class* a subset.

15.3 Relationships

Next consider the relationships between the types. By discovering how many values of what type are related to how many of the other type, find out if any of the *relations* are *functions*. If so, are the functions *total* or *partial*, or *injective*? Are there any relationships between the relations and functions you have found? For example, must they have the same domains or must the domain of one contain the range of another?

If there is a natural order to some values, then think of using a sequence; otherwise it is easier to work with sets.

Remember that it is necessary to state the obvious!

15.4 The state

This investigation of relationships will lead to a schema which represents the *state*. You will also have discovered some *invariants*, which are included in the state schema.

15.5 Initialisation operation

You will need to provide an initialisation operation that sets up an initial state. This *initial* state should be as simple as possible. Often it gives empty sets and relationships as the starting point. It is important that the initial state should not violate any invariant of the system. Ideally, this will be *formally proved*. With a simple initial state it is usually easy to see that the invariants are not violated.

15.6 Operations

Start by considering the behaviour of each operation only when it is given sensible values. Include a check that the values are reasonable (the *pre-condition*), but do not deal with errors at this stage. At a later stage define a separate schema to deal with errors by returning a reply value.

Don't forget to state explicitly what does *not* change as a result of an operation.

15.7 Enquiry operations

Enquiry operations do not *change* the state, and thus cannot violate an invariant. Use a *xi* (Ξ) schema where possible.

15.8 In case of severe difficulty

If the specification is getting too hard, try viewing it in a more *abstract* way. Hide some of the details for now and try to take a broader view. Put the detail back in later when you have a better understanding of what you are trying to do.