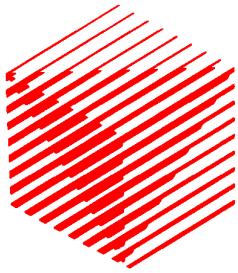
08/97-R050
ERCIM Computer Graphics Network

European Research Consortium
for Informatics and Mathematics

ERCIM



The Changing Face of Standardization: A Place for Formal Methods?

by David Duce, David Duke,
Giorgio Faconti and Ivan Herman

ERCIM Research Reports

The Changing Face of Standardization: A Place for Formal Methods?

David Duce[1], David Duke[2], Giorgio Faconti[3] and Ivan Herman[4]

[1]Rutherford Appleton Laboratory, Chilton, Didcot OX11 0QX, UK
[2]University of York, Heslington, York YO1 5DD, UK
[3]CNR-CNUCE, Pisa, Italy
[4]CWI, Amsterdam, The Netherlands

Keywords: Standards; PREMO; Multi-media; Distributed Systems; Object-Z

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. Standards development, however, has some significant differences from product development, and comes with specific needs and constraints that affect the use of formal description techniques. This paper describes the role of formal methods in standardization, and reports on the authors' experiences in using these methods in the development of a new International Standard for distributed multi-media.

1. Introduction

The authors of this paper have long experience of the development of international standards for computer graphics under the auspices of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The last few years have seen significant changes in the process by which international standards are generated and the methods used therein. For many years concerns have been expressed at the length of time it takes to

Correspondence and offprint requests to: David Duce, Rutherford Appleton Laboratory, Chilton, Didcot OX11 0QX, UK. Email dad@inf.rl.ac.uk

converge on the content of an international standard (4 - 5 years is typical in the computer graphics area) and recently changes have been seen in the ISO/IEC standardization process which aim to reduce the elapsed time from entry into the standardization process to publication of an international standard.

The authors all have experience of using formal methods to support the development of standards for computer graphics and multi-media. In this paper we explore the role that formal methods have played in this process in the past, and the role such methods might play in the future. Key concerns for developers of standards are clarity, consistency and compatibility. The standards that have been developed in the computer graphics and multi-media areas within ISO/IEC fall mainly into two categories: standards that define the functionality of a software system and standards that define file formats for the storage and transfer of graphical and multi-media information. Key concerns for implementors of these standards are that the description of the standard should be clear and concise, free from ambiguity and inconsistency, in order to minimize the time wasted in backtracking design decisions due to false readings of the standard. Implementors are also concerned that their implementation should interwork with other implementations. This leads into the areas of conformance statements and conformance testing. The first is concerned with what requirements an implementation has to conform to and the second with how adherence to the conformance statement can be measured. Formal methods, by improving clarity and reducing inconsistency, can help to reduce implementation costs and help to ensure compatibility between different implementations.

It should be stressed that the views expressed in this paper are those of the authors and are not official views of ISO/IEC.

The remainder of the paper is structured as follows. Section 2 describes the process by which a Standard is developed, and recent changes to this process that increase the need for rigorous description techniques. These general issues are placed into context in Section 3, where issues and problems encountered by the authors in the development of the PREMO standard [ISO96a] are described. Finally, opportunities and challenges for the use of formal methods in future standardization actions are explored in Section 4.

2. The Standardization Process

Standards are defined by ISO in the following way[ISO96c]: '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. International Standards thus contribute to making life simpler, and to increasing the reliability and effectiveness of the goods and services we use.'

Standards in Information Technology are the responsibility of a joint technical committee (JTC 1) established by the Council's of ISO and the International Electrotechnical Commission (IEC) in 1987. The work of standardization is undertaken by subcommittees (SCs) and working groups (WGs).

2.1. The Changing Process

The ethos of standards-making is consensus building. The objective of standardization is to achieve consensus amongst the parties participating in the devel-

opment of a standard, rather than decision making based on counting votes. Consensus is defined as 'general agreement, characterized by the absence of sustained opposition to substantial issues by any important part of the concerned interests and by a process that involves seeking to take into account the views of all parties concerned and to reconcile any conflicting arguments. Consensus need not imply unanimity'. The process of building consensus is, by its very nature, time-consuming.

Computer graphics is one specific area in which ISO has been active since 1978 and thus can provide a useful example of the changes that are taking place. The first international standard for computer graphics programming, the Graphical Kernel System (GKS) was published by ISO in 1985, at the end of a development process that started in 1979. At that time there were five steps in the development of an International Standard [AD90, ISO85]:

1. New Work Item (NWI). When a need for a new standard arises, a New Work Item proposal is made to JTC 1. The proposal is balloted within JTC 1 (a three-month ballot period is allowed); if approved, the work is assigned to a SC.
2. Working Drafts (WD). ¿From a baseline document, a series of WDs were prepared, until a high degree of consensus was reached on the technical content of the standard. Each WD was reviewed by member bodies of the SC, usually with a period of 3 months allowed for comments. Maintenance of the document is the responsibility of a Project Editor (usually called a Document Editor in SC24), appointed by the SC.
3. Draft Proposal (DP). When the major issues had been resolved, the final WD was submitted for registration and ballot as a Draft Proposal. Again this involved a 3 month ballot period. Substantial changes in the content of a DP could result in further ballots at this level.
4. Draft International Standard (DIS). When consensus was reached on the technical content of the standard, the document (now called a DIS) was then circulated for a six month combined ballot by JTC 1 and the SC concerned.
5. International Standard (IS) When comments on the DIS ballot had been resolved, the document was submitted to the ISO/IEC Councils' for approval. International standards are reviewed at intervals of not more than five years. The review process would follow a similar pattern to that outlined above.

By merely looking at the lengths of the balloting periods involved, it is clear that standardization according to this process is a lengthy affair. Since the early days of GKS standardization, ISO/IEC have instigated changes to the basic process designed to ensure the more timely delivery of information technology standards. As a result, the 1995 edition of the ISO/IEC Directives for the Technical Work of JTC 1[ISO95a], describe a rather simpler process.

1. New Work Item (NWI). This stage is essentially unchanged. For approval of a new work item, a majority of all participating members (National Standardization Bodies) of JTC 1 is required, and in addition at least 5 of the participating members of the Subcommittee to which the project will be assigned must commit to active participation in the project.
2. Working Draft (WD). NWI proposers are encouraged to submit a WD with the NWI proposal. Refinement of working drafts takes place at Working Group level. Progression to the next stage is decided at Subcommittee level.

3. Committee Draft (CD). The Committee Draft replaces the Draft Proposal of the earlier procedures. Voting on Committee Drafts takes place at the Subcommittee level. Normally this will be a postal ballot with a subsequent meeting to consider the votes and comments, but the procedures do allow for the vote to take place at a meeting. The ballot period is a minimum of 3 months but can be extended if the complexity of the draft so merits.

4. Draft International Standard (DIS). When consensus has been demonstrated and substantial support obtained from the SC members, the document progresses to Draft International Standard and is voted on at JTC 1 level. Normally the ballot period is four months.

5. International Standard (IS). The final text of the DIS is then submitted for publication as an international standard.

The main difference between the 1995 Directives and the 1985 Directives is in the earlier stages of the process. More of the process is delegated to the Subcommittee level, with fewer ballots at the higher, JTC 1, level.

In a recent development, JTC 1 has extended the process to encourage the 'transposition' of technical specifications from sources outside JTC 1 into international standards. New procedures have been agreed which will come into effect for a trial period from 1 January 1997. The aim is to provide a route for Publicly Available Specifications (PAS) to be transposed into international standards. The procedures use a Fast Track process which allows an organization to submit a document into a final JTC 1 ballot (of six months duration) to become an International Standard. JTC 1 have established criteria as a basis for judging whether an organization submitting a PAS can be recognized and whether the specification can be accepted as a candidate for transposition. The criteria which organizations have to satisfy relate to the nature of the organization (e.g. membership, ease with which representatives of business, government and other entities can participate). The document related criteria are concerned with the quality of the proposed standard and extent of the consensus that the document has achieved. The quality criteria include[ISO95b]:

1. How well are all interfaces specified?

2. How easily can implementation take place without need of additional descriptions?

3. What proof exists for successful implementations?

4. What means are used to provide definitive descriptions beyond straight text?

The PAS route is being discussed for Java standardization. Within the computer graphics area, VRML (Virtual Reality Modelling Language), which was developed outside ISO/IEC by the VRML Architecture Group (now absorbed into the VRML Consortium) is being brought into the ISO/IEC arena, but at an earlier stage of processing than the PAS route requires. VRML is being standardized by mutual agreement between ISO/IEC and the VRML Consortium; the document entered the process at the Committee Draft rather than the final ballot stage, nevertheless, issues of quality also apply to documents entering via this route and careful consideration of such issues is an important factor in ensuring smooth progression of the document through the ISO/IEC process.

2.2. The PREMO Experience

PREMO (PResentation Environments for Multi-media Objects) is an emerging international standard for multi-media presentation, currently under development in ISO/IEC JTC 1/SC24 [ISO96a, HCD+94, HRL96, HCD+97]. All the authors of this paper contribute to the development of this standard. ¿From previous experience in developing international standards, and the use of formal description techniques in the context of computer graphics standards [AD90], the authors argued for the use of formal methods to support the development of the standard. In July 1993, SC24 appointed a Special Rapporteur for Formal Description Languages (G.J. Reynolds, then at CWI, Amsterdam), and invited him to provide an initial report on the applicability of formal description techniques to SC24 standards [RDD94]. The terms of reference of the Special Rapporteur requested 'study of FDTs particularly with regard to formally specifying object behaviour and interfaces'.

The study group chose to base its work on the Object-Z notation [CDD+90, DKRS91, DRS95], because expertise in this notation was available within the group and it was known that other groups within at least one other SC had experimented with Object-Z in their work [ISO97, DRS95]. Some initial studies were carried out[DDtH+94, DDtH+95] which resulted the following recommendations being made to SC24[RDD94].

1. SC24 should actively encourage the use of formal description techniques during the *development* of SC24 standards.
2. SC24 should endorse and encourage the work being carried out using Object-Z in the development of PREMO.
3. SC24 should encourage the publication of formal descriptions of SC24 standards as ISO/IEC Technical Reports, following the successful use of formal description in the development of a particular standard.

These recommendations fall into the first phase for the introduction of formal description techniques in the ISO Directives. The rationale for the recommendations is based on the following considerations.

- Formal description techniques provide a notation for expressing a mathematical model of a system. The value of FDTs is in the mathematical model, not the notation in which it is expressed. The real value of FDTs during the development of a standard is that they cause attention to be focussed on concepts and the relationships between concepts, and on issues such as completeness, consistency and structure to a much greater extent than does a natural language style of description.
- For the description of a standard such as PREMO, which was also in the process of defining an object model within the standard, it is important to choose a flexible descriptive notation rather than a notation which contains an in-built object model which will inevitably contradict that being defined in PREMO.
- It is at least as easy to write a badly structured and opaque specification as it is to write a badly structured and opaque program or natural language description of an artefact. Writing good specifications takes insight, skill, experience and patience. Reading a well-structured specification with a carefully written natural language commentary is not difficult. Because the

structure of formal descriptions and the structure of the artefact being specified are inter-twined, it really is important to use formal description during the development process.

- It is important to convince the developers of standards that formal description techniques do have tangible benefits. In our judgement, this is best done within the development phase of a standard, where the specification can be used to record and communicate key aspects of the design of the standard and can help to highlight errors and omissions in the emerging standard.
- The work done on PREMO modelling shows that a formal description of key parts of the system is both feasible and helpful.

The next section describes the work that has been done to date.

3. The PREMO Specification

The definition of PREMO follows an object-oriented style. The first work carried out[DDtH+94] on the specification of PREMO explored the use of Object-Z to describe event-based synchronization and discrete and continuous presentations [DDtH+94]. A basic approach to these problems was described, which has been refined in subsequent work as the functionality of PREMO itself has evolved. The specification work has tracked the development of PREMO and insights gained through the specification work have been fed back into the development process. It is infeasible to describe all of the technical issues that arose in the development of the PREMO specifications in this paper. Rather, this section gives an insight into two of the most significant areas where specification was carried out, specifically the PREMO synchronization facilities (section 3.1), and the underlying object model (section 3.2). The approach taken to the formal specification of PREMO is extended, in part, into the normative text, where Z-like data type definitions and the structure of Object-Z class definitions are used to present the functional provisions of the Standard. Section 3.3 illustrates this briefly. The section concludes with an evaluation and discussion of the use of formal methods within the development of the PREMO Standard.

3.1. Synchronization

Multi-media applications often wish to use multiple instances of continuous media data concurrently, for example an animation sequence with some accompanying sound, with links to video segments. This problem is referred to as inter-media synchronization, while the term intra-media synchronization is used for the task of maintaining the presentation of data at a sufficient rate and quality for human perception. Both forms have received significant attention in the multi-media literature, see for example [GT95] or [Buf94] for further information and references on the topic. Only inter-media synchronization is discussed in this paper, and for brevity the term synchronization is henceforth used to refer to this form only.

The PREMO synchronization model [HCD+97] is based on the fact that objects in PREMO can be active. Different continuous media (e.g. a video sequence and corresponding sound track) are modelled as concurrent activities that may have to reach specific milestones at distinct and possibly user-definable synchronization points. An event-based synchronization approach forms the basic layer

of synchronization in PREMO, and in line with the object-oriented approach of PREMO, is defined in terms of object types, including:

- synchronizable objects, which have an internal progression space and which form the supertype of, e.g., various media object types;
- synchronization elements, which can be placed on a user-definable subset (the span) of the coordination space of a synchronizable object to generate events; and
- event handlers, which may be used to manage complex synchronization patterns among synchronizable objects through the synchronization elements placed on the span of synchronization objects.

A synchronizable object is a finite state machine that controls the position and progress through an ordered set of coordinates, some of which may contain synchronization elements that can be used to organize the behaviour of a system constructed using such objects. The intention is that object types representing different kinds of media (video, sound, etc.) will inherit from this object type and specialize the coordinate system and state machine in an appropriate way. As the specification of these services was carried out in parallel with the development of the normative text, it was desirable to keep the two descriptions as close as possible. So, for example, rather than use a Z-style free type definition [Spi92] for the various modes of the synchronizable state machine, explicit numerical constants are defined in the specification:

$$SyncMode == \mathbb{N}$$

$$STOPPED, PLAY, PAUSED, WAITING : SyncMode$$

$$STOPPED = 0 \land PLAY = 1 \land PAUSED = 2 \land WAITING = 3$$

As a consequence of adopting this style, operations on the synchronizable object type that accept a *SyncMode* value as input must check that the value denotes a valid state. Operations in PREMO may raise exceptions, and this is mirrored in the way that operations are specified by the adoption of a convention for naming exceptions and defining their cause and effect. For example, events can be associated with the mode transitions of a synchronizable object through *ActionElement*; the fragment of Object-Z given below represents the corresponding state attribute and one of the operations that manipulates it, and illustrates the style of exception specification.

$$actions : (SyncMode \times SyncMode) \pfun ActionElement$$

...

setActionOnPair

$$\Delta(actions)$$
$$stateOld? : SyncMode$$
$$stateNew? : SyncMode$$
$$action? : ActionElement$$

$$stateOld? \in SyncMode \land stateNew? \in SyncMode$$
$$\longrightarrow \boxed{\text{exc}}\ \text{WrongState}$$

$$actions' = actions \oplus \{(stateOld?, stateNew?) \mapsto action?\}$$
$$\text{WrongState} \longrightarrow actions' = actions$$

These may seem like minor concerns, but they are important in making the use of a formal description practical in the context of standards development.

Another aspect of the *Synchronizable* object type both illustrates the power of formal description to describe behaviour succinctly and precisely, and some additional problems that were encountered in shadowing the development of the normative standard through the use of specification techniques. This is the definition of what it means, when an object is placed into 'play' mode, to progress through the coordinate space, an issue which is complicated by a number of factors, including:

- the subset of a synchronizable object's coordination space that is to be traversed (the span) can be set and modified by the application;
- the direction of progression is variable (forward or backward), and the the object can be directed to loop over its span range either a finite or (potentially) infinite number of times;
- in general, only the media content located at a subset of the coordinates in the span will be presented, due to sampling necessary for continuous media or imposed by quality of service requirements; and
- regardless of the coordinates visited for presentation, every synchronization element on the span must be processed as traversal through the span is performed.

An operation, *progressPosition*, is defined in the interface of *Synchronizable* to return the next position in the span at which a datum will be presented. The challenge in the specification was to describe the behaviour that the synchronization object must exhibit *between* successive presentation positions. Although a clear specification of progression might best be achieved through a temporal formalism, e.g. interval temporal logic [MS87], the cost of casting the PREMO object types into this kind of framework would be prohibitive. Instead, the specification uses an 'internal' variable to define a *stepping* mode, within which the current position is moved through the span, processing synchronization elements, until either the next position for presentation is reached, or some other action occurs that causes the object to exit from play mode. For example, when an object encounters a synchronization element, it may be required to enter a *WAITING* mode; this can be used for example to synchronize the playback rates of different media objects. Once in *WAITING* mode, the *stop* operation might be invoked.

To address the issue of looping through the span, a *Location* type was defined, combining span coordinates with an iteration number. This is given below, along with other parts of the object state used to characterize progression behaviour. The *Synchronizable* object type is generic with respect to the domain of the coordinate space, which is constrained in PREMO to be either the integers, reals, or a *time* type, extended with infinity. It inherits from two other PREMO object types not discussed in this paper.

$$\begin{array}{|l} \mathit{Synchronizable}\,[C :: \mathbb{Z}_\infty \mid \mathbb{R}_\infty \mid \mathit{TIME}_\infty] \\ \mathit{EnhancedPREMOObject}\ \textbf{redef}\ (\mathit{initialize}, \mathit{initializeOnCopy}) \\ \mathit{CallbackByName} \end{array}$$

The following declarations are read-only attributes, i.e. each comes with an implicit operation for getting its value, but the value can only be changed by the action of specific operations in the interface of the type.

$$
\begin{array}{l}
currentDirection : Direction \\
loopCounter : \mathbb{N} \qquad \text{[number of loops completed]} \\
currentState : SyncMode \qquad \text{[playing, paused etc]} \\
currentPosition : C \\
maximumPosition : C \\
minimumPosition : C \qquad \text{[fixed bounds of the span]}
\end{array}
$$

The declarations in this next fragment of the specification are readable and writable attributes, i.e. each comes with implicit operations for setting and getting its value. The second pair of attributes identify the user-selectable subset of the coordinate space that is to be processed or presented.

$$
\begin{array}{l}
repeatFlag : \mathbb{B} \qquad \text{[should the presentation cycle?]} \\
nloop : \mathbb{N} \qquad \text{[total number of loops required]} \\
startPosition : C \\
endPosition : C \qquad \text{[user-definable boundary]} \\
\hline
minimumPosition \leq startPosition \\
startPosition \leq stopPosition \\
stopPosition \leq maximumPosition
\end{array}
$$

Next comes the definition of locations, and a total order relation over this type. These definitions are conceptually internal to the specification, that is, they do not describe attributes or structures that need appear in the interface of an implementation.

$$
\begin{array}{l}
Location == C \times \mathbb{N}_\infty \\
\\
\_prec\_ : Location \leftrightarrow Location \\
\hline
\forall\, c_1, c_2 : C;\ n_1, n_2 : \mathbb{N} \bullet \\
\quad (c_1, n_1)\ prec\ (c_2, n_2) \Leftrightarrow n_1 < n_2 \lor (n_1 = n_2 \land c_1 < c_2)
\end{array}
$$

No invariant is given to link the coordinates visited during traversal with the parameters that determine traversal behaviour, for example *nloop* and *startPosition*. Operations defined in the interface of the *Synchronizable* object type can update these parameters, and it simplified the specification if the relationship between these variables was captured as part of a 'framing' schema that could then be used to define the effect of such operations.

The final group of variables are used to define how progress is made during play mode. They include the state component, *stepping*, that indicates when an object is moving from one presentation location to the next, *refpoints*, which defines the synchronization elements that have been associated with specific points on the coordination space, and *loopStart*, which is the coordinate that progression will start from initially. The locations that remain to be traversed when the object is in *PLAY* mode define the *span*, while the relation $\prec$ defines the order in which these locations will be traversed.

$$
\begin{array}{|l}
\hline
stepping : \mathbb{B} \qquad \text{[true while moving from current to new]} \\
requiredPosition : Location \qquad \text{[determined by progressPosition]} \\
point : Location \qquad \text{[location during span traversal]} \\
refpoints : C \pfun SyncElement \qquad \text{[the sync. points]} \\
loopStart : C \qquad \text{[the starting coord for loops]} \\
span : \mathbb{P}\, Location \qquad \text{[locations to be traversed]} \\
\_ \prec \_ : Location \rel Location \qquad \text{[order of traversal]} \\
\hline
\dom refpoints \subseteq minimumPosition \upto maximumPosition \\
currentDirection = forward \\
\quad \Rightarrow loopStart = startPosition \land (\prec) \subseteq prec \\
currentDirection = backward \\
\quad \Rightarrow loopStart = endPosition \land (\prec) \subseteq prec^{-1} \\
\hline
\end{array}
$$

Two operation descriptions illustrate the economy of definition that is supported by the model given above. First, the *progressPosition* operation calculates the next location to be visited; it is expected that it will be specialised by subclasses to address behaviour specific to various types of media. The point in the coordinate space that will be visited next is returned as an output.

$$
\begin{array}{|l}
\hline
progressPosition \\
\hline
\Delta(requiredPosition, stepping) \\
newPosition! : C \\
\hline
currentState = PLAY \land \neg\, stepping \\
\exists\, count : \mathbb{N} \mid count < nloop \bullet \\
\quad (newPosition!, count) \in span \\
\quad requiredPosition' = (newPosition!, count) \\
stepping' \\
\hline
\end{array}
$$

Once a new location has been calculated, the object is placed into a 'stepping' mode. Once in this mode, the next point in the span to be visited must be calculated, bearing in mind requirements related to synchronization elements. A succinct description of this operation is given below.

$$
\begin{array}{|l}
\hline
\Phi step \\
\hline
\Delta(point, span, loopCounter) \\
\hline
stepping \land point \neq requiredPosition \\
point \prec point' \\
\neg\, requiredPosition \prec point' \\
\textbf{let}\, skipped == \{loc : span \mid loc \prec point'\} \bullet \\
fst(\!| skipped |\!) \cap \dom refpoints = \varnothing \\
loopCounter' = loopCounter + |snd(point') - snd(point)| \\
span' = span \setminus skipped \\
\hline
\end{array}
$$

As mentioned earlier, stepping mode is an artefact of the specification introduced to model the sequence of operations that are assumed to take place internally. The decision to use this approach raises an issue that goes beyond ones' taste in specification language, and which has implications for other standards and systems, for example the VRML 2.0 Standard [ISO96b] under development in SC24. In developing a standard, particularly in an area such as graphics

where performance is a non-trivial concern, there may be implicit assumptions about the execution model that will be used to realise the system. In the case of PREMO for example, the specification of *progressPosition* given above, and the semantics of the internal *stepping* mode, involve a level of operational detail that normally one would associate more with a design or implementation. The problem is that a more abstract description of the intended behaviour may be rather more difficult for committee members or even implementors to understand; state machines are after all a well understood engineering concept, a fact that has been borne out by the experience of others in designing languages to document requirements and specifications [LHHR94].

Three further schemas are needed to define the semantics of stepping mode; other operations within the *Synchronizable* object type define moding behaviour, access and modification of the attributes that define the span, and control over the placement of synchronization elements. The full specification of this object type and the other facilities, such as event handlers, that make up the PREMO synchronization model, are given in [DDHF97]. Examples of how the general specification presented here could be specialized to address specific media can be found in [DDtH+94]. Some appreciation for the effect that formally specifying these facilities has had on the PREMO Standard can be gained by comparing the previous reference with the specification developed from the original proposal for the synchronization facilities [Duk95].

3.2. The Object Model

The second aspect of PREMO for which formal specification has been extensively used is the PREMO object model. In common with other SC24 standards, PREMO is defined in a language independent way. However, for an object-oriented standard, the techniques used by SC24 in the past to define programming language independent standards (essentially describing data types and operations over them) were not sufficient to capture the degree of abstraction from a programming language required in PREMO. Different object oriented programming languages are based on different object models. The definition of PREMO was also required to be independent of any particular object model found in a concrete programming language. This led to the work, reported in [DDtH+95], to describe the PREMO object model. During the course of this work, the idea of an object model independent specification was refined considerably and again the result of the formal specification work gave valuable insights that were fed back into the PREMO documents.

The PREMO object model is specified in Z, rather than Object-Z, to avoid confusion between the semantics of objects and their types in Object-Z and those of the objects and types in the PREMO object model itself. The specification encompasses aspects of object models including: object references, identity, object types, operation signatures, inheritance, subtyping and operation dispatching. Much work has been carried out on the semantics of object-oriented systems and models, see for example [BE95, HJ95, DRS95], and a number of the issues mentioned above are, at least now, well understood. Two aspects of the PREMO work are however particularly novel. The first is the treatment of operation dispatching. Objects in PREMO communicate by sending messages that cause the receiver to perform a specified operation using given arguments. Although this is similar to the model of message passing assumed in many languages, operation

dispatching in PREMO is subject to three different semantics, depending upon the mode of the operation receptor defined in the object type, which can be:

Synchronous: the caller of the operation is suspended until the operation returns;

Asynchronous: once the operation has been invoked, the caller can continue with other processing tasks while the callee carries out the operation (no return result is allowed);

Sampled: a call on an operation receptor overrides any earlier, as yet unserviced, call on that operation.

As a further complication, an object can select the set of operation receptors that it is willing to perform at a point in time; synchronous calls to non-selected operations result in the suspension of the caller. Relevant parts of the run-time state are described formally below, beginning with two definitions: *params* is a sequence of actual parameter values, and *request* represents the invocation of an operation on a specific object.

$$params == \mathrm{seq}\, non\text{-}obj$$
$$request == object \times operation$$

The run-time state defines the mode of each operation (async, sync or sampled), the objects that are suspended pending completion of a synchronous request, and, for each operation invocation (request), a bag containing calling objects and parameter values that are pending execution.

RunTime

$$ObjectSystem$$
$$mode : operation \pfun opmode$$
$$suspended : \mathbb{P}\, object$$
$$pending : request \to \mathrm{bag}(params \times object)$$

$$\forall o : object \bullet \forall p : operation \bullet$$
$$\quad mode(p) = sampled \Rightarrow count(pending(o, p)) \leq 1$$

In fact, the invariant is rather more complex than that given above, since valid states of the run-time system are determined in part by the structure of the interfaces defined by the object types, and the objects that exist within the system at any point in time. Full details can be found in [DDtH+95]. The specification of operation dispatch is spread over three schemas, one each for selection, evaluation and return, of which the third is given below:

return

$$\Delta RunTime$$
$$\Xi ObjectSystem$$
$$r : request$$

$$count(pending(r)) > 0$$
$$\mathbf{let}\, r == (opn, callee) \bullet \mathbf{let}\, pending(r) == (args, caller) \bullet$$
$$pending' = pending \oplus \{r \mapsto pending(r) \uplus [\![(args, caller)]\!]\}$$
$$mode(op) = sync \Rightarrow suspended' = suspended \setminus \{caller\}$$
$$mode(op) \neq sync \Rightarrow suspended' = suspended$$

Although it is possible using schema composition to represent the process of performing a single operation request in isolation, the specification does not address the concurrency inherent within PREMO. This is issue is taken up in Section 3.4.

The second aspect of the PREMO object model specification that is problematic concerns the relationship between the Z specification of the object model, and the Object-Z specification of 'real' PREMO object types such as *Synchronizable*. It is intended that concepts such as operation invocation used within the Object-Z specification should be interpreted in terms of the requirements set out in the Z object model specification. The approach used in [DDtH+95] attempts to link the two models explicitly but informally by using annotated names in Object-Z structures to refer to components of the object model. However the amount of detail involved even for a small object type added significant detail to the specification for little insight, and in subsequent specification activities Object-Z was used directly, with an informal understanding that the semantics of certain actions, such as invoking an operation, were subject to interpretation with respect to the PREMO object model. Again, Section 3.4 will review this issue. While on the problem of linking the two levels of specification, it should also be noted that PREMO is reflective, i.e. all PREMO objects are able to return information about their type (and their position in the object type hierarchy), and PREMO requires that its environment provide an 'object factory' that can produce an object that meets certain requirements and is an instance of a named type.

3.3. The Normative Text

The influence of PREMO's object model and organization on the structure of the specification has already been described in this section, but in fact the influence also extends in the other direction. Although the PREMO documents themselves do not contain any formal specification at this time, the editorial style used in the presentation draws heavily on the insights gained in the specification work. Data types are defined using Object-Z notation, and class definitions are given using a notation that draws heavily on Z and Object-Z. For example, here are the data types used in the definition of the *Synchronizable* object type, as they appear in the standard:

$$State == \mathbb{N}$$
$$Direction ::= Forward \mid Backward$$
$$STOPPED : State \mid STOPPED = 0$$
$$PLAY : State \mid PLAY = 1$$
$$\ldots$$

The following is how parts of the *Synchronizable* object type specified in this paper appear in the Standard. Departures from the usual presentation of Z and Object-Z (for example, the absence of a 'sidebar' for the class) are to simplify typesetting.

Synchronizable[*C*]

EnhancedPREMOObject redef (*initialize*, *initializeOnCopy*)
CallbackByName

loopCounter : $\mathbb{N}$ [Retrieve Only]
repeatFlag : *Boolean*
nloop : $\mathbb{N}$

The loop counter, repeat flag, and the number of loops of progression.

Exceptions raised (when setting the attributes):

WrongState	The object state should have been *STOPPED*.

setActionOnPair
$stateOld_{in}$: *State*
$stateNew_{in}$: *State*
$action_{in}$: *RefActionElement*
exceptions : {*WrongState*}

An action is associated with the tuple ($stateOld_{in}$, $stateNew_{in}$); see clause 7.9.1.2 on how this action element is used by the *Synchronizable* object.

Exceptions raised:

WrongState	One of the states $stateOld_{in}$ or $stateNew_{in}$ does not identify a valid state for this object instance. The exception data contains the invalid state name(s).

3.4. Shortcomings and open issues

The work described above was a 'live' exercise in specification, carried out under time constraints imposed by the schedule for progressing the PREMO standard through the phases described in Section 2 of this paper. Although we lack objective or quantifiable data, the view of those involved in both the specification and standards development processes has been that the specification work was invaluable in improving the overall quality of PREMO. It has also been a challenging technical exercise. Even though PREMO in itself does not provide all of the object types needed to build a distributed multi-media application, it is a non-trivial system whose specification has raised some important issues that are relevant to similar activities in the future.

A feature of PREMO (and a fundamental design requirement) is the pervasive use of object-oriented concepts and structures. This poses a fundamental

problem, specifically, whether or not to use an object-oriented formalism when specifying a system that is designed on object-oriented technologies. It is unlikely that the semantics of a given specification formalism will match those of the object model those technologies, and so the problem of aligning the two, described in Section 3.2, must be addressed. The approach described in this paper, of using explicit but informal conventions, or leaving the link informal, was satisfactory in the case of PREMO, as the emphasis was on developing a formal description of particular aspects of the Standard. If however we had wished to demonstrate or argue that particular properties were satisfied by the description, and those properties were contingent on the behaviour of the object model as well as specifics of certain object types, then the informal approach would have been untenable. Of course, a similar problem of integrating levels of detail holds if a non-object-oriented formalism is employed. The extra difficulty then is in relating the formal description to the normative text, and importantly, in maintaining this link as the standard evolves. Given the limited time available for specification activities, it is desirable that localised changes to the base document have localised effects on the specification. Of course, no formalism will simplify the problem of wholesale changes to the base document.

Two approaches seem to hold out promise for controlling the complexity of a specification like that of PREMO, allowing specific sections of material to be captured to the level of detail needed for a standard and also providing a coherent integrated view of the whole. The first is the development of frameworks for integrating partial specifications, possibly written in different formalisms; [ZJ93] is an early example of this direction, and Clarke and Wing [CW96] identify the integration of methods as one of the key areas for future work in formal methods. The second approach is to use a specification logic that supports the development of a 'layered' model, where it is possible to use the formalism first to describe the object model, and then to build larger structures from the primitives specified in the foundation. Lamport's ongoing specification of a 'Threads' API in TLA+ [Lam96] is a good illustration of this style. It can be argued that a similar style can be adopted with any specification formalism, for example Z or Object-Z. The real problem lies though in the overhead involved in encoding or representing higher level structures using the lower level primitives in the specification. For example, how does the amount of work needed to encode an object model in Z, and then represent an object type like *Synchronizable* in terms of the *representation* of the object model, compare with the cost of describing the object type directly within an object-oriented formalism? For application areas like standards, where the emphasis is on description rather than proof, the layered model will need to have a close correspondence to the object-oriented description if the trade-off between formality and ease of description is to favour the former.

Apart from the object model, two other aspects of PREMO create a tension in the choice of appropriate description technique(s), specifically concurrency and real-time concerns. With the exception of one sub-tree, PREMO objects are conceptually active and are expected to engage in concurrent activities, leading to the general concerns of communication, and the potential for deadlock. Process algebras such as CCS [Mil89], CSP [Hoa85] are usually considered the appropriate tools for dealing with such concerns. In the case of PREMO, there is a challenge in representing objects such as Synchronizable as processes in that the allowable behaviour of these objects is determined by a comparatively complex state space, something which process-oriented techniques are not primarily

intended to address. One of the authors (GF) has however been involved in the development of a technique for mapping PREMO objects into LOTOS that has been used successfully with the Synchronizable object type to yield further valuable insight into the behaviour of these objects [FM97]. There is already a good body of existing work on specifying particular aspects of multi-media systems, such as Quality of Service, using these techniques, for example [BBBC95]. However, the success of this work lies in a level of abstraction that would make it difficult to relate to the details of a standard like PREMO, and the problem of handling the large state spaces that occur in the Standard would seem to apply here as well. Time is another area where there is room for further improvement in the specification. It is not currently possible to specify, for example, quality of service requirements, other than through the naive use of values that are intended to denote time units relative to some clock. This is one context where an integrated method, supporting links between a state-oriented style of description, and one based on real-time temporal operators (such as the already-mentioned work of Blair et al [BBBC95, BBS97]) could be useful.

Although the mathematical foundations of formal description techniques are important, there are also some mundane but practical problems that represent a non-trivial hurdle to would-be specifiers in the standards community. Work in standards draws volunteer effort from a wide range of occupations, ranging from large multi-national corporations to research institutes and universities. By and large, they have neither the time or motivation to learn the use of arcane document formatting systems needed to typeset specifications in languages such as Z and Object-Z. Increasingly also, the World Wide Web is becoming the medium of choice for making standards available to the wider community, particularly with the ability, using HTML, to place cross-reference links throughout the document. Even if there is progress towards extensions to HTML that allow the detailed formatting of mathematical text required for specifications, problems such as the availability of suitable character sets will still represent significant hurdles to the use of such languages for widespread use. Although it is possible to write, for example, Z, using standard character sets, potential readers of the specification are then confronted by a representation which is very different from that which appears in the available texts or in most published papers. In this respect, the approach taken by the developers of PVS [ORSvH95], where the specification is by default presented using the standard ASCII character set, has certain practical advantages. These comments apply both now, when specification development is largely independent of the normative text, and in the future, where it is hoped that formal descriptions may be incorporated into the normative text. The view of the PREMO document editors (DJD and IH) is that, if they were to start work on such a document today, the benefits of making the document accessible over the web would over-ride the desire to keep the structure of the specification coupled to that of the normative text. Consequently, Object-Z would not be used as the framework for documenting the functional provisions, as described in section 3.3.

4. Opportunities and Challenges

The ISO/IEC Directives make provision for the introduction of formal descriptions into standards. The Directives identify three phases for the evolutionary introduction of formal descriptions. The first phase is the use of formal descrip-

tion by the committee which is defining a standard. This phase recognizes that there may not be sufficient resources within national bodies to either write or review formal descriptions. Development of standards should be based on conventional natural language approaches, leading to a standard for which the natural language specification is the definitive standard. Where formal descriptions of significant parts of standards are produced, Committees are encouraged to preserve the work through publication as an ISO Technical Report.

In the second phase, a formal description of the standard forms an annex to the natural language description of the standard. The natural language description constitutes the provisions of the standard, the annex being for information. In this stage, knowledge and experience of formal descriptions is more widely available, resources are available to produce formal descriptions, but it cannot be assumed that enough national bodies can review formal descriptions in order to cast votes in a ballot on a formally described standard.

In the third phase, the formal description constitutes the provisions of the standard and is accompanied by a natural language commentary. The formal description and the natural language description are given equal weight in the event of discrepancies.

The ISO/IEC Directives also stipulate that justification has to be given for the use of any formal description technique that is either not already standardized or is in the process of standardization.

¿From our experiences of using formal methods in the development of PREMO, we draw some general conclusions.

1. Education. It is still the case that many practising Information Technology professionals are uncomfortable with formal descriptions. The expertise to read and comment upon formal descriptions is not widespread in the area of standardization in which we work. Experts are comfortable reviewing a very complex natural language text, but are daunted by the prospect of reviewing even a modest formal text. Although it may not be difficult to train people to read formal descriptions[Hal90], it is difficult to find the resources to do this specifically within the context of an ISO/IEC standardization activity.
2. Tools. Again in our area of standardization, many people see a good tool base as vitally important to the acceptance of formal descriptions. We would include in this tools to manage the generation of a formal description, tools to check syntactic and type correctness and tools to exercise the formal text in order to build confidence that the formal behaviour is indeed the intended behaviour. The typesetting issues raised in section 3.4 are also important in the context of tool support. There is a need for a representation of the formal description that can both serve as a basis for typeset documentation and for manipulation by type checkers, proof, tools, etc.
3. Standardization of formal description techniques. The requirement to justify the use of a non-standard formal description technique in the presentation of an ISO/IEC standard can be a stumbling block. It is clear why this is a rather strong requirement, for example, one would not wish the meaning of a standard to change because of some change made to the semantics of the notation in which the standard is specified. If the notation is itself the subject of an ISO/IEC standard, then there are at least mechanisms in place for 'version control'. However, it is vitally important to have access to a range of formal description techniques in order to be able to work with the notation

most appropriate to the task, and if necessary, to extend the notation within the context of the standard for which it is being used.

Having made these points, however, we do firmly believe from our experience that formal methods have much to offer in the standardization arena.

1. As Hall[Hal90] has said, '[Formal methods] work largely by making you think very hard about the system you propose to build'. This is entirely bourne out in our experience. Any technique that causes one to ask new questions about the standard under development is potentially beneficial in the fight against inconsistency and ambiguity.
2. Rushby[Rus93] has argued that formal methods seem to find their most effective application early in the life-cycle for a number of reasoning, including the view that 'formal methods provide a repertoire of mental building blocks that assist and encourage the development of specifications that are precise yet abstract'. That view also is bourne out in our experience. The work we have done was carried out at an early stage in the PREMO development process, when often there were many details undefined. The key details of the proposed synchronization mechanisms in PREMO were captured at an early stage at an appropriate level of abstraction. It was important to be able to abstract away from inessential detail, for example by techniques such as the use of generic abstract types. It is also important in the context of standardization to be able to leave some choices to implementors. The consensus forming process can often lead to the outcome 'implementation dependent'. This outcome can be captured in a specification in a satisfactory manner.

The questions posed at the end of section 2.1 invite a response from the formal methods community. We argue that if formal description techniques have been used during the development of a PAS, then ISO/IEC is in a stronger position to assess the answers to these questions than would otherwise be the case.

1. How well are all interfaces specified? For a formal description, there are levels of consistency checking that can, and should, be applied automatically.
2. How easily can implementation take place without need of additional descriptions? The level of abstraction in the formal text and checks of properties that have been performed on the text is one perspective from which to answer this question. We would argue that this is still a useful aspect of the submission to explore, even if the specification work only addresses some aspects of the system, or models only some parts of the system.
3. What proof exists for successful implementations? In some areas of ad hoc standardization, reference implementations are considered essential before the standard itself is promulgated. Formal description may offer the prospect of proving evidence of implementability at lower cost than constructing a reference implementation.
4. What means are used to provide definitive descriptions beyond straight text? The formal description itself.

Acknowledgements

The work described in this paper has been carried out under the auspices of the ERCIM Computer Graphics Network, with financial support from the CEC Human Capital and Mobility Programme (contract CHRX-CT93-0085).

References

[AD90] D. B. Arnold and D. A. Duce. *ISO Standards for Computer Graphics - The First Generation.* Butterworths, 1990.

[BBBC95] L. Blair, G. Blair, H. Bowman, and A. Chetwynd. Formal specification and verification of multimedia systems in distributed open processing. *Computer Standards & Interfaces*, 17:413–436, 1995.

[BBS97] G. 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.

[BE95] T. Bryant and A. Evans. Formalizing the Object Management Group's core object model. *Computer Standards & Interfaces*, 17:481–489, 1995.

[Buf94] J.F. Koegel Buford. Architecture and issues for distributed multimedia systems. In *Multimedia Systems.* ACM Press/Addison Wesley, 1994.

[CDD+90] 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.

[CW96] E.M. Clarke and J.M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[DDHF97] D.J. Duke, D.A. Duce, I. Herman, and G. Faconti. Specifying the PREMO synchronization objects. Technical Report 02/97-R048, European Research Consortium for Informatics and Mathematics (ERCIM), 1997. http://www-ercim.inria.fr/publication/technical_reports/.

[DDtH+94] D. A. Duce, D. J. Duke, P. J. W. ten Hagen, I. Herman, and G. J. Reynolds. PREMO - an initial approach to a formal definition. *Computer Graphics Forum*, 13(3):C–393 – C–406, 1994.

[DDtH+95] 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 & Interfaces*, 17:491 – 509, 1995.

[DKRS91] R.W. 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.

[DRS95] 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.

[Duk95] D.J. Duke. Time and synchronisation in PREMO: A formal specification of the NNI proposal. Technical Report OME-116, ISO/IEC JTC1 SC24/WG6, 1995. ftp://ftp.cwi.nl/pub/premo/RapporteurGroup/Miscellaneous/OME-116.ps.gz.

[FM97] G. Faconti and M. Massink. Investigating the Behaviour of PREMO Synchronization Objects. In *Proceedings 4th Eurographics Workshop on Design, Specification and Verification of Interactive Systems.* Springer, 1997. To appear.

[GT95] S.J. Gibbs and D.C. Tsichritzis. *Multimedia Programming.* ACM Press/Addison-Wesley, 1995.

[Hal90] A. Hall. Seven Myths of Formal Methods. *IEEE Software*, 7(5):11 – 19, 1990.

[HCD+94] I. Herman, G.S. Carson, J. Davy, P.J.W. ten Hagen, D.A. Duce, W.T. Hewitt, K. Kansy, B.J. Lurvey, R. Puk, G.J. Reynolds, and H. Stenzel. Premo: An ISO standard for a presentation environment for multimedia objects. In D. Ferrari, editor, *Proceedings of the Second ACM International Conference on Multimedia (MM'94).* ACM Press, 1994.

[HCD+97] I. Herman, N. Correia, D.A. Duce, D.J. Duke, G.J. Reynolds, and J. Van Loo. A standard model for multimedia synchronization: PREMO synchronization objects. *Multimedia Systems*, 1997. To appear.

[HJ95] I.S.C. Houston and M.B. Josephs. A formal description of the OMG's Core Object Model and the meaning of compatible extension. *Computer Standards & Interfaces*, 17:553–558, 1995.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Series in Computer Science. Prentice Hall International, 1985.

[HRL96] I. Herman, G.J. Reynolds, and J. Van Loo. PREMO: An emerging standard for multimedia. Part I: Overview and framework. *IEEE MultiMedia*, 3:83–89, 1996.

[ISO85] *Directives for the Technical Work of ISO Committees.* ISO Central Secretariat, Geneva, Switzerland, second edition, 1985.

[ISO95a] *ISO/IEC Directives, Procedures for the technical work of ISO/IEC JTC1 on Information Technology.* ISO Central Secretariat, Geneva, Switzerland, third edition, 1995. Available through http://www.iso.ch/.

[ISO95b] The Transportation of Publicly Available Specifications into International Standards - A Management Guide. Document ISO/IEC JTC1 N3279, 1995. Draft details of the process are also available through http://www.iso.ch/.

[ISO96a] Information processing systems - Computer graphics and image processing - Presentation Environments for Multimedia Objects (PREMO). Draft International Standard ISO/IEC 14478, 1996.

[ISO96b] Information processing systems - Computer graphics and image processing - Virtual Reality Modelling Language (VRML). Draft International Standard ISO/IEC 14722, ISO/IEC Central Secretariat, 1996.

[ISO96c] Introduction to ISO. http://www.iso/ch, 1996.

[ISO97] Information technology - Open Distributed Processing - ODP trading function - Part 1: Specification. International Standard ISO/IEC 13235-1, 1997.

[Lam96] L. Lamport. The Windows Win32 Threads API Specification. Public draft, http://www.research.digital.com/SRC/personal/Leslie_Lamport/tla/threads/threads.html, 1996.

[LHHR94] N.G. Leveson, M.P.E. Heimdahl, H. Hildreth, and J.D. Reese. Requirements specification for process control systems. *IEEE Transactions on Software Engineering*, 20(9):684–707, 1994.

[Mil89] R. Milner. *Communication and Concurrency*. Series in Computer Science. Prentice Hall International, 1989.

[MS87] P.M. Melliar-Smith. Extending interval logic to real time systems. In *Temporal Logic in Specification*, number 398 in Lecture Notes in Computer Science, pages 224–242. Springer-Verlag, 1987.

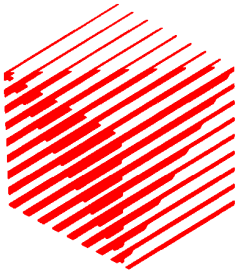
[ORSvH95] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.

[RDD94] 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, International Organization for Standardization, 1994.

[Rus93] J. Rushby. Formal Methods and the Certification of Critical Systems. Technical Report SRI-CSL-93-07, SRI International, Computer Science Laboratory, 1993.

[Spi92] J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, second edition, 1992.

[ZJ93] P. Zave and M. Jackson. Conjunction as composition. *ACM Transactions on Software Engineering and Methodology*, 2(4):379–411, 1993.



The European Research Consortium for Informatics and Mathematics (ERCIM) is an organisation dedicated to the advancement of European research and development, in the areas of information technology and applied mathematics. Through the definition of common scientific goals and strategies, its national member institutions aim to foster collaborative work within the European research community and to increase co-operation with European industry. To further these objectives, ERCIM organises joint technical Workshops and Advanced Courses, sponsors a Fellowship Programme for talented young researchers, undertakes joint strategic projects, and publishes workshop, research and strategic reports as well as a newsletter.

ERCIM presently consists of thirteen research organisations from as many countries:

Central Laboratory of the Research Councils

Rutherford Appleton Laboratory
Chilton, Didcot
GB-Oxon OX11 0QX

Tel: +44 123582 1900
Fax: +44 1235 44 5385
http://www.cis.rl.ac.uk/

Centrum voor Wiskunde en Informatica

Kruislaan 413
NL-1098 SJ
Amsterdam

Tel: +31 205929333
Fax: +31 20 592 4199
http://www.cwi.nl/

Consiglio Nazionale delle Ricerche

IEI-CNR
Via S. Maria, 46
I-56126 Pisa

Tel: +39 50 593 433
Fax: +39 50 554 342
http://bibarea.area.pi.cnr.it/ERCIM/welcome.html

Czech Research Consortium for Informatics and Mathematics

FI MU
Botanicka 68a
602 00 Brno

Tel: +34 3 401 72 89
Fax: +34 3 401 62 10
http://www.utia.cas.cz/CRCIM/home.html

Danish Consortium forInformation Technology

CIT
Forskerparken
Gustav Wieds Vej 10
8000 Århus C

Tel: +45 8942 2440
Fax: +45 8942 2443

Foundation of Research and Technology – Hellas

Institute of Computer Science
P.O. Box 1385
GR-71110 Heraklion, Crete

Tel: +30 81 39 16 00
Fax: +30 81 39 16 01
http://www.ics.forth.gr/

GMD – Forschungszentrum Informationstechnik GmbH

Schloß Birlinghoven
D-53754 Sankt Augustin

Tel: +49 2241 14 0
Fax: +49 2241 14 2889
http://www.gmd.de/

Institut National de Recherche en Informatique et en Automatique

B.P. 105
F-78153 Le Chesnay

Tel: +33 1 39 63 5511
Fax: +33 1 39 63 5330
http://www.inria.fr/

Instituto de Engenharia de Sistemas e Computadores

Rua Alves Redol 9
Apartado 13069
P-1000 Lisboa

Tel: +351 1 310 00 00
Fax: +351 1 52 58 43
http://www.inesc.pt/

Swedish Institute of ComputerScience

Box 1263
S-164 28 Kista

Tel: +46 8 752 1500
Fax: +46 8 751 7230
http://www.sics.se/

Schweizerische Gesellschaft zur Förderung der Informatik und ihrer Anwendungen

Math. Dept.,
ETH-Zentrum
CH-8092 Zürich

Tel: +41 1 632 22 25
Fax: +41 1 632 10 85
http://www-dbs.inf.ethz.ch /sgfi/

Stiftelsen forIndustriell og Teknisk Forskning ved Norges Tekniske Høgskole

SINTEF Telecom & Informatics
N-7034 Trondheim

Tel :+47 73 59 30 00
Fax :+47 73 59 43 02
http://www.informatics.sintef.no/

Magyar Tudományos Akadémia – Számítástechnikai és Automatizálási Kutató Intézete

P.O. Box 63
H-1518 Budapest

Tel: + 361 166 5644
Fax: + 361 166 7503
http://www.sztaki.hu/

Technical Research Centre of Finland

VTTInformation Technology
P.O. Box 1200
FIN-02044 VTT

Tel:+358 9 456 6041
Fax :+358 9 456 6027
http://www.vtt.fi/

Central Office:
ERCIM
Domaine de Voluceau, Rocquencourt, B.P. 105, F-78153 Le Chesnay Cedex, FRANCE
Tel: +33 1 39 63 53 03 Fax: +33 1 39 63 58 88
http://www-ercim.inria.fr/