Abstract. VDM and B are two "model-oriented" formal methods. Each gives a notation for the specification of systems as state machines in terms of a set of states with operations defined as relations on that set. Each has a notion of refinement of data and operations based on the principles of reduction of non-determinism and increase in definedness. This paper makes a comparison of the two notations through an example of a communications protocol previously formalised in [BA91]. Two abstractions and two reifications of the original specification are given. Particular attention is paid to three areas where the notations differ: the use of postconditions that assume the invariant as opposed to postconditions that enforce it; the explicit "framing" of operations as opposed to the "minimal frame" approach; and the use of relational postconditions as opposed to generalised substitutions.

1 Introduction

In [BA92], Bruns and Anderson describe a communications protocol in CCS with value-passing. A data model for the values is given which is, in effect, a model of the state of the device. This model is defined in terms of the usual data constructors of model-oriented specification, but without the use of invariants.

The part of the protocol described is a mechanism for manipulating a series of flags that indicate the status of some shared-memory buffers. These flags are used to ensure that there is no "data-tearing" as multiple processors simultaneously read and write to the buffers. For the operations that update these flags, semaphores are used to ensure that each operation has uninterrupted access to the flags. Thus this part of the behaviour can be described as a purely sequential system.

This paper considers some alternative data-models for the specification (and reification) of these status flags. In particular, attention is paid to the use of invariants in the data model and frames of reference in the operation definitions, neither of which are available in the data modelling language of [BA92]. It is argued that these features can play a key role in describing the system in a "natural" fashion and can thus help to deepen our understanding of the model.

VDM [Jones90] and B [Abrial92b] are used for the analysis, and particular attention is paid to some areas where the notations differ: the use of postconditions that assume the invariant as opposed to postconditions that enforce it; the
explicit “framing” of operations as opposed to the “minimal frame” approach; and the use of relational postconditions as opposed to generalised substitutions. In this small example, there is little scope for the effective use of structuring of specifications that is one of the major features of the B method. Familiarity with the basic concepts and notation of VDM and B is assumed.

The remainder of this first section is an informal description of the application and desired protocol. The second through fifth sections present the development in VDM. Section two presents a formal specification of the system at a level of abstraction similar to the “abstract” description of [BA92]. Motivated by an analysis of the invariant of that specification, section three describes two further abstractions that can be made. Section four provides an alternative model of the system that makes it possible to write more useful framing information about the operations, the fifth section extends this model to the “improved” protocol of [BA92]. The sixth section considers the development again using B, presenting those elements of the development that highlight the differences in the notations. The last section is a discussion of some of the points arising from the example and their treatment in the two notations.

1.1 The Multiprocessor Shared-Memory Information Exchange

The Multiprocessor Shared-Memory Information Exchange (MSMIE), is a protocol that addresses intra-subsystem communications with “several features which make it ideally suited to inter-processor communications in distributed, microprocessor based nuclear safety systems” [Santoline89]. It has been used in the embedded software of Westinghouse nuclear systems designs.

The protocol uses multiple buffering to ensure that no “data-tearing” occurs, that is, it ensures that data is never overwritten by one process whilst it is being read by another. One important requirement is that neither writing nor reading processes should have to wait for a buffer to become available; another is that “recent” information should be passed, via the buffers, from writers to readers. In the simplification considered in [BA92] it is assumed that information is being passed from a single writing “slave” processor, to several reading “master” processors.

The information exchange is realised by a system with three buffers. Very roughly, at any time, one buffer is available for writing, one for reading and the third is either in between a write and a read and hence contains the most recently written information, or between a read and a write and so is idle.

The status of each buffer is recorded by a flag which can take one of four values:

s - “assigned to slave.” This buffer is reserved for writing, it may actually be being written at the moment or just marked as available for writing.

n - “newest.” This buffer has just been written and contains the latest information. It is not being read at the moment.