ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

**D. J. Duke , University of York, UK and A.S. Evans, University of Bradford,
UK (Eds)**

2nd BCS-FACS Northern Formal Methods Workshop

Proceedings of the 2nd BCS-FACS Northern Formal Methods
Workshop, Ilkley, 14-15 July 1997

Formal Methods and Standards - An Idiosyncratic View (Invited Paper)

D.A. Duce

Springer

©Copyright in this paper belongs to the author(s)

Published in collaboration with the
British Computer Society

Formal Methods and Standards - An Idiosyncratic View

David A. Duce

Rutherford Appleton Laboratory
Chilton, Didcot, United Kingdom

Abstract

Many of the reported experiences in the industrial use of formal methods concern the development of products or product families, where the utility of the method is linked to direct savings in development costs or improved assurance of quality. However, one other area in which formal description techniques make a valuable contribution is in the development and documentation of International Standards, where the cost of using formal methods can be paid off both through increased quality of products that implement a given standard, and through the improved inter-operability of different implementations that comes from having a precise definition of the expected behaviour of a conforming implementation. The process of standardization within ISO/IEC is complex, and affords the opportunity to use formal methods at different stages and in different ways. This paper illustrates how formal methods have been used in the development of two standards, the computer graphics standard, GKS, and the multi-media standard, PREMO. Formal methods have been used at different points during their development. This paper concludes with an appraisal of the work done and some thoughts about future directions.

1 Introduction

This paper is based on the author's experiences over a period of some 12 years, applying formal methods (more precisely formal description techniques) in the development of International Standards for computer graphics and multi-media systems.

The work has involved a large number of collaborators, including David Arnold, Ken Brodlie, Ljiljana Damnjanovic, David Duke, Giorgio Faconti, Elizabeth Fielding, Paul ten Hagen, Ivan Herman, Bob Hopgood, Robert van Liere, Philip Nehlig, Graham Reynolds, and Fabio Paterno. Financial support for the work from the Engineering and Physical Sciences Research Council and the European Union's Human Capital and Mobility Programme (through the ERCIM Computer Graphics Network) is gratefully acknowledged.

2 Standardization

The underlying motivation for standardization in Information Technology is 'to protect the investment in increasingly expensive IT by extending the useful life of systems' [1]. Extension is achieved by building into the design of a system as much independence from particular technology and a particular supplier of that technology as possible. Portability across hardware platforms, input and output devices and particular programming languages were major driving forces and design criteria in the development of the first International Standard for computer graphics programming, the Graphical Kernel System, GKS.

Standards come in many different flavours, for example, de facto standards, de jure standards, local, national and international standards. The focus of this talk is upon international standards, and in particular, international standards in information technology promulgated by ISO/IEC (the International Organization for Standardization/ International Electrotechnical Commission).

IEC, the International Electrotechnical Commission was created in 1906. ISO, the International Organization for Standardization, grew out of the International Federation of the National Standardizing Associations, which was created in 1926 but ceased to operate in 1942. ISO came into being on 23 February 1947, with a purpose 'to facilitate the international coordination and unification of industrial standards'. The primary motivation for international standards

is to eliminate ‘technical barriers to trade’; the view is that world-wide standards help rationalize the international trading process. The author well remembers meeting a man who was involved in the standardization of beds. His industry was certainly hampered by the fact that standard bed sizes in the UK are different to the USA, are different to Scandinavia, etc. UK suppliers of mattresses find that standard sizes do not fit bed frames made in Norway, for example!

IEC covers the field of electrical and electronic engineering; all other subject areas are attributed to ISO. A joint committee, called JTC1, has been created by ISO and IEC to look after standardization activities in the information technology area. For more background information, see the ISO web site [21].

Standards are defined by ISO [21] in the following way:

Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose.

Standardization is defined as:

The activity of establishing, with regard to actual or potential problems, provisions for common and repeated use, aimed at the achievement of the optimum degree of order in a given context.

Different nations tend to have different views of what a standard is and what standardization is for. In the graphics area, the view in the UK and Europe tends to be that standards define a product. Implementation dependencies should be reduced to an absolute minimum and there should be rigorous testing of products that claim to adhere to standards to demonstrate that they do in fact satisfy the provisions of the standard. Competitive edge in this view is gained by having a product that is sleeker and more efficient than those of the competition. The standard is a description of an artefact that is to be built precisely according to the provisions of the standard.

In the USA on the other hand (at least in the graphics area), the view has tended more to the idea that standards provide a core set of functionality or ideas and competitive edge is gained by extensions and implementation dependent features. Limiting implementation dependencies is seen as stifling innovation. The standard is not defining a product that is to be built precisely according to the provisions of the standard.

These differing views continue to cause interesting exchanges in standards meetings!

The process by which standards are made continues to evolve. There has been much criticism of the length of time it takes to develop international standards (five years is quick!) and ISO/IEC are addressing this by reviewing and modifying the creation process. Currently there are six stages to the process [21]:

Proposal stage A new work item is proposed and voted upon by the appropriate Technical Committee.

Preparatory stage A working group prepares the ‘best’ technical solution, initially as a working draft which is refined through meetings and commenting periods until a stable technical document is arrived at. This is known as a Committee Draft (CD).

Committee stage The CD is voted upon by the Subject Committee responsible for the work, to build a consensus amongst the nations participating in the project on the technical material. When there is a consensus, the document progresses to Draft International Standard (DIS).

Enquiry stage The DIS is circulated to all member bodies in the Technical Committee for a 5 month vote. Success requires a 2/3 majority of members participating in the work and not more than 1/4 of the total vote cast are negative. If the document succeeds, it becomes a Final Draft International Standard (FDIS). If it fails, the rules of snakes and ladders apply!

Approval stage The FDIS is circulated to all member bodies for a 2 month vote (approval rules as for DIS). If successful, the final text is prepared.

Publication stage The final text is published by the ISO Central Secretariat.

The cycle of ballot, comment, revise, re-ballot, is a long one, but it does ensure that technical material is given thorough review. Published standards are reviewed at least every 5 years, at which stage the standard may be confirmed, revised, or withdrawn.

There are three principles that underlie ISO/IEC standardization (adapted from [21]):

Consensus views of manufacturers, vendors and users, consumer groups, testing laboratories, governments, engineering professions and research organizations are all taken into account. All may be represented in national standards-making bodies (such as BSI) and all have the opportunity to participate in the work.

Industry-wide International Standards are meant to be global solutions to satisfy industries and customers world wide.

Voluntary standardization is market driven and therefore based on voluntary involvement of all interests in the market place.

The voluntary nature of standards work can prove problematical, especially with regard to timescales. Standards documents in the IT area are frequently large and complex. Maintaining these in one's spare time can be taxing in the extreme!

ISO Central Secretariat derives some 30 percent of its revenue from the sales of standards and other publications. In the IT area there is strong pressure to make standards more readily available, for example through the web. However the threat of a potentially serious loss of revenue to ISO if standards were made freely available is resulting in slow progress in this direction.

Some keywords may be extracted from the above discussion:

- Quality
- Technical specifications, precise criteria
- Documenting agreements
- Common and repeated use
- Fitness for purpose
- Optimum degree of order.

These keywords suggest some ways in which formal methods may be used in the development of standards and some reasons why we might look to formal methods during development and in presentation. Some (for example, common and repeated use) distinguish this application of formal methods from some other applications. A standard is meant to be a specification that will be implemented by a wide range of developers; many, perhaps the majority, of whom will have had no part (as organizations) in the development of the standard.

The last item 'optimum degree of order' (the context being 'aimed at the achievement of the optimum degree of order in a given context') intrigues the author. What is the concept of optimum degree of order in a subject area?

3 Formal Methods and Standards

3.1 Introductory remarks

Professor Hoare [20], in his inaugural lecture to the University of Oxford, said:

“The standardization of languages and interfaces in hardware and software is a vital pre-condition for free competition and for propagation of technical advances to large numbers of satisfied customers. The construction of a mathematical specification for these standards offers the same promise of improved quality in design that I have described before; there is also a hope of reducing the ambiguities and misunderstandings which lead to errors and incompatibility in various implementations of the standard, and which have prevented full benefit from being obtained from existing standards.”

In his lecture, Professor Hoare then went on to describe the work in Oxford on formal definition of the occam language. His emphasis on quality, reduction of ambiguities and misunderstandings is an emphasis which this author would strongly support.

ISO envisages three phases to the use of formal methods in standardization:

- Phase 1 - use by the Committee defining a standard;
- Phase 2 - a formal description as an annex to a natural language description;
- Phase 3 - the formal description constitutes the provisions of the standard, accompanied by a natural language commentary.

Formal methods are being applied in a wide range of activities within ISO and other standards-making bodies, for example (the list is certainly not meant to be exhaustive):

- Programming languages
- Communications protocols
- Open Distributed Processing
- Graphics and multimedia
- Open Document Architecture
- IEEE Floating point standard
- Test case generation (e.g. IEEE POSIX)

A recent special issue of the journal *Computer Standards and Interface* gives a good overview of current work in some areas of standardization on applications of formal description techniques [26].

In the following sections, some of the work in graphics and multimedia systems, in which the author has been involved, is described.

3.2 The Graphical Kernel System, GKS

The first ISO standard for computer graphics programming was published in 1985 after a lengthy period of development. The Graphical Kernel System, GKS, is a standard for 2D interactive graphics [22]. The author was one of the document editors of GKS and it was this experience that first aroused the author's interest in applying formal techniques to the description of the standard. The specification work done with GKS:1985 was *post hoc*.

The definition of GKS started in the days when storage tubes and vector refresh displays were the main types of output device in common usage. Raster displays were reserved for the rich, though by 1985 this situation had changed quite dramatically. The need to accommodate storage tubes and vector refresh displays (which have quite different updating characteristics) led to some complex mechanisms in the standard for controlling when picture changes would take place and when the picture would be redrawn. The earliest work constructed a simple model of the GKS system and investigated the circumstances under which equivalent pictures would be produced [11]. The model was expressed in VDM.

An approach to describing the output primitives in GKS was described in [10]. Essentially a primitive was described by the set of points in a world coordinate space covered by the primitive and the attribute values associated with each point. The paper described the different mechanisms in GKS for binding attribute values to primitives and the effects of different styles (such as filling style for area primitives).

The input devices and their operating modes in GKS were described in two papers [12, 13]. Input in GKS is described in the standard in terms of a number of processes. Their behaviours were modelled in CSP. The behaviour expressions revealed some similarities between the operating modes that had not been apparent before. The second

paper of the pair went on to discuss how hierarchical input devices could be built, based in part on the fresh insight gained through the specification work.

The 1985 standard came up for review, officially in 1990, though almost as soon as the document was published work started on gathering comments and suggestions for changes. Eventually a revised standard was published in 1994 [23, 4]. Quite a lot of specification work went into the new standard, though this was not an official part of the ISO process.

The specification work on GKS:1985 revealed that the standard did not have a very clean architecture. Other considerations, for example the relationship between GKS and the Computer Graphics Metafile standard, also revealed architectural shortcomings. Much effort went into defining a clean architecture for the revised standard. A specification was given in OBJ [7].

GKS-94 also incorporated a new output primitive, called the design primitive. A design primitive is constructed by extruding a tiling through a stencil. Formal specification was also used to model the design primitive and this helped to eliminate some problems and errors [27].

The specification work also influenced the style of the GKS-94 definition.

The impact of the specification work will be familiar from other areas:

- it enabled us to detect problems at an early stage and revise the design accordingly;
- the UK comments on the drafts of GKS were driven by the specification in some areas;
- above all the specification work helped us to think very carefully about what we were doing which led to improved models and concepts in the standard.

3.3 PREMO

When the work on GKS revision began, it was recognized that not all the emerging requirements could be accommodated within a single revised standard. Some requirements went far beyond the scope of GKS. Consequently work started on a ‘New Application Program Interface’ standard to address these emerging requirements. Progress initially was very slow as there was no clear direction to the work, however, a direction eventually emerged and a new project called PREMO (PResentation Environments for Multimedia Objects) came into being. At the time there were already a number of standards published dealing with multimedia, for example MHEG (dealing with interchange of multimedia/hypermedia documents), HyTime (structured hypermedia documents at a high level of abstraction) and MPEG and JPEG (coding/decoding techniques for particular media). However, there were large application areas that these standards did not address, for example:

- high-end simulation systems;
- scientific/engineering visualization systems combining various media (e.g. sound and graphics);
- systems with advanced multimodal user interfaces;
- virtual reality systems and their applications.

PREMO addresses application areas such as these.

The major features of PREMO can be briefly summarised as follows.

- *PREMO is a Presentation Environment.* PREMO aims at providing a standard ‘programming’ environment in a very general sense. The aim is to offer a standardized, hence conceptually portable, development environment that helps to promote portable multimedia applications. PREMO concentrates on the application program interface to ‘presentation techniques’; this is what primarily differentiates it from other multimedia standardization projects.

- *PREMO is aimed at Multimedia presentation*, whereas earlier standards concentrated either on synthetic graphics or image processing systems. Multimedia is considered here in a very general sense; high-level virtual reality environments, which mix real-time 3D rendering techniques with sound, video, or even tactile feedback, and their effects, are, for example, within the scope of PREMO.
- *PREMO is a framework*. This means that the PREMO specification does not provide all the possible object types for making graphics or multimedia. Instead, PREMO provides a general programming framework, a sort of middleware, where various organizations or applications may plug in their own specialized objects with specific behaviour. The goal is to define those object types which are at the basis of any multimedia development environment, thereby ensuring interoperability.

There are four parts to PREMO, all of which have just reached the status of Draft International Standard.

- *Part 1: Fundamentals of PREMO* - defines the PREMO object model.
- *Part 2: Foundation component* - defines basic object types, e.g. for event handling, synchronization, etc.
- *Part 3: Multimedia system services* - this part was derived from material submitted by the Interactive Multimedia Association (IMA) and deals with media format abstractions, capabilities and constraints, data flow abstractions, media stream abstractions and grouping of resources.
- *Part 4: Modelling, rendering and interaction* - this part deals with modelling, presentation and interaction with multidimensional data using multiple media in an integrated way and provides an extensible framework for primitives and device hierarchies.

In July 1993, SC24 (the ISO/IEC Committee responsible for the PREMO project) appointed a Special Rapporteur (Graham Reynolds, then at CWI, Amsterdam) to report on the applicability of formal description techniques to SC24 standards. His report was presented in June 1994 [28]. SC24 endorsed the recommendations of the report and passed resolutions:

- encouraging use of formal description techniques during the development of SC24 standards;
- endorsing the use of Object-Z [5, 15, 16] in the development of PREMO;
- encouraging publication of formal descriptions of standards according to ISO guidelines.

A group of people then started to use formal description techniques to describe parts of the emerging PREMO documents. The main areas addressed have been:

- the object model [8]
- approaches to media content description (this goes beyond what is actually defined in any of the parts of PREMO) [9]
- synchronization objects [14]

The object model description [8] used a two level approach, the first describing how objects can interact with each other (object model and runtime behaviour) and the second how individual objects behave. The first level specification used Z, the second used Object-Z. There was an informal linkage between the two levels, notational conventions were adopted for referring to first level entities in the second level specifications. The reasons for choosing this two level approach were:

- there is no standardized object model (yet);
- different application domains are likely to require, or at least use, different object models;

- the PREMO object model did not match the object model in any existing object-oriented specification language and indeed, it is unlikely that the object model in a multimedia system will match that in a general purpose specification language.

The drawback to this approach is that the linkage between the levels is informal, which means that reasoning between models can at best be rigorous.

The synchronization functionality is the area of PREMO that has received the most attention in the specification work. Proposals for synchronization functionality were developed alongside the specification work [17] so the text of the PREMO document and the specification have been kept closely aligned. The specification work has revealed some quite subtle problems in drafts of the natural language text.

PREMO provides an event-based synchronization mechanism, based on the idea that PREMO objects can be active and continuous media are modelled in PREMO as concurrent activities that have to reach specific milestones at user-definable synchronization points. PREMO also provides time-related synchronization that is built on this basic event model.

The key concepts are:

- Active objects progress along a 1-D progression space, which may be extended real, integer or time. Time is an abstract type in PREMO, which makes no commitment to discrete, continuous, or other form of representation. The notion of extension (by plus and minus infinity) is useful for describing unbounded media streams, for example microphone input and cyclic presentations, such as cine loops.
- Synchronization elements are attached at reference points on the progression space of synchronizable objects.
- A synchronization element consists of a reference to an event, a reference to a PREMO object that implements a Callback interface, and a wait flag. The Callback object will often be an event handler object, whose function is to dispatch the event to other objects that have registered interest in events of this type.
- When a synchronizable object reaches a particular reference point, the callback operation on the object specified in the synchronization element is invoked, with the associated event passed as a parameter.
- A synchronizable object encompasses a finite state machine that controls traversal through the internal progression space.

Some difficulties were encountered in this work, which are discussed in detail in [14].

- It was necessary to introduce additional structure in the specification, which is not in PREMO, in order to describe the progression behaviour of an active object and how reference points in the interval between progression points are treated.
- Representing the extended real, integer and time types is problematic. PREMO itself would have benefitted from a more abstract description which pulls out the important properties of these types (for example the notions of progression, bounded and unbounded space, cyclic behaviour). This could have been done in the formal description however this would have increased the gap between the structure of the formal description and that of the normative part, making it more difficult to understand consequences of behaviour in terms of provisions of the standard.
- PREMO is explicit about the use of object references (specific types are defined in PREMO for object references), whereas in Object-Z references are part of the semantic model of the language.
- The PREMO object model has the concepts of read-only attributes, read/write attributes and internal (i.e. protected) operations; these features are not directly mirrored in Object-Z. This led to the introduction of some naming and other conventions.
- Some notational extensions were developed to deal with exceptions in the specification in a clean way.

- Object initialization is treated differently in PREMO and in Object-Z. PREMO, for example, has an `initializeOnCopy` operation. It is difficult to see how to model these types of operations in the specification without resorting to a low, operational, level of detail in the specification.
- Name overloading is used in PREMO in ways not allowed in Object-Z.

Faconti and Massink have also looked at the PREMO synchronization objects, they used an approach based on LOTOS. This work is described in a paper to this workshop (*Using LOTOS for the Evaluation of Design Options in the PREMO Standard*) and in a paper to the Eurographics DSVIS '97 workshop [18]. In the latter paper they show a new way to obtain a specification in a constraint oriented style. In their approach, they let methods correspond to actions and values of control variables with processes. Each process consists of actions that are enabled for the value of the control variable that is modelled by the process. This approach leads to Basic LOTOS specifications that are directly suitable for computer assisted analysis such as model checking and simulation.

Blair *et al.* [3, 2] have also described an approach to specification of multimedia systems in the context of the ISO work on Open Distributed Processing. Their work is motivated by multimedia systems in general, they do not discuss PREMO in particular. They are concerned with Quality of Service issues and real-time synchronization issues (e.g. maintaining lip synchronization between video and audio channels). They propose a framework which encourages the co-existence of different languages and avoids directly embedding timing assumptions in the language. The key features are:

- separation of concerns between *behaviour* and *requirements* - the *actual* behaviour of a system and the *desired* behaviour (expressed as properties);
- they distinguish between behavioural time and real-time. Behavioural time is time associated with time passing in an algorithm, a clock tick, etc., but does not relate an event to real time;
- real-time assumptions ground behavioural time in real-time;
- in the ODP context, they discuss refinement of behaviour and refinement of requirements.

They then describe an approach to specification which is based on standard LOTOS for describing abstract behaviour and QTL (Quality of Service Temporal Logic) for the specification of requirements. QTL is a logic designed to be compatible with LOTOS, whose features include linear time, bounded operators, discrete time domain and past-tense operators (these are useful in environments where messages may be lost). LOTOS events may be propositions in QTL. They are developing verification approaches for QTL/LOTOS.

Overall, formal methods have been applied in a lightweight way during the development of PREMO. The use of formal methods has helped us to formulate the PREMO concepts more clearly and has helped to identify shortcomings, errors, omissions etc. at an early stage. The notation used in the PREMO documents for describing object classes and operation signatures owes much to Object-Z and there is quite a close level of notational correspondence between the formal specification of the synchronization objects in [14] and the corresponding parts of the PREMO document itself, though the PREMO document does not formally describe behaviour of object classes.

No attempt has been made to replace the informal description of PREMO by a formal text, for a number of reasons.

- It is important that experts in all participating nations can read the description of the standard. Currently within the PREMO project, this would not be the case, were a formal text to constitute the provisions of the standard. The lack of experts who can read and write specifications is an impediment to their uptake in our area of standardization. People want to be convinced of the benefits before they will invest in the learning process.
- There are some technical difficulties, alluded to earlier, concerning mismatches between the formal specification language and the PREMO object model which would be difficult to 'paper over' in a formal text.
- Keeping a large specification readable is a continuing problem.
- Writing a specification of a system of the complexity of PREMO is a serious undertaking. The effort is far less than is needed for an implementation, but is nevertheless significant and it is difficult to find the funding to do this kind of work.

4 Appraisal and Directions

Our experience in applying formal methods in standards mirrors that of other people applying such methods in other areas. Three quotations from articles in a recent special issue of *IEEE Computer* on formal methods summarize our experience.

Jones [24], writes “.. it was important to understand the formal basis but to use - in most cases - a less than completely formal approach; this course was proposed on the assumption that one was capable of filling in the formal details where necessary ... I today teach courses on how to sketch abstract models of systems where a minimum of emphasis is put on the notational details, and the central idea is that of presenting an abstract state for a system. It is amazing how much understanding of an architecture is captured in this abstract state”. Jones goes on to write “Today, formal methods are mainly used in the safety critical area where their detailed application can be justified because of the danger of loss of life. The use of formal methods in a lighter way is both a key to using them on larger-scale applications and a way of penetrating fields outside the safety-critical area.”

Hall’s article [19] contains the remarks “... we are making fairly ‘shallow’ use of formality - we did not attempt any proofs of consistency or of particular properties. Nevertheless, we found the specification enormously useful in pinning down just what it was that we were going to build”. He concludes: “I believe the right question to ask is ‘what can formal methods contribute to improve the quality and decrease the cost of our systems?’ ”.

P. Zave writes “The conceptual gap between application domains and mathematics must be bridged by building mathematical models of the application domains. Within an appropriate model, formal language is extended to include the vocabulary and relationships of the domain. The lack of appropriate application models, on the other hand, constitutes a large barrier to the use of formal methods in an application domain”. She concludes “because of the difficulty of the task ... the conclusion is obvious: Finding the best way to use formal methods in an application domain is research, not development. It is an unusual kind of research, although certainly not unheard-of. It is intellectually challenging and rewarding, at least when the standards for results are set high. And it is probably the most effective thing we can do to bring formal methods into widespread use”.

In this author’s view, there is still some way to go before we arrive at a mathematical model for multimedia systems. The work described in this paper has taken some steps; other authors have also taken steps, but there remains the need to draw many strands of work together in one framework.

The work described here also points to the need to be able to use different methods to describe different aspects of the behaviour and properties of a system and to be able to do so in a consistent and meaningful way. This is a point taken up by Clarke and Wing in a recent (December 1996) *Computing Surveys* article [6], in which they point out the need, in the context of methods of integration for ‘finding a suitable style for using different methods together; and finding a suitable meaning for using different methods together’.

The work described in this paper started with computer graphics systems and has now moved on to multimedia systems. The author leaves the last word with Professor Jones:

“Another very speculative area is that of multi-media systems. These are clearly becoming important in practice and it would be desirable if formalism could be applied before the systems become too unstructured for that to be an appetising possibility.”

from a paper entitled ‘Some Practical Problems and Their Influence on Semantics’ [25].

References

- [1] D.B. Arnold and D.A. Duce. *ISO Standards for Computer Graphics: The First Generation*. Butterworths, 1990.

- [2] G.S. Blair, L. Blair, and J.B. Stefani. A specification architecture for multimedia systems in Open Distributed Processing. *Computer Networks and ISDN Systems*, 29:473–500, 1997.
- [3] L. Blair, G. Blair, H. Bowman, and A. Chetwynd. Formal specification and verification of multimedia systems in open distributed processing. *Computer Standards and Interfaces*, 17(5-6):413–436, 1995. Special Issue on Formal Methods and Standards.
- [4] K.W. Brodlie, L.B. Damnjanovic, D.A. Duce, and F.R.A. Hopgood. GKS-94: An Overview. *IEEE Computer Graphics and Applications*, 15(6):64–71, 1995.
- [5] D.A. Carrington, D.J. Duke, R.W. Duke, P. King, G.A. Rose, and G. Smith. Object-Z: An object-oriented extension to Z. In S. Vuong, editor, *Formal Description Techniques (FORTE'89)*. North Holland, 1990.
- [6] E.M. Clarke and J.M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.
- [7] D.A. Duce and L.B. Damnjanovic. Formal Specification in the Revision of GKS: An Illustrative Example. *Computer Graphics Forum*, 11(1):17–30, 1992.
- [8] D.A. Duce, D.J. Duke, P.J.W. ten Hagen, I. Herman, and G.J. Reynolds. Formal methods in the development of PREMO. *Computer Standards and Interfaces*, 17(5-6):491–509, 1995. Special Issue on Formal Methods and Standards.
- [9] D.A. Duce, D.J. Duke, P.J.W. ten Hagen, and G.J. Reynolds. PREMO - an initial approach to a formal definition. *Computer Graphics Forum*, 13(3):C–393–C–406, 1994.
- [10] D.A. Duce and E.V.C. Fielding. Towards a Formal Specification of the GKS Output Primitives. In A.A.G. Requicha, editor, *Eurographics '86*, pages 307–323. North Holland, 1988.
- [11] D.A. Duce, E.V.C. Fielding, and L.S. Marshall. Formal Specification of a Small Example Based on GKS. *ACM Transactions on Graphics*, 7(3):180–197, 1988.
- [12] D.A. Duce, P.J.W. ten Hagen, and R. van Liere. Components, Frameworks and GKS Input. In F. R. A. Hopgood W. Hansmann and W. Strasser, editors, *Eurographics '89*. North Holland, 1989.
- [13] D.A. Duce, P.J.W. ten Hagen, and R. van Liere. An Approach to Hierarchical Input Devices. *Computer Graphics Forum*, 9(1):15–26, 1990.
- [14] D.J. Duke, D.A. Duce, I. Herman, and G. Faconti. Specifying the PREMO Synchronization Objects. Technical Report 02/97-R048, ERCIM Research Reports, 1997. Available from <http://www-ercim.inria.fr/>.
- [15] R. Duke, P. King, G. Rose, and G. Smith. The Object-Z specification language: Version 1. Technical Report 91–1, Software Verification Research Centre, University of Queensland, 1991.
- [16] R. Duke, G. Rose, and G. Smith. Object-Z: A specification language advocated for the description of standards. *Computer Standards and Interfaces*, 17(5-6):511–533, 1995. Special Issue on Formal Methods and Standards.
- [17] D.J. Duke (editor). Time and synchronisation in PREMO: A formal specification of the NNI proposal. Technical Report Technical Report OME-116, ISO/IEC JTC1 SC24/WG6, 1995. Available at <ftp://ftp.cwi.nl/pub/Premo>.
- [18] G.P. Faconti and M. Massink. Investigating the Behaviour of PREMO Synchronization Objects. In M.D. Harrison and J.C. Torres, editors, *4th Eurographics Workshop on Design, Specification and Verification of Interactive Systems*. Springer-Wien, 1997.
- [19] A. Hall. What is the Formal Methods Debate About? *IEEE Computer*, 29(4):22–23, 1996.
- [20] C.A.R. Hoare. *The Mathematics of Programming*. Clarendon Press, Oxford, 1986.

- [21] ISO. <http://www.iso.ch/>.
- [22] ISO 7942. *Information processing systems, Computer graphics, Graphical Kernel System (GKS) functional description*. Geneva ISO Central Secretariat, 1985.
- [23] ISO 7942. *Information technology, Computer graphics and image processing, Graphical Kernel System (GKS) Part 1: Functional description*. Geneva ISO Central Secretariat, 1994.
- [24] C.B. Jones. A Rigorous Approach to Formal Methods. *IEEE Computer*, 29(4):20–21, 1996.
- [25] C.B. Jones. Some Practical Problems and Their Influence on Semantics. In *Programming Languages and Systems - ESOP96*, number 1056 in Lecture Notes in Computer Science, pages 1–17. Springer-Verlag, 1996.
- [26] H. Kilov. The formal way (Guest editorial). *Computer Standards and Interfaces*, 17(5-6):409–412, 1995. Special Issue on Formal Methods and Standards.
- [27] Ph. W. Nehlig and D.A. Duce. GKS-9x: The Design Output Primitive, An Approach to a Specification. *Computer Graphics Forum*, 13(3):C–381–C–392, 1994.
- [28] G.J. Reynolds, D.A. Duce, and D.J. Duke. Report of the ISO/IEC JTC1/SC24 special rapporteur group on formal description techniques. Technical Report ISO/IEC JTC1/SC24 N1152, ISO/IEC, 1994.