context of model-oriented specification. With the other approaches to formal specification the technical details of refinement are different from that of model-oriented specification, but the spirit is the same – verifying that we are adding detail, or otherwise enriching specifications, in a manner which is consistent with the initial specification.

Our brief discussion has also focused largely on the verification aspects of refinement, and not on the guidelines for proceeding from a high-level to a low-level specification. Typically these guidelines will (or should) cover issues of functional decomposition, and also consider non-functional properties of systems. That is, the guidelines should recognise that non-functional issues such as performance, reliability and so on, can **drive** the refinement process. Unfortunately current refinement approaches do not deal adequately with such issues so, for example, there are no refinement rules which deal adequately with fault tolerance – an approach would need to show that the fault models plus fault recovery mechanisms at one level 'satisfied' the fault models at the next higher level. This remains an area of research.

A.4 Summary and comparison of approaches

There are a wide variety of types of formal methods, each with different characteristics, which means that generalisations about formal methods may be more misleading than helpful. It is also rather difficult to appreciate what the methods are like in use from simple definitions and descriptions – by way of analogy, consider how difficult it is to appreciate the utility of a programming language without trying it out on a few problems. This is why we have taken the trouble to give fairly extensive examples of two rather different types of formal method. We are now in a position to make some comparisons, although we steer clear of value judgements regarding utility.

First, we can now see clearly that the two methods enable us to do quite different things. Z enabled us to give quite detailed specifications of the required behaviour of the actions to be carried out in the system, but was rather poor at modelling communication and has no way of representing concurrency. In contrast, MAL is much clearer about system structure – including potential for parallel execution – and communication, although it is relatively weak at defining functionality. MAL allows us to make statements about timing behaviour, whereas Z does not.

Some of the above differences are partly a reflection of the way in which we have used the notations. For example it is possible to specify timing in Z,[10] but we believe we have accurately characterised the 'natural' way to use the core specification languages in each case. Thus we must conclude that different methods have quite different expressive powers.

Second, we believe that it is quite difficult to use the techniques outside their natural domains. This does not mean to say it is impossible; as we indicated above, it is possible to extend the techniques to deal with additional properties of systems but it is not entirely straightforward – for example adding a deontic component to Z would be quite difficult, especially when it came to defining the semantics for the extended notation. However, since the methods illustrate different facets of systems they can be used together – assuming we can map adequately between the notations. Thus we believe that it is both possible and beneficial to use an eclectic approach to specification, although this is rarely, if ever, done in practice.

Third, the mathematics, although valuable for its precision, does not stand on its own. In the example it was essential to use prose to define what it was that the specifications were meant to relate to – 'in the real world'. It is always necessary to support formal specifications with prose and, without this, we have no way of knowing what the specifications mean. More technically we know what they mean in terms of the underlying logic, but we do not know what they mean in relation to the systems we hope to build. This is a general property of formal approaches, not just a characteristic of our examples, but one which we hope is adequately borne out by the examples.

Fourth, there is considerable difference in the conciseness or verbosity of the notations. Again, this is partly an effect of the examples chosen and the way the problems have been addressed. This is important as conciseness influences intelligibility, although there is not a simple relation. Extremely terse and extremely verbose notations may be equally hard to read and, ideally, we require concise notations so we do not have much to read, but which are still as easy to read as ordinary english prose. This is, of course, a difficult compromise to achieve – and we shall leave the reader to draw his own conclusions about which, if any, of the two notations used above satisfy this requirement.

Finally, it is important to stress the point that the methods are genuinely different in their capabilities and any generalisations about formal methods (other than this one!) may be quite misleading and inappropriate to some particular class of method.

Announcement

1–4 JUNE 1993

Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-93), The City Chambers, Edinburgh, Scotland

This is sponsored by the International Society of Applied Intelligence in cooperation with major international organisations and universities, including ACM/SIGART, AAAI, IEEE Computer Society, ECCAI, CSCSAI, IAKE, INNS, JSAI, and Southwest Texas State University. For further information, contact Dr Moonis Ali, General Chair, IEA/AIE-93, Department of Computer Science, Southwest Texas State University, San Marcos, TX 78666-4616, U.S.A. Telephone: (512) 245-3409; Fax (512) 245-3804; E-mail: MA04@SWTEXAS.BITNET.