# STEP

## An Introduction

Jon Owen

INFORMATION GEOMETERS

**PRODUCT DATA ENGINEERING**

*Series Editor* Jon Owen

*Jon Owen* has been active in the development of STEP (Standard for the Exchange of Product Model Data) since 1985. He chaired Committee SG4 from 1987, and became Convener of WG6 when it was created in 1990. Jon is one of the seven original members of the European Commission's Product Data Technology Advisory Group (PDTAG).

Jon did his doctorate at the University of Leeds, and then spent two years as a lecturer in the Computer Studies Department and five years in the Geometric Modelling Project. He played a significant role in founding the CAD-CAM Data Exchange Technical Centre (CADDETC) at Leeds in 1986. He led the R&D activities at CADDETC until 1992, when he returned to the Mechanical Engineering Department as a Senior Fellow.

# STEP

## An Introduction

Jon Owen

INFORMATION GEOMETERS

First published December 1993

Information Geometers Ltd
47 Stockers Avenue
Winchester
SO22 5LB
UK



ISBN 1-874728-04-6

British Library Cataloguing-in-Publication Data.
A catalogue record for this book is available
from the British Library.

Typeset by the author.

Printed in Great Britain by
Bookcraft (Bath) Ltd
Wheelers Hill
Midsomer Norton
BA3 2BX

*La distance n'y fait rien; il n'y a que le premier pas qui coûte.*
*(The distance is nothing; it is only the first step that is difficult.)*
Mme du Deffand (1697-1780); letter to d'Alembert, 7th July 1763.

The early chapters in this text introduce the reasons why an enterprise may wish to exchange product data, and the problems encountered with the standards and specifications available in the 1980s. These standards are not described, but references to them can be found in the Bibliography. The requirements for a new standard which addresses the problems are discussed, followed by a brief description of the conception and development of STEP. (The process by which ISO produces a standard and a detailed history of STEP ballots is not included in this chapter, but in appendices for reference.)

The material in these early chapters provides the disposition for the structure of STEP, in a number of classes. These classes are characterized in the following chapters, noting the titles of the parts in each class, and the relationships between the classes. Each class is then described in turn. The text concludes by outlining possible future developments in STEP, how it is likely to be used by industry, and the benefits which would be gained by its use. Abbreviations and details of standards bodies are also included as appendices.

It should be stressed that this text is an overview: more detailed explanations of the STEP parts can be found in other texts in this series.

# Contents

# List of Figures

# Foreword

This book is the first in a series which describes the international standard for the exchange of product model data (ISO 10303), known as STEP. Each volume provides a commentary on an individual part or collection of parts of ISO 10303, thus fulfilling the role of an accompanying text rather than a replacement for or an alternative to the standard itself.

This series will be useful to managers who need to introduce STEP into their operations, to users of STEP, to people writing STEP-based software, and to teachers of courses on product data standards and their students.

This particular book is a companion to ISO 10303-1 ('Overview and fundamental principles'). It places STEP in the context of other product data exchange specifications and standards, describes the structure of the entire standard and sets down its history, which has resulted in that structure. The majority of the text comprises a description of each of the classes of parts and the relationships between them. The text also indicates how STEP is likely to be used and the benefits it will provide to those who adopt it, either directly or indirectly.

ISO 10303 was originally conceived as the standard for the exchange of product model data. However, it came to be known by the acronym "STEP", and it is by this to which it is usually referred. I have continued the practice in this text for the simple reasons that "STEP" is both more familiar and easier to read than either "ISO 10303" or "the standard for the exchange of product model data".

The pronoun 'he' should be read as 'he or she' throughout.

A distinction has been made between standards and specifications. Whilst the latter may be *de facto* standards, they have not

undergone the national or international review which would make them into standards.

There are many examples included in the text, particularly of EXPRESS constructs. Whilst many of the entity names can be found in STEP, the resemblance may end there; the explicit attributes chosen for this text illustrate a particular feature, and may deliberately highlight poor EXPRESS style. There are many instances of a Cartesian point in this text, and none of them reflect the consensus which resulted in the entity which appears in Part 42.

The history of STEP is provided as an appendix so that readers can perceive the evolution of STEP into the infrastructure it now provides. Summaries of voting and the status of the parts will necesarily become out of date, but they are included as a snapshot.

It is difficult to judge exactly when to write a text about a standard. If it is written when the standard is already available, then the text is not timely; readers require a 'user friendly' introduction but there isn't one. Conversely, if the text is produced before the standard, it runs the risk of describing something that might change, perhaps substantially. At best it is redundant and at worst it is disinformation.

I have tried (as have others before me) to time the writing of such a book correctly. Many of the STEP parts have been balloted as Committee Drafts, the issues from the national bodies have been incorporated and an integrated set of Draft International Standards has been released for full international review; the time appears to be as apposite as it can be. Further, many of the concepts described here have been fixed for several years. For example, while EXPRESS was agreed as a DIS in 1991, many of its basic facilities have been stable since 1986; what has changed since then is that it is now much better understood and more widely accepted, and a vast improvement has been made in its documentation due to the wide review received.

However, some details will necessarily change in STEP after the publication of this text. Consequently, I would first apologize to the reader (but seek his absolution, given that he is now aware of the timing problem). Secondly, I would reiterate the statement that these texts are an adjunct to the standard rather than a replacement

for it. If you are going to use or implement STEP in earnest, then you need to acquire the official documents. My hope is that the texts in this series, and this volume in particular, will provide the relevant background and preparation.

## Acknowledgements

The writing of this series and this book in particular would not have been possible without the encouragement and technical help from many groups of people. I would like to acknowledge many friends and colleagues.

- Members of the STEP working groups, particularly the conveners.
- My colleagues in Working Group Six, who have helped to ensure that conformance testing has the high profile it enjoys.
- Members of the parent subcommittee, SC4, and especially its convener Brad Smith.
- The national representatives on the PMAG, and especially its convener (until June 1993) Jerry Weiss.
- Past and present colleagues at the CAD-CAM Data Exchange Technical Centre (CADDETC) in the University of Leeds.
- Colleagues in collaborative European research and development contracts.
- Members of the European Commission's Product Data Technology Advisory Group (PDTAG), and especially its chairman Professor Horst Nowacki.
- Members of the UK BSI Committee AMT/4, and especially its chairman Howard Mason, which has provided expertise to STEP since its inception in 1985.
- Colleagues at the University of Leeds, particularly Professor Alan de Pennington and Dr Susan Bloor, who have created the thriving atmosphere for research in computer-aided engineering in which this book was conceived and written.

Figures from ISO 10303 are reproduced with the permission of the International Organization for Standardization. The complete standards can be obtained from the ISO Central Secretariat or from national standards bodies, details of which are in Appendix D.

This text has been prepared using LaTeX [18].

# 1

# The background to STEP

## Requirements

Many engineering enterprises have a requirement to exchange data concerning their products in computer-readable form. This not only enables internal and external communication (both within an organization and with clients, contractors, subcontractors, suppliers and partners) but also makes the engineering data generated by one application program readable by other application programs. This increases the aid (or automation) provided by the computer, as the 'A' in CAD, CAM or even CAE[1], which helps to reduce costs or increase the effectiveness of the enterprise.

There is often an additional requirement for long-term archiving, which is the same problem over a longer period. The complexity is increased in this case because the scope of the application program which will eventually read the data is not known at the time that the archive is written.

## General problems in data exchange

There are several areas in which problems may occur when attempting to exchange product data in computer-readable form. The most common is difference in system functionality: two CAE systems may have different domains (mechanical, electrical ...) or dimensionality (two- or three-dimensional). Even if both are three-dimensional systems, there is still potential for many differences. They may be wireframe, surface or solid; if they are both solid then either

[1]In the rest of this text, the acronym CAE is used to encompass all computer-aided activities in an engineering enterprise.

constructive solid geometry or boundary representation are possibilities; and the latter may allow facetted, analytic or sculptured surfaces.

Even if two CAE systems are compatible, problems with media often arise: with utilities to read and write magnetic tapes, with network protocols or simply with the physical form of the media. Receiving a disk or cartridge of the wrong size is depressingly common. Drawing-office conventions can also cause problems; enterprises use different layering or coordinate systems, for example. All of these potential problems need to be addressed before any data is exchanged, irrespective of the strategy chosen to do so.

## Strategies for data exchange

There are two strategies available for enterprises wishing to exchange product data in computer-readable form. They may use:

1. Proprietary direct translators.

2. A public domain neutral intermediate format.

For the first strategy, a single computer program is written which will convert the data produced by one CAE system (the sender) into that required by another (the receiver). If it is required to return the data from the second system to the first, then the 'inverse' program also needs to be written. It invariably *is* the case that such a requirement exists. Each pair of programs can be optimized to suit each pair of systems.

Alternatively, a neutral intermediate format may be used, with each CAE system vendor providing a pre-processor for writing the neutral file and a post-processor for reading from the neutral file, as shown in Figure 1.1.

## Problems with direct translators

In general, where data exchange takes place between $n$ systems, $n(n-1)$ programs are required for the first strategy and $2n$ programs for the second; consequently, if more than three systems are involved, the neutral intermediate format approach requires fewer programs. If another CAE system is added to an existing scenario,

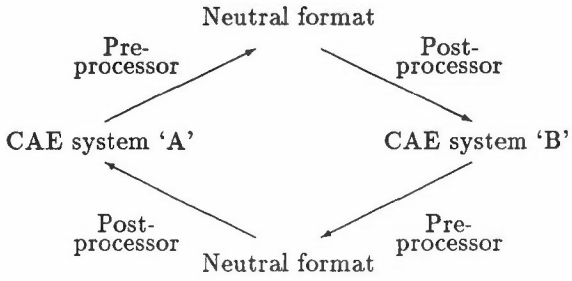



Figure 1.1. Data exchange using an intermediate neutral format.

then $2n$ additional programs are required for the first strategy, but only two for the second.

As well as the cost of maintaining a large number of programs with the direct translator approach, detailed knowledge is needed of *both* proprietary formats between which the program is converting. Each new release of each system from each vendor may require a rewrite of many programs, and the onus is not necessarily on the vendor to do this maintenance.

## Current standards and specifications for neutral file formats

There are several standards and specifications for neutral file formats in use today:

- IGES [15].
- VDA-FS [40, 41].
- SET [35].
- VDA-PS [43].
- VDA-IS [42].
- EDIF [7].

A number of vendor-defined specifications are also used for data transfer, such as AutoCAD's DXF, Intergraph's ISIF and IBM's IIF

file formats. This book does not provide an overview of these standards and specifications; there are several descriptions available of the facilities in each [36, 19, 28, 30, 22]. However, the distinction needs to be made between this set of data exchange specifications and those which address graphics, such as:

- GKS [8].
- CGM [6].
- PHIGS [25].

This is best illustrated by an example. Consider a dimension line on an engineering drawing, which comprises two arrows (leaders), two witness lines and some text, probably including both a dimension and a tolerance. In product data exchange standards, it is represented as precisely that, and may even be tied to the geometry which it is dimensioning. It can be manipulated as a composite entity. However, in a graphics standard, it is likely to have been decomposed into many entities: polylines for the arrowheads and tail segments, and so on. Whilst it is still interpreted by a human reader as a linear dimension when it is plotted on an engineering drawing, it has lost its 'linear dimension-ness', and cannot be processed by the computer as such. In short, the levels of information exchanged by product data and graphics standards are different.

### Problems with neutral formats

Although there are problems with the strategy of using direct translators, there are problems with neutral formats too. It takes time to develop them, as it commonly involves volunteer effort and a democratic process. This usually leads to retrospective standards; by the time the standard has been published, there are new facilities available in CAE systems for which there is no provision in the standard. There are also limitations in coverage; it will not be possible for *every* feature available in *all* CAE systems to be captured by the standard. If the standard is poor or ambiguous, then it will lead to misinterpretations. The neutral-format approach is not as efficient as using direct translators, because two translations occur with each transfer rather than one; consequently, not only is there a loss of

efficiency, but double the opportunity for the introduction of errors and loss of information.

A standard is usually written in natural language, not in a formal specification language, nor is it based on information modelling methodologies. There is no clear distinction between logical, application and physical specifications. There is a lack of conformance clauses and there are insufficient independent accredited testing laboratories, which means that users are reliant on the claims of vendors for the quality of their processors. Often, no subsets were defined by the standard, meaning that vendors can legitimately make an *ad hoc* choice of which constructs to support; consequently, there is almost invariably a mismatch in the coverage of two processors being used in a data transfer.

All of these problems are discussed further in later chapters in this book.

## Solutions

Despite the problems described above, solutions can be and have been put in place by enterprises wishing to exchange product data in computer-readable form [3, 4, 2].

The first set of solutions are concerned with co-operation between the two enterprises. Many problems have arisen from the arrival of a magnetic tape—often unlabelled—with little previous communication. A clear statement of the objectives and of the level of success required should be available *before* production use of exchange begins, and will enable the enterprises to determine whether the exchange has been successful. A feasibility study, in which typical data are exchanged, will also highlight any problems before the real exchanges start. Harmonization of drawing-office practice and a clear statement of hardware practices will also help.

Conformance testing will ensure that a processor at least conforms to the standard (or subsets of it); tests are more credible when they are undertaken by an accredited conformance testing laboratory which is independent of the software vendor. The results, which will save vendors and users undertaking many of their own trials, can be used as a prerequisite to user acceptance testing, including robustness, performance and interoperability tests.

As part of a feasibility study, a transfer forecast could be made, based upon vendor claims or the results of conformance testing, or both. This enables any constructs in the sending system which will not transfer to the receiver to be identified. Their use can then be disallowed or modified, or some special action taken to ensure their transfer. Once a neutral format file has been produced by the sending system—again, preferably in the feasibility study—it can be submitted for syntax and semantic checking. This highlights any errors originating in the sending system's pre-processor. Such errors can then be corrected: either by that vendor, or by the use of editing or flavouring software, which ensures that the data in the neutral format file is both correct and processable by the receiving system's post-processor.

It may be that there is an ambiguity or even an error in the standard itself. Mechanisms do exist to make corrections; ISO has rapid amendment procedures as well as an in-built review cycle. IGES has *Requests For Change* which, following review and debate, generate *Edit Change Orders* which define and document the changes between one version of the specification and the next. Recommended practices for IGES implementors also provide clarification [1].

## Conclusion

All of the existing standards and specifications mentioned above are eminently workable, and have been used to exchange product data between enterprises as a matter of course. However, as will be shown in the next chapter, some of the problems in their use arise as a result of not using formal methodologies in their development and documentation.

When work on STEP started, many lessons had already been learned from the development, publication and use of these earlier standards, which were starting to proliferate [46]. A number of research and development activities had also contributed significantly to the area. As well as the standards and specifications mentioned already, the following were also relevant to the genesis of STEP:

- CAD*I [33, 17][2].

[2]The CAD*I reports provide further detail: [31, 38, 32, 34, 29, 10, 11].

- PDDI [24].
- PDES [37].
- XBF [51].
- CAD-LIB [27].
- MIL-D-28000A [20].

It was felt that a single international standard which built on the experience that had been gained would benefit the field of computer-aided engineering considerably.

# 2

# The requirements of STEP

## Overview

STEP is an ISO activity to develop a new engineering product data exchange standard, which will be documented as ISO 10303. Rather than have many separate national standards, the aim is to produce a single and better standard, to cover all aspects of product life cycle in all industries. The life cycle is particularly long when one considers the initial survey of a green-field site, with the consequent storage of geographical and geophysical data, to the ultimate decommissioning (several decades later) of the building sited there.

Although some provision for the exchange of engineering drawings would necessarily be included, STEP was *not* intended to be a 'graphics' standard: this field was already being addressed within ISO [8, 6, 9, 25, 5, 26].

## Design goals

The design goals of STEP were established at the start of its development [45]; the major ones are summarized here:

*Completeness:* STEP should allow a complete representation of a product, for both exchange and archiving.

*Extensibility:* with such a wide scope, STEP should provide a framework into which extensions of domain can be built.

*Testability of additions:* before any addition is made to the standard (and this includes everything in the initial release), it should be subjected to peer review and, if possible, undergo further testing by being implemented.

*Efficiency:* STEP should be efficient in terms of both file size and the computer resources needed for processing (this requirement is now less pressing because of the substantial reduction in hardware costs since 1986).

*Compatibility with other standards:* STEP should be compatible with other standards, as far as possible, in order to ease migration from existing standards. It should also use facilities in other standards where practicable (such as those for graphics, character sets and terminology).

*Minimal redundancy:* there should be only one way of representing a particular concept.

*Computing environment independence:* STEP should be independent of particular hardware and software.

*Logical classification of data elements:* STEP should define (standard) subsets for implementations as it would clearly be a large standard.

*Implementation validation:* a framework for conformance testing should be part of the standard, in order to allow conformance testing services to be available on publication of STEP.

One of the main differences between STEP and other standardization activities has been and is that it is forward-looking, not retrospective. Many of those involved in its development are from research and development projects; others are from CAE system developers; the ideas being discussed for STEP are often not yet available in a commercial CAE system, but are being implemented as the standard itself is being developed. Consequently, when STEP is published, the CAE systems with those facilities will be coming on to the market, and so will be compatible.

In order to be adopted, STEP has to be better than today's solutions: learning from existing standards, it will supersede them. However, in order to do this, it also needs to provide migration from existing standards.

## Enabling technologies

Among the lessons learned from the existing initiatives was that there was going to be a number of enabling technologies which would be of critical importance and value in developing STEP. These would all help to avoid the ambiguities of current standards.

A three-layer architecture was adopted. This was derived from the ANSI/SPARC committee report on DBMS architectures [39], which defined the three levels as:

*External:* the information (subset) relevant to a specific application.

*Conceptual:* the information that describes all domains of interest.

*Internal:* the implementation of the conceptual schema as a computer file system.

This separation enables multiple application views and implementations to be defined. In STEP, the layers were defined originally as application, logical and physical. Further, information modelling methods were adopted [47, 48] to define conceptual and application reference models, and a formally defined data specification language (EXPRESS) was used for the specification of the logical and application models. Equally, a formal specification of the implementation method (including the mapping to the physical layer) was to be provided. Finally, conformance requirements and test purposes were to be specified as part of the standard to enable processor testing.

As a result of these approaches, there are two important types of information models in STEP, which correspond to two of the layers in ANSI/SPARC:

1. *Resource information models (conceptual):* these provide information in a well-defined *generic*, or context-independent, domain such as units, geometry, topology, shape, product structure and configuration management. They are an integrated *resource* which can be used by several applications.

2. *Application protocols (external):* application protocols provide information in a specific *application* domain, such as explicit and

associative draughting, boundary and surface geometry for mechanical design, and configuration-controlled three-dimensional design. These include conformance requirements, and provide the basis for the implementation of processors to be used for industrial data exchange.

The STEP implementation methods correspond to the third layer in the ANSI/SPARC model.

This chapter has concentrated on the requirements of STEP and how the problems experienced with the existing product data exchange standards and specifications were to be avoided. The next chapter describes how these requirements and STEP itself evolved in the international arena.

# 3

# The genesis of STEP

In 1984, deficiencies in the existing generation of product data standards and specifications had been identified and were well known. Various national initiatives had produced interim or alternative solutions to these problems. However, there was a danger that the specifications for effecting product data exchange would proliferate, and themselves cause a fresh set of problems. With this in mind, representatives of the various national initiatives attended the inaugural ISO TC184/SC4 meeting in Washington DC, in July 1984, with a view to producing a single international standard. As well as providing mutual education on the initiatives, the following resolution was passed:

> "SC4 recognises the need for a new standard for the external representation of product model data. This standard will be based upon existing data exchange initiatives including the US IGES and PDDI, the French SET, the German VDA/VDMA-FS, and the UK NEDO.
>
> Technical work will be accomplished by existing and future national projects, organizations, and resources which will be coordinated and monitored by the SC4 committee. SC4 will set design objectives, establish priorities, arbitrate differences, and ensure that objectives are met and consistency is maintained."
>
> *[Resolution 1 (July 1984, Washington)]*

It soon became clear that it would take some time to develop such a single standard. Before it would become available, there was a need for the existing national standards to have their coverage

increased, to meet the immediate needs of their industries. It was suggested that effort be channelled into one of these standards so that it could become an interim international solution.

However, the consensus was that none of the existing initiatives was acceptable, and that trying to fix something for an interim solution would almost certainly take longer than starting afresh. Even so, it was acknowledged that enhancements to existing standards and specifications would continue, and that it would be sensible for such developments to be undertaken simultaneously. It was also at this time that the acronym "STEP" was coined, and that migration from the present generation of standards to STEP was recognized as a prerequisite for STEP itself to be used. The following two resolutions and joint agreement were passed:

> "SC4 reconfirms its goal expressed in resolution 2.
>
> To develop as soon as possible a single international standard for exchange of product definition data to be called standard for exchange of product model data (STEP).
>
> SC4 will not concern itself with an interim solution based on any existing national standards although it recognizes that these standards will continue to be in parallel use until STEP becomes fully operational.
>
> All efforts will be concentrated to achieve a first version of STEP so that its effective industrial use can start in 1990."
>
> *[Resolution 9 (March 1985, Paris)]*

> "SC4 agrees that, whenever existing national standards are enhanced, development should be undertaken in parallel with the corresponding STEP effort, using the same reference model to ensure that compatible concepts are used. This will cause standards to converge towards STEP and will simplify the eventual migration from the current standards to STEP."
>
> *[Resolution 10 (March 1985, Paris)]*

> "The US national body is designated to take the leadership role for the development of the ISO STEP standard.

> The US national body will direct its effort towards the development of STEP. The ongoing PDES initiation effort will be completed.
>
> Following completion of STEP development, the national standards and the ISO standard must be identical."
>
> *[Joint agreement]*

As can be seen in Resolution 9, it was anticipated that the development of the standard would take about five years. While the 'effective industrial use' of STEP has not yet come to pass in the way envisaged at Paris, the first set of STEP documents was balloted following the Tokyo meeting in 1989. At that time, the Esprit CAD*I project had demonstrated exchange of physical files containing surface geometry, solid geometry and finite-element information between several CAE systems, using technology similar to STEP. This experience was fed into the development of STEP. All of the major vendors have, for some time, been able to demonstrate a 'STEP capability' based on the Tokyo version of STEP, or (more usually) a later version. In particular, the Esprit CADEX project has demonstrated exchange of files between the major European CAE vendors based on 1992 versions of the documents. The companies in the PDES Inc consortium have also demonstrated STEP processors. Many companies are using the techniques developed in STEP to improve the efficiency of their own enterprises, as described in Chapter 10.

The history of STEP voting is summarized in Appendix C. However, the first ballot—conducted following the Tokyo meeting in 1989 as DP 10303—resulted in a significant reorganization of the structure of STEP, and so is considered here, and also elsewhere [49, 50]. A single document was circulated for comment, comprising several clauses and three normative annexes, totalling over nine hundred pages. It was accompanied by a longer informative annex, giving the requirements which dictated the form and content of the integrated product information models. Clauses and subclauses were at different stages of maturity, so national standards bodies were asked to provide an unofficial vote on each clause and annex, as well as an official vote for the overall document.

It was recognized that the standard was going to be very large, and that implementations were needed of subsets. The concept of application protocols, already being developed in the IGES/PDES Organization, was brought forward to ensure that the earlier practice of vendors choosing, on an *ad hoc* basis, which constructs to implement would not occur for STEP. Coupled with the need for a more explicit framework for the product information models and the need for development of sections of STEP to progress at different rates, STEP was divided into a number of classes of parts, with well-defined relationships between them.

> "SC4 adopts the general strategy for breaking up the DP 10303 into a number of smaller volumes, each to be processed separately through approval of a Committee Draft and then to be balloted upon as a set of qualified and integrated documents representing a Draft International Standard."
>
> *[Resolution 55 (January 1990, Paris)]*

The single working group, in which all technical work had previously been undertaken, was also divided to reflect the new documentary structure; this is described in Appendix B.

The content of the initial release was determined soon afterwards, reflecting the importance assigned by the participating nations to particular domains in product data. The resolution reproduced below captures this, and indicates that technical effort would concentrate on particular areas, rather than on those topics in which the technical volunteers were interested, but which were not needed immediately.

> "SC4 resolves that STEP Version 1.0 comprises the following parts:

| | |
|---|---|
| Overview | Part 1 |
| EXPRESS | Part 11 |
| Physical File | Part 21 |
| Conformance Testing | Part 31 |
| Generic Product Data Model | Part 41 |
| Shape Representation | Part 42 |
| Presentation | Part 46 |
| Drafting | Part 101 |

and one or more Application Protocols as per Resolution #62.

Additional parts may be considered for Version 1.0; however, no additional part will be included if its inclusion will result in a schedule slippage.

Moreover, if there is slippage in the adopted schedule of any part in the above list, then SC4 may vote to defer that part to a future version of STEP.

SC4 recognises the importance of both the documentation architecture and schema framework, and directs the PMAG to ensure their inclusion in STEP Version 1.0."

*[Resolution 68 (June 1990, Göteborg)]*

"SC4 adopts the following recommendation of the ad hoc Application Protocols group:

Following the WG1 decision that STEP Version 1.0 must include at least one draughting related AP, the SC4 ad hoc group on APs recommends that STEP Version 1.0 should include the proposed AP #3,

> "Exchange of 2D geometrically explicit CAD drawings with explicit annotation"

Proposed AP #3 is the top priority for STEP Version 1.0. The proposed APs #4, 5, 6 and 18 should be reviewed by the end of October 1990 to assess their adequacy. These APs could be included in STEP Version 1.0 if they do not require extensive SC4 resources for completion. None of these APs should be allowed to delay the delivery of STEP Version 1.0.

NB: PMAG must assign the responsibility for assessing the adequacy of the proposed APs to the new WG "Qualification and Integration".

| | | |
|---|---|---|
| AP | #4 | Exchange of CAD drawings with reference to 3D geometry model and with explicit annotation |
| AP | #5 | Exchange of configuration controlled 3D product definition data |
| AP | #6 | Exchange of sculptured surface models |
| AP | #18 | Exchange of boundary representation models" |

*[Resolution 62 (June 1990, Göteborg)]*

The application protocols numbered 3, 4, 5, 6 and 18 were chosen from a list of eighteen candidate application protocols voted on by the nations participating in SC4. They now correspond to Parts 201 to 205.

The minimum set required for the initial release of STEP has changed slightly since the resolutions were passed at Gothenburg, as shown in Appendix C, but the structure has remained the same.

# 4

# The structure of STEP

The STEP documentation is partitioned into several classes of parts, which reflect the structure of the standard itself. This, in turn, reflects some of the original requirements: the use of formal methods to describe the information models unambiguously, the separation of application requirements from their fulfilment in a set of integrated models, and the realization of these latter on to a physical medium. The classes are shown below, with the numbers of the parts allocated to each class.



The numerical division within the integrated resources acknowledges that there are certain resources which are generic in nature, and others which apply to an application or range of applications. Figure 4.1 depicts this classification graphically.

It is worthwhile studying Figure 4.1 because of its importance. Firstly, it depicts the structure of STEP, which is one of the main objectives of this book. Secondly, it is the basis of other figures which appear later in the text, which help to focus on particular aspects of STEP.

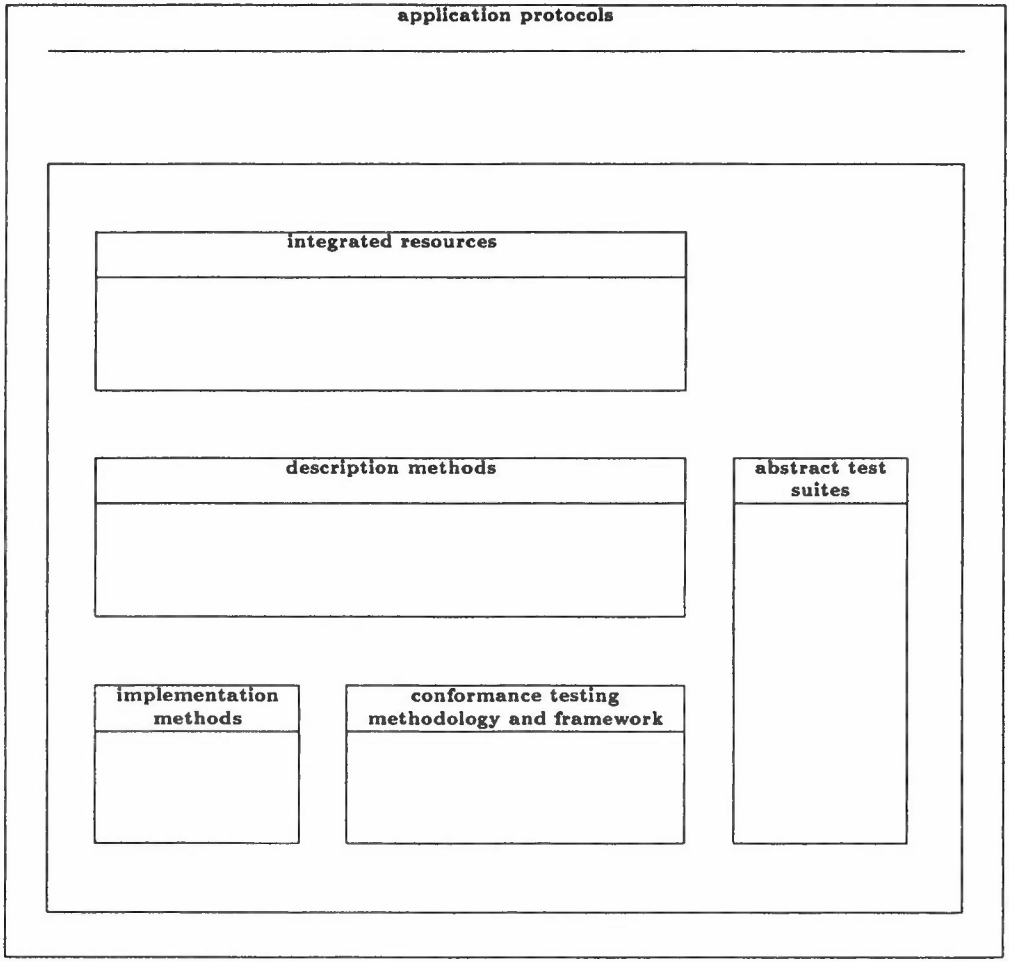



Figure 4.1. The STEP classes.

The six classes are depicted as boxes (the introductory class is omitted). Those which have adjacent edges have some sort of relationship between them. Consequently, the class of description methods is depicted at the heart of the diagram, for this is related to all other classes. The class of application protocols at the top has the strongest relation with the resource models, but the application protocol box is extended to surround all of the other classes. This shows that the application protocols not only interface with all of the other classes too, but that they form the link from STEP to the outside world. A user will read the scope statement of an application protocol to see if it is useful for his own activities, and an implementor will read its self-contained EXPRESS schema with a view to writing software.

The original three-layer architecture of STEP is also preserved in the diagram, with application protocols at the top, the integrated resources in the middle, and implementation methods at the bottom.

Each of these classes is described in the remainder of this text; the four statements which follow demonstrate the role of each class:

1. Formal *description methods* are used in the definition of *integrated resources.*

2. *Application protocols* are developed for a particular application context using the *integrated resources* and *description methods.*

3. An *application protocol* is combined with an *implementation method* to form the basis of a STEP implementation.

4. A STEP implementation is tested for conformance to the standard using the *conformance testing methodology and framework* and the *abstract test suite* associated with the *application protocol.*

At this point, in order to make the classes more tangible, Figure 4.1 showing the structure of STEP is reproduced, populated with parts' titles, as Figure 4.2. All parts with a number allocated at the time of writing have been included; for proposed application protocols please consult Figure 10.2. The SC4 Secretariat or the national standards bodies will be able to provide the latest information.

Note that critical parts (i.e. those intended for the initial release of STEP), as determined (see page 17) at the Paris WG and Gothenburg SC4 meetings in 1990, are marked with a †; those indicated form the original set of nine documents. Those marked with a ‡ are now also included in the minimum set required for the initial release.

Figure 4.2. The STEP classes showing constituent parts.

(See next page.)

**application protocols**

**201** Explicit draughting †
**202** Associative draughting
**203** Configuration controlled design ‡
**204** Mechanical design using boundary representation
**205** Mechanical design using surface representation
**206** Mechanical design using wireframe representation
**207** Sheet metal die planning and design
**208** Life cycle product change process
**209** Design through analysis of composite and metallic structures
**210** Electronic printed circuit assembly, design and manufacture
**211** Electronics test, diagnostics and remanufacture
**212** Electrotechnical plants
**213** NC process plans for machined parts
**214** Core data for automotive design processes
**215** Ship arrangement
**216** Ship moulded forms
**217** Ship piping
**218** Ship structures
**219** Dimensional inspection process planning for coordinate measuring machines using tactile and video sensors

**integrated resources**

generic

**41** Fundamentals of product description and support †
**42** Geometric and topological representation †
**43** Representation structures ‡
**44** Product structure configuration ‡
**45** Materials
**46** Visual presentation †
**47** Shape variation tolerances
**48** Form features
**49** Process structure, property and representation

application

**101** Draughting †
**103** Electrical applications
**104** Finite element analysis
**105** Kinematics

**description methods**

**11** The EXPRESS language reference manual †
**12** The EXPRESS-I language reference manual

**abstract test suites**

**1201** ATS for 201
**1202** ATS for 202
**1203** ATS for 203
**1204** ATS for 204
**1205** ATS for 205
**1206** ATS for 206

**implementation methods**

**21** Clear text encoding of the exchange structure †
**22** Standard data access interface

**conformance testing methodology and framework**

**31** General concepts †
**32** Requirements on testing laboratories and clients
**33** Abstract test suites
**34** Abstract test methods

It is convenient at this point to expand the four statements made above in order to explore the relationships between each of the classes. Until the reader has an appreciation of the parts which constitute each class, which will be gained from the ensuing chapters, he is advised to skim this section, and then return to it later. It is included here because it concerns the structure of STEP. Figure 4.1 is reproduced in Figure 4.3, showing inter-relationships between the classes. This is repeated in a tabular form in Figure 4.4, with the corresponding assertions.

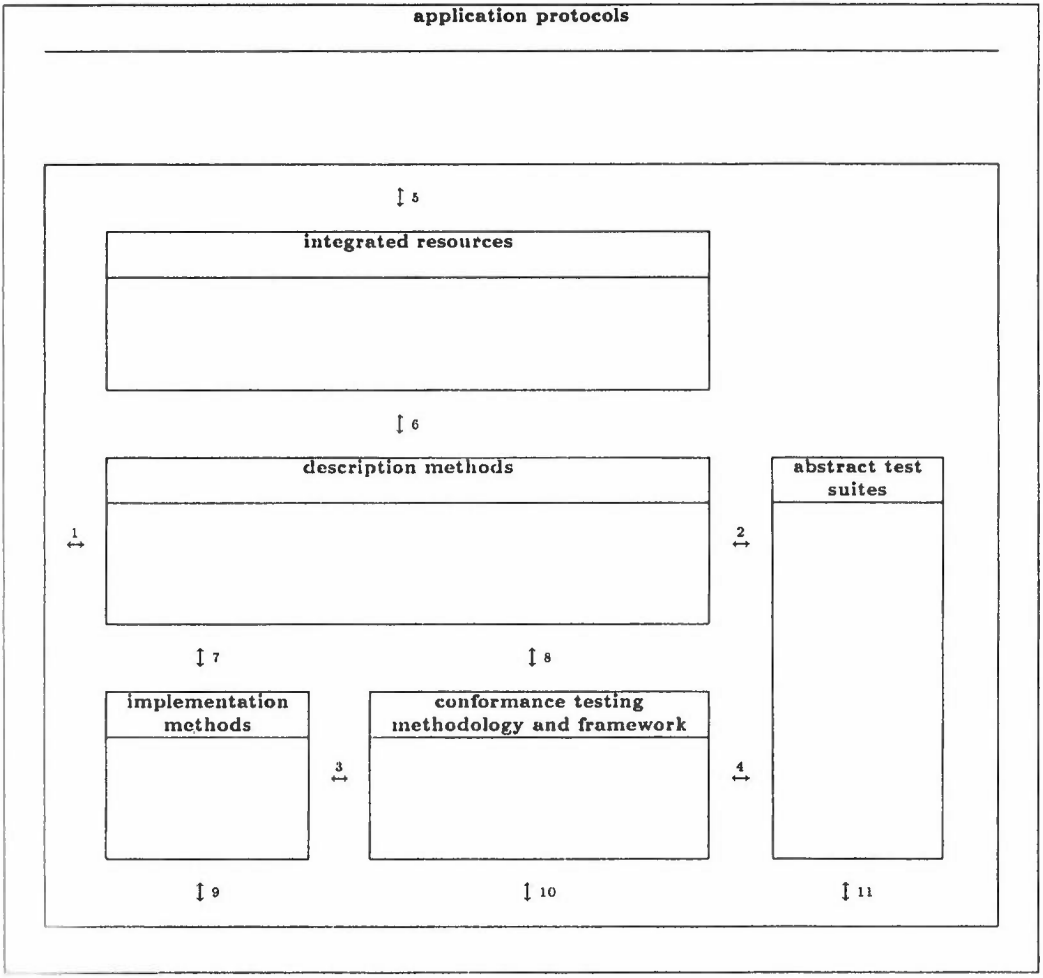



Figure 4.3. The STEP classes showing inter-relationships.

| class title | DM | IM | CT | IR | AP | ATS |
|---|---|---|---|---|---|---|
| description methods | | 7 | 8 | 6 | 1 | 2 |
| implementation methods | 7 | | 3 | none | 9 | none |
| conformance testing | 8 | 3 | | none | 10 | 4 |
| integrated resources | 6 | none | none | | 5 | none |
| application protocols | 1 | 9 | 10 | 5 | | 11 |
| abstract test suites | 2 | none | 4 | none | 11 | |

Figure 4.4. Matrix of inter-relationships between the STEP classes.

1. The normative information model of an application protocol is written in EXPRESS.

2. The abstract test cases within an abstract test suite are written in EXPRESS-I.

   The abstract test cases reflect the structure of and options within the information model written in EXPRESS.

3. For each implementation method, there is a corresponding abstract test method which is used during the conformance testing of a STEP implementation.

   Each implementation method provides requirements (specific to that implementation method) which are used during conformance testing to assess the implementation (e.g. is the syntax of the physical file correct?).

   The different types of conformance requirements used in implementation methods are described in the conformance testing class.

4. The conformance testing class describes the structure of an abstract test suite, and how it is used during conformance testing (and, in particular, its realization into an executable form).

5. The integrated resources are interpreted to produce an application protocol, which provides for a specific application context. This involves the general-purpose constructs in the integrated resources being adapted and refined in the application protocol.

6. The normative information model of an integrated resource is written in EXPRESS.

7. Each implementation method provides a mapping of how each EXPRESS construct is realized in the implementation method. Thus, for any EXPRESS schema, the form of an instance of it in a given implementation method is well defined.

8. The conformance testing class dictates how EXPRESS-I is used in the definition of abstract test cases.

   The conformance testing class describes how an implementation of EXPRESS undergoes conformance testing.

   The different types of conformance requirements used in the description methods are described in the conformance testing class.

9. An application protocol is combined with an implementation method to form the basis of a STEP implementation.

10. The conformance testing class describes how an implementation of a STEP application protocol undergoes conformance testing.

    The different types of conformance requirements used in application protocols are described in the conformance testing class.

11. For each application protocol, there is a corresponding abstract test suite which is used during the conformance testing of a STEP implementation.

The next chapters describe, in turn, the parts which form each class.

# 5

# Description methods

## Overview of Express

EXPRESS is a textual conceptual schema language, as defined in ISO TR9007 [16], based on the entity-attribute-relationship model with generalization and constraint-specification constructs. It is the language used to specify the normative part of all the information models in STEP, both in the integrated resources and the application protocols. It is therefore the source for the definition of multiple implementation methods. Two of its main requirements are that it is both human-readable and computer-processable, and so it conforms to a formal syntax and can be processed by computer software. Although it was originally developed specifically for modelling engineering data, it is gaining wide acceptance in many industrial and academic projects outside STEP activities.

It is *not* a methodology (although one is being developed) and it is *not* a programming language.

EXPRESS has a graphical form, EXPRESS-G, and an instance form, EXPRESS-I, which are described at the end of this chapter.

## Constructs

EXPRESS has seven declarative constructs.

1. *Schema.* A schema is used to define a topic of interest (universe of discourse), such as geometry, and hence to structure and partition the data. It is possible to have inter-schema referencing, which enables a common resource (such as geometry) to be defined independently and then used by several other schemata (such as for draughting and piping).

2. *Type.*

3. *Entity.* Types and entities are used to describe the data, and the relationships between them. They would be used to create the data structures required in a CAE system. It is possible to define generalization-specialization lattices for entities.

4. *Constant.*

5. *Function.*

6. *Procedure.*

7. *Rule.*

   These last three are algorithmic units, for enabling constraints on the data to be specified.

An example EXPRESS schema is shown in Figure 5.3.

### Simple types

There are seven basic (pre-defined) types in EXPRESS: number, integer, real, string, logical, boolean and binary. These may be used as the types of an explicit attribute in an entity declaration:

```
ENTITY point ;
  x : REAL ;
  y : REAL ;
  z : REAL ;
END_ENTITY ;
```

Once an entity has been defined, it may be used as an attribute type in another entity:

```
ENTITY line ;
  p0 : point ;
  p1 : point ;
END_ENTITY ;
```

An instance of this information would appear in a physical file as follows:

```
#21 = POINT (1.0,  3.0, -4.5) ;
#22 = POINT (1.0, -3.0, -4.5) ;
#23 = LINE ( #21, #22 ) ;
```

The third entity in the physical file representation above (the line) contains two references to entities elsewhere in the physical file: they are external to the line definition and are thus available to other entities. If either of the points were deleted by an application receiving this information, the line would become 'invalid'.

## Subtypes and supertypes (generalization)

The subtype and supertype mechanism in EXPRESS enables the inheritance of data and constraints. Any explicit attribute or constraint which applies to a parent supertype also applies to all of its offspring subtypes. If any of these subtypes are themselves supertypes, the data and constraints are passed to their subtypes in turn. This mechanism has analogies in object-oriented programming and database management systems, although *multiple* inheritance is allowed in EXPRESS: an entity may inherit data and constraints from more than one parent, resulting in a network that is a graph rather than a tree.

The next example, in Figure 5.1, introduces the idea of subtypes and supertypes: the `right circular cone` inherits any attributes from its parent entity `primitive with one axis`. This, in turn, would inherit from `csg primitive` and this last from `solid model`. Knowing the alternatives for a `solid model`, it is clear that a `right circular cone` is but one example of a particular subtype of `csg primitive`. Even so, it is possible to review the aspects of a cone in the isolation of its own definition; a review of the alternative representation of a solid model would require much more information than is presented for this example. In this example, all subtype and supertype relationships are assumed to be `oneof`, rather than `and` or `andor`.

Thus, the right circular cone has three attributes which are specific to it, and it inherits its position from its parent. Other examples of a primitive which has one axis of symmetry, such as a right circular cylinder or a torus, would each have their own attributes, but would also inherit their positions from the same parent.

```
ENTITY right_circular_cone
  SUBTYPE OF (primitive_with_one_axis);
  semi_angle : REAL;
  radius     : REAL;
  height     : REAL;
END_ENTITY;

ENTITY axis1_placement;
  location : point;
  axis     : direction;
END_ENTITY;

ENTITY primitive_with_one_axis
  SUBTYPE OF (csg_primitive);
  position   : axis1_placement;
END_ENTITY;

ENTITY csg_primitive
  SUBTYPE OF (solid_model);
END_ENTITY;

ENTITY solid_model
  SUPERTYPE OF (ONEOF (boolean_expression, csg_primitive,
                       csg_solid, facetted_brep,
                       half_space, manifold_solid_brep,
                       solid_instance, swept_area_solid)) ;
  SUBTYPE OF (shape_model);
END_ENTITY;
```

Figure 5.1. Example EXPRESS subschema showing inheritance.

## Aggregate types

Aggregates may also be defined, either of the basic types (as shown below) or of user-defined entities. All are mapped on to the physical file using the same mechanism, but they differ in their behaviour: sets, lists and bags may have an indeterminate upper bound; each member of a set is unique; arrays and lists are ordered whereas sets and bags are unordered.

```
ENTITY aggregate_example ;
  attribute1 : SET [1:3] OF INTEGER ;
  attribute2 : ARRAY [-1:2] OF INTEGER ;
  attribute3 : LIST [0:?] OF LOGICAL ;
  attribute4 : BAG [1:2] OF INTEGER ;
END_ENTITY ;
```

The upper and lower bounds of the declarations for the set, bag and list indicate the maximum and minimum number of elements that the aggregate may have. Despite its similarity in appearance, those for the array define the size and are used for indexing; in the example shown above, the array has exactly four elements.

## Other types

It is also possible to define enumerated types, as can be done in high-level programming languages such as Ada and Pascal. These, in turn, may be used elsewhere in a schema, perhaps as attributes in entities. Names within different enumerations need not be unique.

```
TYPE primary_colour =
  ENUMERATION OF ( red, green, blue ) ;
END_TYPE ;

TYPE traffic_signal =
  ENUMERATION OF ( red, amber, green ) ;
END_TYPE ;
```

Types may also be 'renamed' by using a *defined type*, which provides greater semantic precision when it is used. For example, a new type 'length' could be defined as a real, as could 'area'. Either

of these would give greater meaning to an attribute than defining it directly as a simple real.

The *select type* declares two or more disparate entities to be of the same type. That type can then be used for an attribute or parameter to an algorithmic unit. For example, an assembly comprises both assemblies and components; in turn, these assemblies are also composed of assemblies and components. A select type, in which both are available, could be used to model such a structure.

Local *where* rules, which are described later in this chapter, may also be added to type declarations.

### Derived attributes

Alternative or auxiliary representations of an entity may be provided by using the *derive* construction. The circle below is represented by three points as its explicit attributes. Derivations of its centre and radius may then be provided by means of two functions; this information can be reconstructed from the explicit attributes whenever it is required. (Note that in this example, the names of the explicit attributes do not explain their role: they could represent three points on the circumference, or centre, start and end points.)

```
ENTITY circle ;
  p0 : point ;
  p1 : point ;
  p2 : point ;
DERIVE
  centre : point := f (p0, p1, p2) ;
  radius : real := g (p0, centre) ;
END_ENTITY ;
```

A more realistic example would represent the circle with a centre and a radius as explicit attributes, and provide the algorithm for deriving its area. The function `calculate` would need to be declared in the same scope as this entity, and may be used elsewhere in this scope. Note that any single arithmetic expression may appear on the right-hand side of the assignment symbol.

```
ENTITY circle ;
  centre : point ;
  radius : real ;
DERIVE
  area : real := calculate (radius) ;
END_ENTITY ;
```

### Optional

Some attributes in an entity may be designated as optional, in which case their values need not be provided in an instance. In the example below, the entity captures the nature of a point in either two or three dimensions (although entities which make use of it need to be constructed with care). If the application expects a value and one is not provided (an error), some exceptional action will need to be taken by an implementation.

```
ENTITY point ;
  x : REAL ;
  y : REAL ;
  z : OPTIONAL REAL ;
END_ENTITY ;
```

### Inverse

Inverse relationships may also be captured:

```
ENTITY point ;
  x : REAL ;
  y : REAL ;
  z : REAL ;
INVERSE
  centres : SET [1:?] OF circle FOR centre ;
END_ENTITY ;
```

This states that each point must be used in the role of at least one circle centre. `[0:1]` states "no more than one" whereas `[1:1]` states "exactly one".

### Uniqueness

Attributes in an entity can also be constrained to be unique:

```
ENTITY product_item;
  name        : STRING;
  description : STRING;
  item_id     : STRING;
  versions    : LIST [1 : ?] OF product_item_version;
UNIQUE
  ur1 : item_id;
END_ENTITY;
```

Rather than constraining the value of a single attribute instance to be unique, combinations of such values may be constrained.

### Local rules

Additional constraints may be placed on particular attibutes of an entity by the use of local *where* rules. These are conditions which must be satisfied each and every time the entity is used (instantiated); that is, they are context-independent. They are applicable not only when reading a physical file representation but also in higher-level implementations such as a database.

Some of the local *where* rules in the integrated resources are included but would be very difficult, if not impossible, to code (for example, the extent of an infinite line). However, they do indicate to the reviewer what requirements are being placed on the entity.

```
ENTITY right_circular_cone
  SUBTYPE OF (primitive_with_one_axis);
  semi_angle : REAL;
  radius     : REAL;
  height     : REAL;
WHERE
  semi_angle > 0;
  semi_angle < 90;
  radius >= 0;
  height > 0;
END_ENTITY;
```

## Operators

Seven classes of operator are provided in EXPRESS:

*arithmetic:* `+ - * / ** DIV MOD`

*relational:* `= <> > < >= <= :=: :<>: IN LIKE`

*binary, string:* relational with indexing `[ ]` and concatenation `+`

*logical:* `NOT AND OR XOR`

*aggregate:* `[ ] * + - <= >= QUERY = <> IN`

*component:* `. \ = <> :=: :<>: ||`

## Standard constants, functions and procedures

Seven standard constants are predefined:

- Two mathematical constants: `PI` and e (`CONST_E`).
- The logical constants `FALSE`, `UNKNOWN` and `TRUE`.
- `SELF`.
- *Indeterminate* (`?`), used to specify the upper bound of certain aggregates.

The following standard functions are defined, many of which are familiar from high-level programming languages, and two standard procedures are defined, for use with aggregates:

| | | | | | | |
|---|---|---|---|---|---|---|
| abs | acos | asin | atan | blength | cos | exists |
| exp | format | hibound | hiindex | length | lobound | log |
| log2 | log10 | loindex | nvl | odd | rolesof | sin |
| sizeof | sqrt | tan | typeof | usedin | value | |
| remove | insert | | | | | |

### Statements

The following 'executable' statements are provided in EXPRESS, for the specification of constraints:

- alias
- assignment
- case
- compound (begin/end)
- escape (transfer out of repeat block)
- if then else
- null
- procedure call
- repeat (increment, until, while)
- return
- skip (to end of repeat block)

### Overview of Express–G

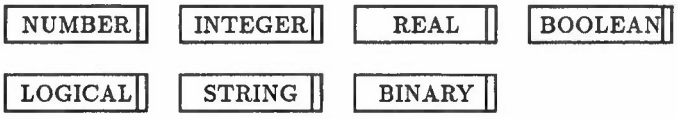
EXPRESS–G is a formal graphical notation of a subset of EXPRESS. It is defined in a normative annex to Part 11 and is used for human communication. It provides constructs for the following:

- Schema and inter-schema links.
- Entity and entity generalization.
- Attribute.
- Relationship and cardinality.
- Type.
- Multi-page referencing.

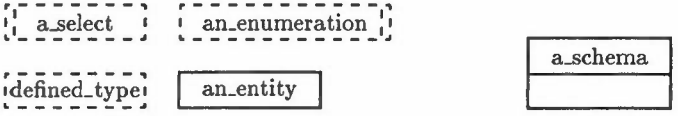
It does not provide support for the specification of constraints.

Most of its symbols are provided in a key in Figure 5.1. An example EXPRESS schema is provided in Figure 5.3 and its EXPRESS–G rendition in Figures 5.4 and 5.5.

**Simple data types:**




**Other definition symbols:**




**Relationship line styles:**

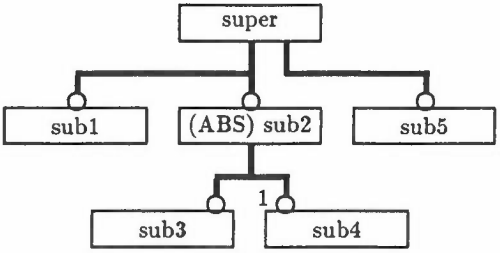
Normal line

Dashed line (optional, reference)

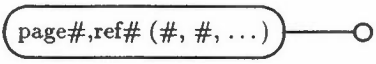
Thick line (sub/supertype)

Figure 5.2. EXPRESS–G: key to symbols.

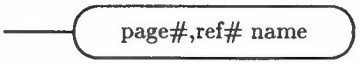
**Inheritance graph example:**




**Page references:**

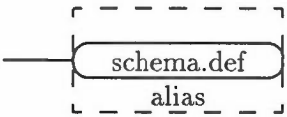



reference from another page on to this page

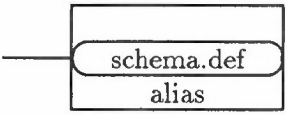



reference from this page on to another page

**Inter-schema references:**




definition REFERENCEd from another schema




definition USEd from another schema

```
SCHEMA royal_example;

TYPE date = ARRAY [1:3] OF INTEGER;
END_TYPE;

TYPE hair_type = ENUMERATION OF
     (fair, brown, black,
      red, grey, bald);
END_TYPE;

ENTITY person SUPERTYPE OF (ONEOF(female, male));
  first_name : SET [1:?] OF STRING;
  last_name  : STRING;
  title      : OPTIONAL STRING;
  birth_date : date;
  death_date : OPTIONAL date;
  children   : SET [0:?] OF person;
  hair       : hair_type;
DERIVE
  age : INTEGER := years(birth_date);
  alive : BOOLEAN := NOT EXISTS(death_date);
INVERSE
  parents : SET [0:2] OF person FOR children;
END_ENTITY;

ENTITY female SUBTYPE OF (person);
  husband     : OPTIONAL male;
  maiden_name : OPTIONAL STRING;
WHERE
  w1: (EXISTS(maiden_name) AND EXISTS(husband)) XOR
      NOT EXISTS(maiden_name);
END_ENTITY;

ENTITY male SUBTYPE OF (person);
  wife : OPTIONAL female;
END_ENTITY;

RULE married FOR (female, male);  (* to be written *)
(* checks pairwise relationship between spouses *)


FUNCTION years(past : date): INTEGER; (* to be written *)
```

Figure 5.3. Example EXPRESS schema.

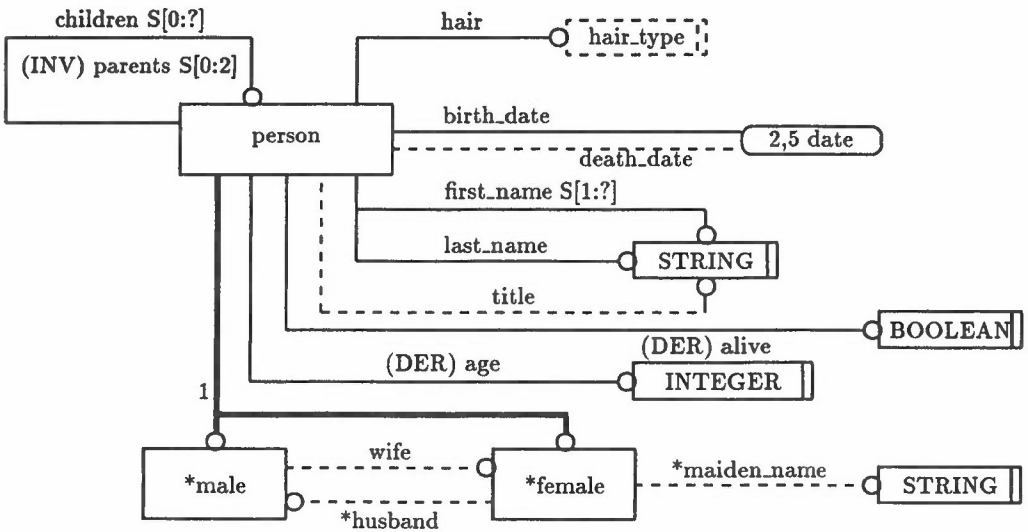



Figure 5.4. Complete entity-level diagram of example schema shown in Figure 5.3 (page 1 of 2).

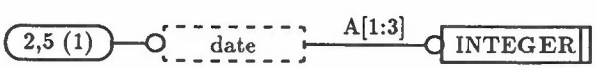



Figure 5.5. Complete entity-level diagram of example schema shown in Figure 5.3 (page 2 of 2).

### Overview of Express–I

Whereas EXPRESS and EXPRESS–G are languages which enable the form and structure of information to be defined, EXPRESS–I is an instantiation language. This enables instances of a schema, or parts of that schema, defined in EXPRESS, to be displayed.

There are two major parts of the language.

1. Display of data instances, at any of the following levels:
    - o entity (object instance);
    - o schema (schema instance);
    - o collection of schema instances (model).
2. Specification of abstract test cases used in conformance testing (see Chapter 9).
    - o context, which may have formal parameters and default values;
    - o test case, which is built from one or more contexts.

Mappings from EXPRESS schemata and data types to EXPRESS–I instances are also defined. However, EXPRESS–I is not intended to be an implementation method—hence its inclusion in the class of description methods as Part 12.

### Software tools

Several software tools are already available, both in the public domain and commercially; many are being developed. They fall into one of the following classes:

- o Editors, including both language-sensitive editors for EXPRESS and graphical editors for EXPRESS–G.
- o Parsers.
- o Syntax checkers.
- o Semantic checkers.

- Compilers, which either produce object code directly or a high-level language which itself can be compiled into object code.
- System-building tools (for example: a schema manager).

# 6

# Integrated resources

The class of integrated resources, as its name suggests, provides an assimilated set of information models, which are the resources from which application protocols are built. The class contains two types of integrated resources: generic (which have a general applicability) and application (which support a single application or range of similar applications). This context-independent product data is encapsulated in an implementation-independent form in EXPRESS and is implemented indirectly using an application protocol.

## Contents

Each information model is developed from requirements, specified by a resource reference model, from a different area or aspect of product data; the reference model, written in EXPRESS–G, may be given as an informative annex. In practice, the application protocols, which serve an industrial need, provide many of the specific requirements, although the original design goals of STEP and the structure of the standard itself supply others.

The normative data content of the integrated resources is formulated in EXPRESS. Such definitions are independent of the many possible ways in which this data might be implemented. All of the parts in the class of integrated resources have the same structure. Some aspects of this are dictated by the form of an ISO standard; some are self-imposed by the STEP community to provide a uniformity which enables a reader to assimilate the information more easily. The structure is shown in Figure 6.1.

In order to illustrate these principles, consider a sample resource information model—Part 101. This is a general draughting resource which provides general-purpose entities which are common to all ap-

| | Clauses |
|---|---|
| | Foreword |
| | Introduction |
| 1 | Scope |
| 2 | Normative references |
| 3 | Definitions, symbols and abbreviations |
| 4 | Requirements (EXPRESS schema) [may be repeated] |

| | Annexes |
|---|---|
| A | Short names of entities (electronic form also available) |
| B | EXPRESS listing (electronic form only) |
| C | EXPRESS-G, NIAM or IDEF1X model |
| D | Bibliography |
| E | Model scope (data planning and activity models) |
| F | Examples |

Appendix A is normative and required; B is informative and required; the remainder are informative and optional.

Figure 6.1. Table of contents for an integrated resources part.

plications of draughting. It is implemented using application protocols which give context and constraints:

- **Part 201:** Explicit draughting.
- **Part 202:** Associative draughting.

## Overview

Figures 6.2 and 6.3 show the part numbers and titles of the generic integrated resources and application integrated resources respectively. An indication of the content of each part is also provided.

It would be inappropriate to describe each of the integrated resource information models in an overview text such as this. However, in order to demonstrate their role in more detail, and hence to show the structure of STEP, the first three of the generic resources are discussed. These have been chosen because their contents will necessarily appear in every application protocol in some

form, whereas other resources—such as draughting—will appear only in the related application protocols. These first three integrated resource parts provide material common to all aspects of product data.

| Part | Title | Contents |
|---|---|---|
| 41 | Fundamentals of product description and support | framework for integrated resources, product definition, context, properties, physical quantities and their units, management resources |
| 42 | Geometric and topological representation | geometry, topology, shape types |
| 43 | Representation structures | representation interface |
| 44 | Product structure configuration | parts, versions, assemblies, components |
| 45 | Materials | material properties |
| 46 | Visual presentation | colours, symbols, libraries, line styles, patterns, text, views |
| 47 | Shape variation tolerances | three dimensional shape variation tolerances |
| 48 | Form features | classification and representation of areas of shape regions |
| 49 | Process structure, property and representation | elements of a process plan and relationships between them |

Figure 6.2. The generic integrated resources.

| Part | Title | Contents |
|---|---|---|
| 101 | Draughting | annotation, dimension representation, sections, notes, drawing, sheet, views |
| 102 | (Ship structures) | (now deleted) |
| 103 | Electrical applications | schematics; resources to support electrical and electronic connectivity |
| 104 | Finite element analysis | FEA model, control, result |
| 105 | Kinematics | kinematics model, analysis, control, result |

Figure 6.3. The application integrated resources.

### Part 41: Fundamentals of product description and support

Part 41 provides the fundamentals of product description and support in several schemata, which are organized into three major subdivisions in the document:

1. Generic product description resources.
2. Generic management resources.
3. Support resources.

The generic product description resources provide a framework for all of the integrated resources. When this framework is combined with the other integrated resource parts, an integrated set of product description resources is the result. This is the foundation upon which all application protocols are built, and is shown in Figure 6.4.

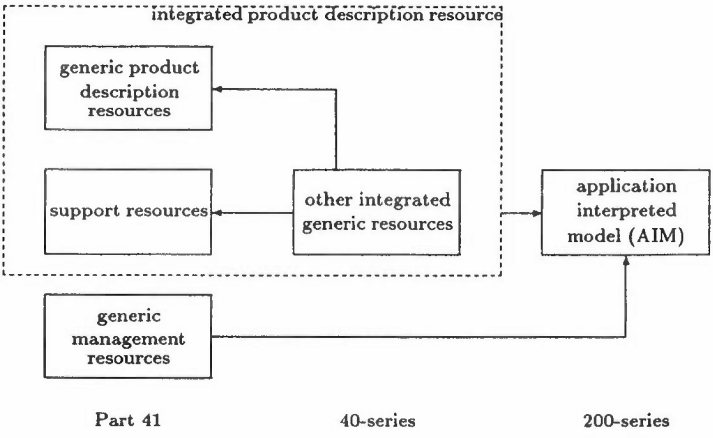



Figure 6.4. STEP architecture for product description and support.

The framework provides a structure so that products and their properties, which includes their shape, can be defined. It also enables products to be identified, categorized and associated with one

another; and versions of products and the relationships between them to be specified. How the product data are to be used can also be described. There are four schemata:

- application context;
- product definition;
- product property definition;
- product property representation.

The generic management resources comprise a single schema which enables administrative data to be associated with the product data. This is required by an enterprise in order to be able to manage the product data.

The support resources specify those constructs which are shared by other resource schemata, including those in other integrated resource parts. They comprise the following schemata:

*document:* allows references to specifications, including standards other than STEP;

*action:* enables the specification of work, requests for work, status of work items;

*certification:* allows certification information to be referenced;

*approval:* enables authorization data to be specified;

*contract:* allows identification of contracts;

*security classification:* provides means to specify levels of confidentiality;

*person organization:* provides mechanisms for the identification of people and organizations, and the roles they play;

*date time:* dates and times;

*group:* provides a mechanism for grouping items;

*external reference:* enables the identification of information not represented explicitly in the domain of an application protocol;

*support resource:* basic data types (identifier, label, text);

*measure:* physical quantities (e.g. length, mass, time, density).

### Part 42: Geometric and topological representation

Part 42 provides geometric and topological representations in three schemata. The subtype and supertype structure of each schema is shown in Figures 6.5–6.9. Although these do not show how each of the entities is used as attibutes in other entities, they do provide the classification of the geometry, topology and geometric models.

Figure 6.5 shows that geometric representation item (defined in Part 43) has seven subtypes within the geometry schema:

1. placement;
2. cartesian transformation operator;
3. point;
4. vector;
5. direction;
6. curve (see Figure 6.6);
7. surface (see Figure 6.7).

The basic building blocks for the geometry schema itself are the `cartesian point` and the `direction`. These are used to define `placements`, which give a position and an orientation in two- or three-dimensional space. In turn, a `curve` or `surface` is positioned using an appropriate `placement`. Entities are used in the definition of others, including those within a common parent supertype: a `composite curve` necessarily uses other `curves` and a `surface replica` copies a `surface` at another `placement`.

It should be noted that most geometry entities have an associated parameterization, and that the representation for each entity has been chosen to minimize ill-conditioning.

Figure 6.8 shows that topological representation item (from Part 43) has ten subtypes:

1. vertex;
2. edge;
3. path;
4. loop;
5. face;
6. face bound;
7. vertex shell;
8. wire shell;
9. connected face set;
10. connected edge set.

Vertices are built into edges which, in turn, are used to define paths, loops, faces and, finally, shells. These topology entities can be used for defining *any* information where connectivity is fundamental. A geometry entity may be associated with the topology (e.g. a point with a vertex, or a surface with a face) so that the shape of objects can be defined as boundary representations, but the association is optional. Thus, products can be defined where the connectivity is important but the position or size is not (e.g. electrical wiring).

The geometric model schema, shown in Figure 6.9, provides further subtypes of the geometric representation item (from Part 43); a box domain is isolated in terms of the subtype and supertype tree but it is used in the definition of the other entities and so is shown for completeness. The geometric models can be constructive solid geometry primitives, solid models, or one of a set of incompletely defined models:

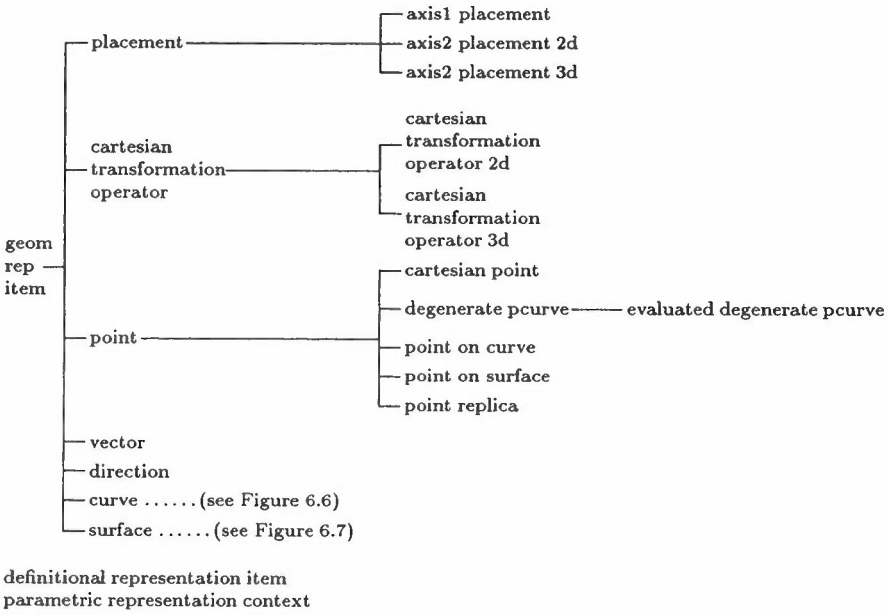



Figure 6.5. Geometry schema.

1. shell based surface model;
2. face based surface model;
3. shell based wireframe model;
4. edge based wireframe model;
5. geometric set.

These three schemata can be used to define the shape of an object in a particular context defined by an application protocol.

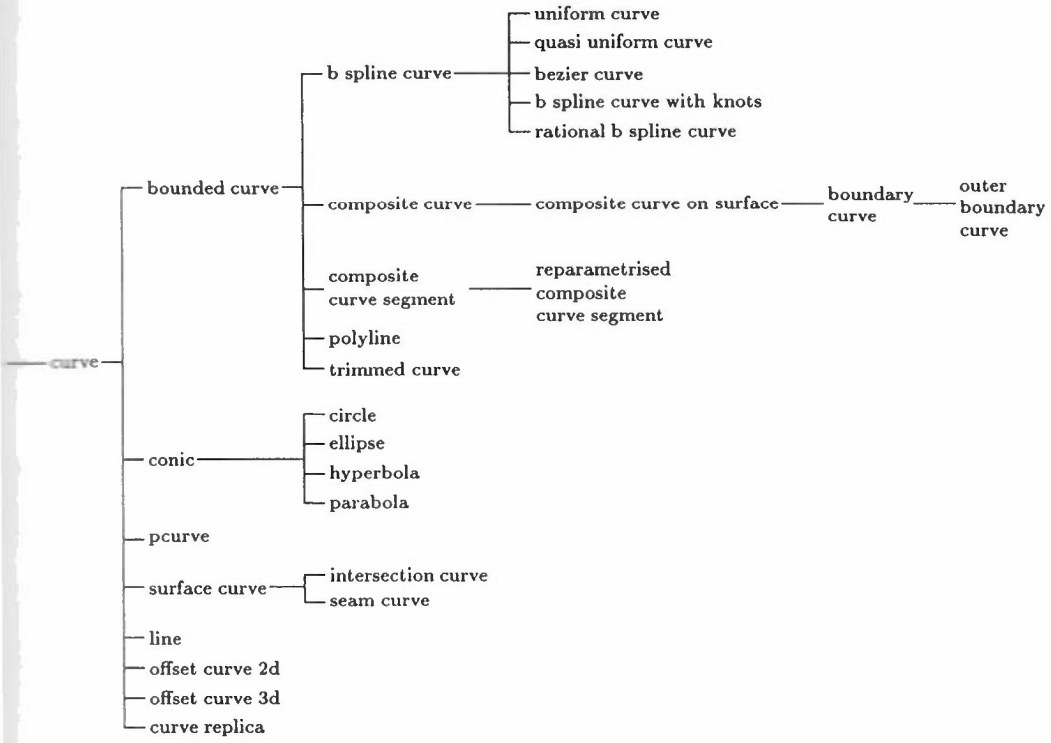



Figure 6.6. Geometry schema (curve structure).

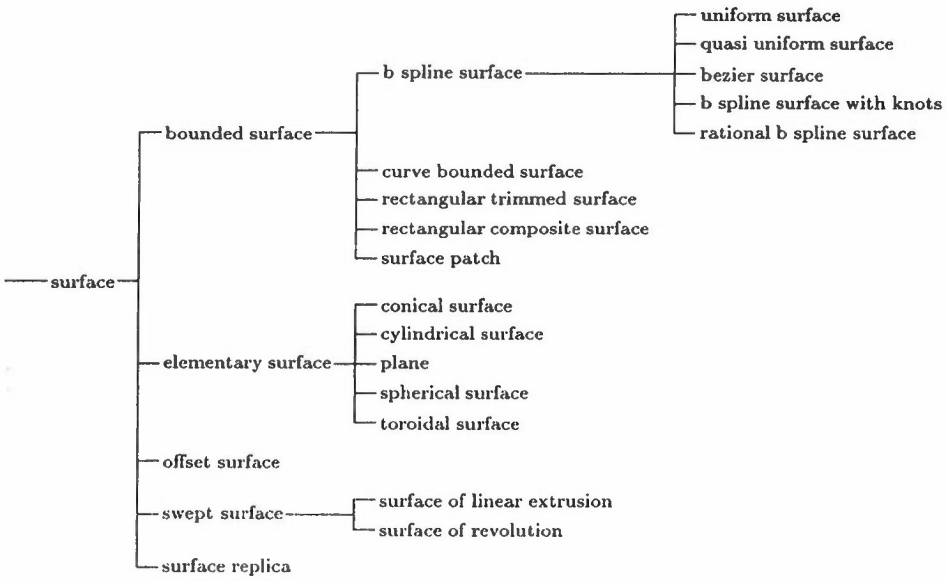
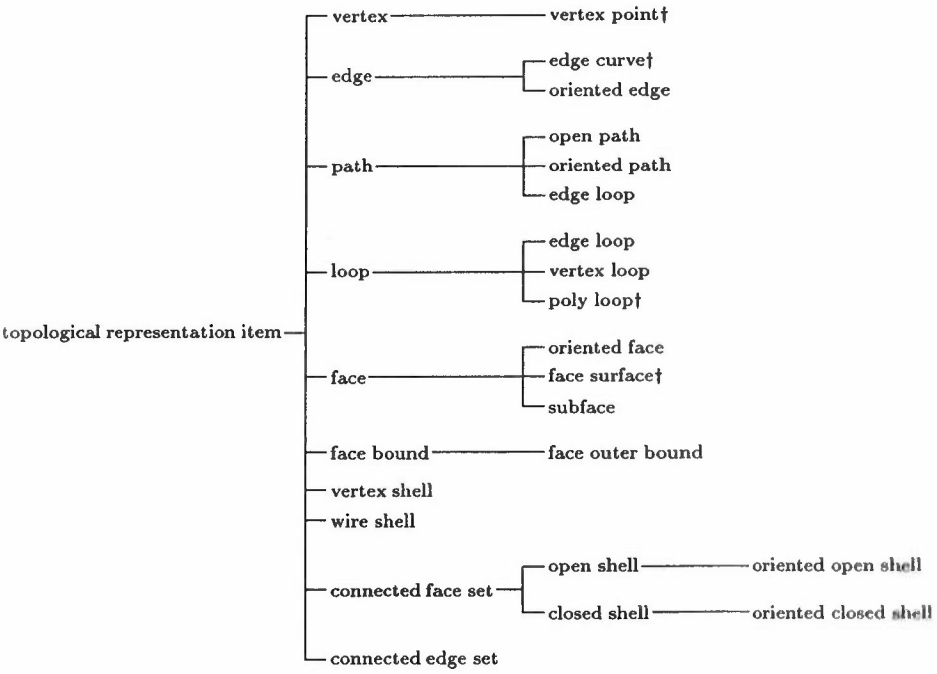
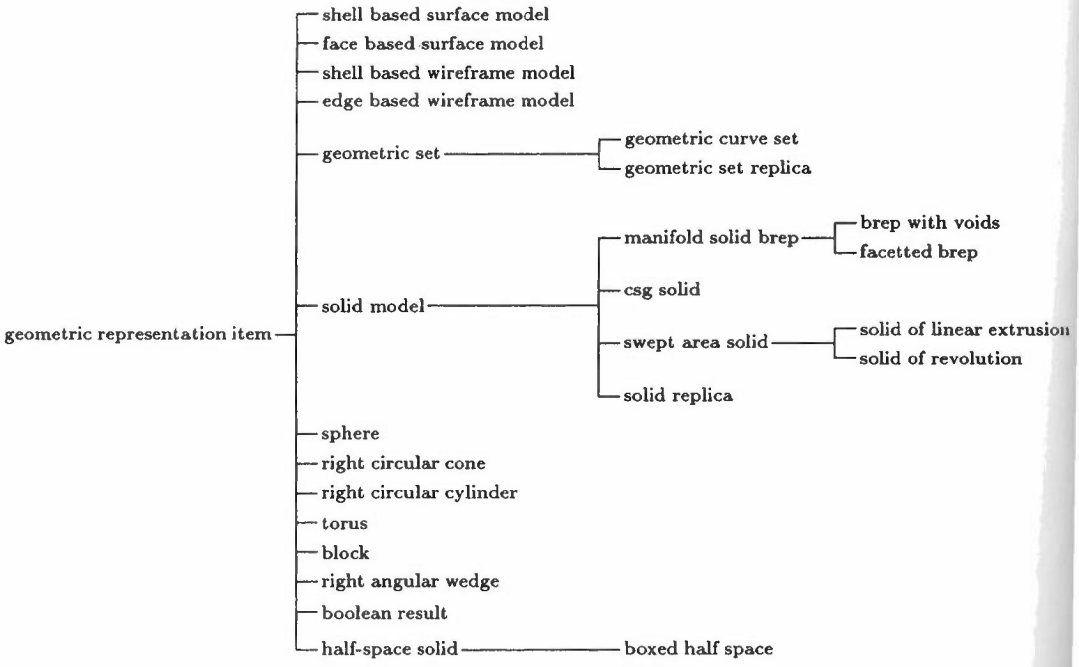



Figure 6.7. Geometry schema (surface structure).




*Note:* those entities marked with a † are also subtypes of geometric representation item.

Figure 6.8. Topology schema.




*Note:* box domain has no supertype.

Figure 6.9. Geometric model schema.

## Part 43: Representation structures

Part 43 specifies the overall structure for representation, of which geometric representation is a special (and obvious) case. It enables entities which are used to describe representation to be associated into collections, so that such items can be identified as being related or not. Relationships may also be specified between the collections.

For example, a geometric element defined using the constructs provided in Part 42 does not have any meaning until it is placed in a particular context: a coordinate system. Several components of a product may be defined in a series of local coordinate systems, which are defined relative to one another using a transformation. Each local coordinate system will, ultimately, be placed in the world, or master, coordinate system.

As well as providing the structure and support for representations in general, Part 43 defines the constructs which are required particularly for geometric representations. Consequently, it provides the articulation between the framework for products in Part 41 and the geometric and topological constructs in Part 42. These three parts—as stated earlier—provide material common to all aspects of product data.

More details of the integrated resource information models are provided elsewhere in this series of books.

# 7

# Application protocols and abstract test suites

## Introduction

Without application protocols, system vendors would be free to implement non-standardized subsets of the integrated resources, repeating the problems of existing standards (such as IGES). Consequently, an application protocol provides the comprehensive requirements for implementations, by defining the application domain (or context). This is achieved by constructing the application protocol from an application-specific interpretation of the context-independent entities present in the integrated resources. In addition, an application protocol defines the conformance requirements that provide the basis for conformance testing; specific implementation-method characteristics may also be included. A STEP implementation is produced from the combination of an application protocol and a particular implementation method.

In order to reflect these basic principles, an application protocol comprises four main sections:

1. Scope.
2. Requirements (application reference model).
3. Application interpreted model.
4. Conformance requirements.

The full structure is shown in Figure 7.1; as with the integrated resources, the uniformity across parts in the same class is intended to help the reader.

### Scope

The scope enables implementors and users to make an initial evaluation of the standard against system capabilities or user requirements. The application protocol project uses the scope to capture and document the industrial requirements that the application protocol addresses: the resulting application protocol defines the data necessary to support the 'in scope' information flows. It is documented in one or more of the following:

- IDEF0 (or SADT): application activity model.
- IDEF1X, NIAM, EXPRESS–G: planning model.
- English.

### Requirements (application reference model)

The application reference model defines a model giving complete details of the information units required to support the activities identified as lying within the application protocol. It is defined and documented using one of:

- NIAM [44].
- IDEF1X [14].
- EXPRESS–G.

### Application interpreted model

The application interpreted model (AIM) comprises a schema which is the result of the interpretation of the integrated resources consistent with the application reference model constructs. In other words, requirements in the application reference model are satisfied by taking the general-purpose integrated resources and interpreting

them for use within the application context provided by the application protocol. Thus, AIM development is a joint effort between an application protocol development project and integrated resource experts. The AIM includes a short form of an EXPRESS schema; it specifies the names of constructs used and referenced from the integrated resources, the schema in which each construct is defined, and the additional constraints applied to them in this context. This is expanded into a long form, which is an equivalent, self-contained EXPRESS schema. Although this was provided in paper form in early versions of the application protocols, it is now included electronically. The reason for its inclusion is to save a reader of the document (whether a potential user or implementor) having to cross-reference many parts which form the integrated resources. (As an example, Part 201 requires information from Parts 41, 42, 43, 46 and 101.) The third section of the AIM is the mapping table, which shows how each requirement in the application reference model is satisfied by one or more EXPRESS constructs, and the role that each EXPRESS construct plays in the application protocol. There are potentially many of the latter.

## Conformance requirements

The application reference model is used in conjunction with the long form of the application interpreted model to develop the detailed test purposes used in conformance testing. Each implementation of an application protocol shall satisfy the conformance criteria in that individual part and any normative references made from it. Part 31 requires that an application protocol states which conformance requirements are mandatory, which are conditional and which (if any) are optional. Test purposes identify all options that are to be exercised in conformance testing of implementations of the application protocol. In practice, they highlight all forms which the information model defined by the application protocol may take, and thus help the application protocol project to determine the accuracy and relevance of some of the forms. The inclusion of a particular test purpose may result in the scope of the application protocol being reduced, with consequent changes being made in the application reference model, the application interpreted model and the conformance requirements. The conformance requirements and test pur-

poses are used in creating an abstract test suite for the application protocol. From this abstract test suite, testing laboratories create the executable test cases to be applied during conformance testing. The abstract test suite is itself standardized in the 1200-series class of parts. Although the test purposes were documented originally in the application protocol, the class of abstract test suites was added to the structure of STEP, and so they are now included here.

In practice, an application protocol is often divided into levels, and requires that a vendor implements all of one (or more) particular level; an incomplete implementation of a particular level would be non-conforming. Each of these levels can itself be regarded as a miniature application protocol.

The options or levels implemented in the software are elicited during conformance testing by using the PICS proforma, which is provided as a normative annex of each application protocol.

| | Clauses |
|---|---|
| | Foreword |
| | Introduction |
| 1 | Scope |
| 2 | Normative references |
| 3 | Definitions, symbols and abbreviations |
| 4 | Information requirements |
| 5 | Application interpreted model (AIM) |
| 6 | Conformance requirements |

| | Annexes |
|---|---|
| A | AIM EXPRESS annotated listing |
| B | AIM short names (electronic form also available) |
| C | Implementation method specific requirements |
| D | PICS proforma |
| E | Application activity model (usually IDEF0) |
| F | Application reference model (NIAM, IDEF1X, EXPRESS or EXPRESS-G) |
| G | Application interpreted model EXPRESS–G diagrams |
| H | Application interpreted model EXPRESS listing (electronic form only) |
| J | Bibliography |
| K | Application protocol usage guide |
| L | Technical discussions |

Annexes A to D are normative; E to J informative and required; K and L informative and optional.

Clause 4 comprises the units of functionality, the application objects and the application assertions (which document the relationships between the application objects). Clause 5 comprises the mapping table between the application objects and the application interpreted model, organized by the units of functionality, and the short form of the application interpreted model EXPRESS schema.

Figure 7.1. Table of contents for an application protocol part.

### Overview

In the previous chapter, the integrated resources were not described in detail. Similarly, it would be inappropriate to describe each of the application protocols. This is the role of other texts in this series, which will be able to report on work now in progress (see Chapter 10). However, an overview of the contents of Parts 201 to 206 is given, in order to provide a flavour of the coverage of the first STEP application protocols.

### Part 201: Explicit draughting

Part 201 is used for the exchange of individual (not multiple) technical CAD drawings. It enables the presentation of product shape in terms of explicit two-dimensional geometry and of product properties conveyed by explicit two-dimensional annotation. There is no association between geometry and annotation. There is, however, provision for administrative data and drawing layout.

Drawings are organized into drawing sheets, each with its own administrative data if required. Product shape is represented as two-dimensional geometry and presented in different drawing views. The presentation of annotation is placed in a drawing view or on a drawing sheet, and includes sheet layout. Both annotation and geometry may be grouped, using layers, groups or subfigures. Annotation elements may be aggregated into units which represent dimensions (angular, radius, ordinate, diameter, point and both linear and non-linear curves) and tolerances.

### Part 202: Associative draughting

Part 202 has a similar provision to Part 201, with the important difference that the drawing views may be associated with the geometric representation of the shape, and the dimensions with the dimensioned geometry. Consequently, shape may be represented in either two or three dimensions.

The geometric representation of shape may be one of:

1. Advanced Brep.

2. Elementary Brep.

3. Facetted Brep.
4. Surface without topology.
5. Non-manifold surfaces with topology.
6. Manifold surfaces with topology.
7. Wireframe geometry without topology.
8. Edge-based wireframe geometry with topology.
9. Shell-based wireframe geometry with topology.

Parts 201 and 202 have a common application reference model.

### Part 203: Configuration controlled design

Part 203 enables the exchange of configuration-controlled design information, with or without the shape of the object. It has six levels, the first of which must be supported in an implementation. Five further levels provide configuration control including the shape using the representation shown:

1. Constructs representing configuration-controlled design information without shape.
2. Wireframe models which have no surface topology.
3. Wireframe models with topology.
4. Manifold surface models with topology.
5. Facetted Brep.
6. Advanced Brep.

### Part 204: Mechanical design using boundary representation

Part 204 has three functional levels, distinguished by the complexity of the shape being represented:

1. Facetted Brep (planar surfaces).

2. Elementary Brep (analytic surfaces: planes, cones, cylinders, spheres, tori, and also general swept surfaces based on lines and conics).

3. Advanced Brep (elementary surfaces, sculptured surfaces using B-splines, and swept surfaces—linear or rotational extrusions based on analytic or free-form curves).

Whereas the first level has implicit topology, levels two and three have explicit topology, which provides the connectivity and trimming information for the unbounded geometry.

Presentation (predefined colour, line styles and line widths) may be attached to any geometry or topology. Annotation text is provided in three dimensions, and can be combined with leaders (arrows). A layer mechanism is also available.

### Part 205: Mechanical design using surface representation

Part 205 defines three functional levels, and has the same presentation attributes as Part 204.

1. Geometrically bounded surface models.

2. Non-manifold surface models.

3. Manifold surface models.

### Part 206: Mechanical design using wireframe representation

Part 206 has four functional levels:

1. Geometry bound wireframe model (trimmed curves used to bound lines and open conics; geometry may be grouped into geometric sets).

2. Topology bound wireframe model (edges bound all curve geometry; topology may be grouped into connected edge sets).

3. Shell based wireframe model (edges bound all curve geometry; loop information is maintained; topology may be grouped into vertex and wire shells).

4. Mixed based wireframe model (both shell-based and edge-based).

It has been designed to be compatible with existing standards (it is a superset of VDA-FS and VDA-IS) and with Parts 204 and 205 (of which it is a subset).

## Application interpreted constructs

During the simultaneous interpretation of several application protocols, it became evident that certain groups of constructs were going to be used in each application protocol. Consequently, the idea of the application interpreted construct was formulated. Rather than build application interpreted models from the atomic EXPRESS constructs—entities, types, rules, functions and so on—such constructs are grouped into larger building blocks. For example, the application interpreted construct for facetted boundary representations (that is, with only planar surfaces) has been defined and documented. This is then used in the application protocols for both Parts 203 and 204, and as one of the shapes for which associative draughting is provided in Part 202. It is included in a different context in each case. This not only enables application interpreted models to be built more quickly, but opens up the possibility of sharing information between implementations of different application protocols which use the same application interpreted constructs: the concept of interoperability. Whilst this will not allow *all* of the information in a physical file reflecting one application protocol to be read by a processor of another, *some* of the information could be processed. However, this area does require further work, even though the possibility is now present.

The original Figure 4.1 showing the STEP classes is reproduced in Figure 7.2 showing the role of application interpreted constructs and their relationships with the original classes.

1-11. For an explanation of relationships numbered from 1 to 11, the text associated with Figures 4.3 and 4.4 should be consulted.

12. An application interpreted construct is built from the integrated resources, and comprises a logically-contained group of constructs which provide a particular function.

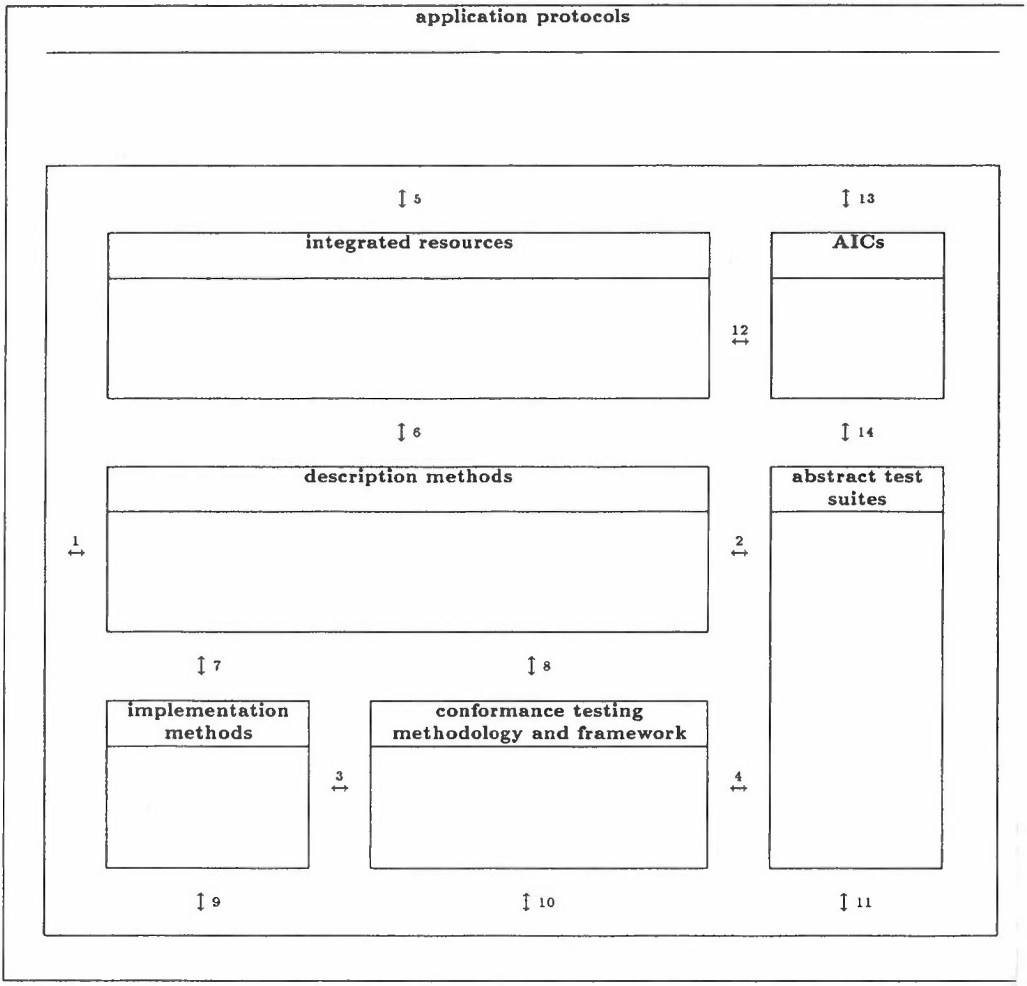



Figure 7.2. The STEP classes showing the role of AICs.

13. An application protocol is built using application interpreted constructs, which provide the basis for the same information structures to be shared by different application protocols.

14. An abstract test suite is built using contexts in EXPRESS–I, which are instances with default values of the information structures defined by the application interpreted constructs and documented with them.

The following application interpreted constructs are available and documented, and will become ISO Technical Reports:

1. mechanical design context;
2. name assignment;
3. basic shape presentation;
4. topologically bounded elementary surface;
5. topologically bounded surface;
6. geometrically bounded surface;
7. manifold surface;
8. non-manifold surface.

In addition, application interpreted constructs are available which correspond to each of the functional levels in Parts 204, 205 and 206. Several others are being developed.

Parts 204, 205 and 206 received considerable development effort from the CADEX project and, as a result, share information structures by using the same application interpreted constructs, as shown in Figure 7.3. In turn, these AICs are used by Parts 202 and 203.

## Non-standard application protocols

Although such application protocols may become international standards, and conforming implementations of them (using whatever implementation method) also may become available, the method of building an application protocol is of fundamental importance

| AIC name | 204 | | | 205 | | | 206 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F | E | A | 1 | 2 | 3 | 1 | 2 | 3 | 4 |
| geometric measures | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| mechanical design context | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| name assignment | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| basic shape presentation | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| topologically bounded elementary surface | | √ | √ | | √ | √ | | | | |
| topologically bounded surface | | | √ | | √ | √ | | | | |
| facetted Brep | √ | | | | | | | | | |
| elementary Brep | | √ | | | | | | | | |
| advanced Brep | | | √ | | | | | | | |
| geo bound surface | | | | √ | | | | | | |
| non manifold surface | | | | | √ | | | | | |
| manifold surface | | | | | | √ | | | | |
| geometry bound wireframe | | | | | | | √ | | | |
| topology bound wireframe | | | | | | | | √ | | √ |
| shell based wireframe | | | | | | | | | √ | √ |

Figure 7.3. AICs used by Parts 204, 205 and 206.

[23]. The construction uses structured methods to define scope, requirements and reference model. Conformance requirements and test purposes are included as part of the standard. The structure enables information requirements to be separated from implementation methods. The construction method can therefore be used for application protocols which will either become national standards, or industrial sector based 'standards', or specifications used only within a single engineering enterprise. This is discussed more fully in Chapter 10.

## Abstract test suites

For each application protocol, there is a corresponding standard abstract test suite, documented as a part with a number which is one thousand higher than that of the application protocol. The abstract

test suite provides the set of abstract test cases to be used during conformance testing of any implementation of the application protocol. These abstract test cases are derived from the conformance requirements (documented in the application protocol) and the test purposes (documented in the abstract test suite). Abstract test cases are written in a formally-defined language, EXPRESS–I. The abstract test cases are both human readable and computer processable, and provide the basis for generating the executable test suite used during the conformance assessment process, as described in Chapter 9.

Each abstract test case contains:

- A unique test case identifier.
- The test purpose.
- One or more references to specific clauses of the relevant standards.
- Verdict criteria.
- A test model.

This last is constructed from a series of EXPRESS–I context blocks, which correspond to the application interpreted constructs as the building blocks for the application protocols, and are documented with them. The abstract test cases are written using parameters so that different values can easily be used.

Each abstract test suite contains an administrative header. There are also verdict criteria which are applied to groups of abstract test cases. Finally, there is a mapping from the PICS proforma to the abstract test cases so that the appropriate abstract test cases are selected depending upon which options within the application protocol have been implemented in the software.

# 8

# Implementation methods

One of the first problems identified with the existing generation of standards for product data exchange was the blurring of an application requirement with its fulfilment and then its realization on a physical file (at that time, the only implementation method available). An application expert would start reading a passage of text expressed in terms specific to that application, only to find references concerning where in a file a particular item of information should appear, or a series of scoping rules.

Consequently, for STEP, the class of *implementation methods* was created, enabling the structure of the physical file to be defined without requiring *any* knowledge about the application. This satisfied the original requirement for this separation, but also left the way clear for implementation methods other than the physical file to be defined. Four were suggested originally, but only the first was well-defined.

1. Physical file—text file.
2. Active file exchange—software assisted.
3. Shared databases (DBMS)—relational and network database technologies.
4. Intelligent knowledge-based systems.

The second implementation method can be regarded as an application programmers' interface: a set of callable functions and procedures which manipulate (create, interrogate, delete, modify) the entities which support the application using a binding to a particular high-level programming language. A data repository is therefore

assumed, which contains instances of the information structures defined by the application, to which the user has access using the application programmers' interface.

The third level implies concurrent access to such a data repository, which could either be held centrally or distributed (with users requiring a high-speed network in order to access it). This brings further problems, but they are not unique to STEP.

The fourth method looks even further into the future than the third, but research work investigating the coupling of databases and knowledge bases has already started [12].

The remainder of this chapter concentrates on the physical file—the most familiar implementation method which has been used by IGES, SET and VDA-FS.

The STEP physical file is sequential and in free format (there is no column-based information, as in IGES, which can lead to white space being stored) and comprises two sections: header and data. It is thus a stream of characters, with no carriage-return characters. The syntax is formally defined using Wirth Syntax Notation [52] and it has a specified alphabet and tokens which enable it to be parsed. Part 21 also contains a formal mapping from EXPRESS to the file structure, which dictates how an instance of *any* EXPRESS schema will appear in a physical file. In these ways, the original requirements to avoid ambiguity and inefficiency, and to keep the implementation method separate from the the application, have been fulfilled.

### Wirth Syntax Notation

Wirth Syntax Notation (WSN) is a syntax description language which is used to define programming and other languages. This meta-language can be used to define its own syntax, as shown below:

```
syntax =      { production } .
production =  identifier "=" expression "." .
expression =  term { "|" term } .
term =        factor { factor } .
factor =      identifier
              | literal
              | "{" expression "}"
              | "[" expression "]"
              | "(" expression ")" .
identifier =  letter { letter } .
literal =     """" character { character } """" .
```

The vertical bar separates two alternatives; the curly brackets denote repetition—that is, their contents may be omitted or appear once or many times—and square brackets denote optionality: the contents may be included or omitted. Parentheses serve merely to group. In order to represent a double quote (") inside a literal, it is repeated.

## Wirth Syntax Notation example

A further example, that of the definition of the syntax of a real number, is also provided. It is both concise and unambiguous. It is instructive to compare this with the English definition in an early product data exchange specification.

```
real =
       [ sign ]
       digit { digit } "."
       { digit }
       [ ( "E" | "D" ) [ sign ] digit { digit } ] .
```

"A real constant may be either a basic real constant, a basic real constant followed by an exponent, or an integer constant followed by an exponent. A real constant may be of either single or double precision.... A double precision constant may be either a basic real constant followed by a double precision exponent, or an integer constant followed

by a double precision exponent. The form of a basic real constant is, in order, an optional sign, an integer part, a decimal point, and a fractional part. Both the integer part and the fractional part are strings of digits; either of these parts may be omitted, but not both...."

### Physical file tokens

The physical file syntax comprises a series of tokens and token separators. The tokens are keywords and the simple types (integer, real, string, entity name, enumeration and binary). The token separators are spaces, comments (delineated by `/*` and `*/`) and explicit print-control directives.

### Physical file sections

The first section of a STEP physical file, the header, contains the details shown in Figure 8.1.

The second section, the data, comprises a list of *entity_occurrences*. Each entity occurrence has a unique identifier and is an instance of a type specified in the application protocol, of which the file as a whole is an instance.

### Mapping from Express to the physical file

As stipulated by the original design goals of STEP, each implementation method is required to define how each EXPRESS construct may be instantiated. Figure 8.2 provides a summary of how each EXPRESS construct is mapped on to the physical file. It should be noted that many constructs do not need to be instantiated in the physical file.

### Example STEP file

Given the schema in Figure 5.3, there are an infinite number of instances of this type of information; one such instance is presented as a STEP physical file in Figure 8.3. The file has been pretty-printed, with use of white space, to aid the human reader: a single stream of characters would not have been useful, even if it were

| file_description | |
|---|---|
| *description* | informal description of contents |
| *implementation_level* | required implementation level of post-processor |
| **file_name** | |
| *name* | name of exchange file |
| *time_stamp* | file creation time and date |
| *author* | person who created the exchange file |
| *organisation* | organization with which author is identified |
| *preprocessor_version* | identification of the software used to create the file (name and version) |
| *originating_system* | the system from which the data originated |
| *authorisation* | who authorized the sending of the exchange file |
| **file_schema** | |
| *schema_identifiers* | identifies schema(ta) which specify the instances in the data section of the file (usually the single name of the long form of the application interpreted model: see Chapter 7) |

Figure 8.1. Contents of physical file header section.

printable on this paper size. Comments have also been added. It should be noted that the information in the file is incomplete.

| EXPRESS construct | physical file rendition |
|---|---|
| array | list |
| bag | list |
| binary | binary |
| boolean | enumeration |
| constant | no instantiation |
| derived attribute | no instantiation |
| entity | entity |
| entity as attribute | entity name |
| entity as supertype | no instantiation (if internally mapped) |
| entity as supertype | entity (if externally mapped) |
| enumeration | enumeration |
| function | no instantiation |
| integer | integer |
| inverse | no instantiation |
| list | list |
| logical | enumeration |
| procedure | no instantiation |
| real | real |
| remark | no instantiation |
| rule | no instantiation |
| schema | no instantiation |
| select | no instantiation |
| set | list |
| string | string |
| type | no instantiation |
| where rule | no instantiation |

Figure 8.2. Mapping from EXPRESS to the physical file.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('British royal family details (incomplete)'), '1');
FILE_NAME('ROYAL.STEP', '1992-08-19 T09:00:00',
          ('Jon Owen'), ('The University of Leeds', 'England'),
          'Hand-crafted systems, plc',
          'Generic Almanac 1992, London, England',
          'not approved for release');
FILE_SCHEMA(('ROYAL_EXAMPLE')) ;
ENDSEC;
DATA;
#1 = MALE (
     ('Albert','Frederick','Arthur','George'), /* first name */
     'Windsor', 'George VI',                   /* last name, title */
     (14, 12, 1895), (06, 02, 1952),           /* dates */
     (#2 /* and Margaret Rose */ ), .GREY.,    /* children, hair */
     $ ) ;                                     /* wife (not included) */
#2 = FEMALE ( ('Elizabeth', 'Alexandra', 'Mary'), 'Windsor',
    'Elizabeth II, by the Grace of God, of the United Kingdom of
    Great Britain and Northern Ireland and of her other Realms and
    Territories Queen, Head of the Commonwealth, Defender of the Faith',
    (21, 04, 1926), $, (#3, #8, #6, #7), .GREY., $, $ ) ;
#3 = MALE ( ('Charles', 'Philip', 'Arthur', 'George'), 'Windsor',
    'HRH The Prince of Wales', (14, 11, 1948), $, (#4, #5), .BROWN., #99 ) ;
#8 = FEMALE ( ('Anne', 'Elizabeth', 'Alice', 'Louise'), 'Phillips',
    'HRH The Princess Royal', (15, 08, 1950), $, (), .BROWN., $, $ ) ;
#6 = MALE ( ('Andrew', 'Albert', 'Christian', 'Edward'), 'Windsor',
    'HRH The Duke of York', (19, 02, 1960), $, (), .BROWN., $ ) ;
#7 = MALE ( ('Edward', 'Anthony', 'Richard', 'Louis'), 'Windsor',
    'HRH The Prince Edward', (10, 03, 1964), $, (), .BROWN., $ ) ;
#99 = FEMALE ( ('Diana', 'Frances'), 'Windsor',
    'HRH The Princess of Wales', (01, 07, 1961), $,
    (#4, #5), .FAIR., #3, 'Spencer' ) ;
#4 = MALE ( ('William', 'Arthur', 'Philip', 'Louis'), 'Windsor',
    'HRH Prince William of Wales', (21, 06, 1982), $, (), .FAIR., $ ) ;
#5 = MALE ( ('Henry', 'Charles', 'Albert', 'David'), 'Windsor',
    'HRH Prince Henry of Wales', (15, 09, 1984), $, (), .FAIR., $ ) ;
ENDSEC;
END-ISO-10303-21;
```

Figure 8.3. Example physical file (corresponding to schema in Figure 5.3).

### Physical file parsing

It is possible to parse this example file at two levels. The first way is to use the Wirth Syntax Notation of the physical file. This would provide a fast syntax check of the file, and so might be useful as a first coarse check. However, it would be parsing at the level of:

> "Find the identifier, equals sign, opening parenthesis, parameters, closing parenthesis, semicolon."

As long as this information was in place, and the parameters used were legal (integer, real, string, binary, enumeration, entity, or list) and were separated by commas, then the parser would accept the file.

The type of the parameters would not be important. However, given the schema, it is known that when a particular entity is encountered, then there should be a value to match the type of each of its explicit attributes inside the parentheses. Consequently, a more intelligent parser can be built that reads and checks the parameters based on a set of prescriptions derived automatically from the parent schema and the general mapping rules from EXPRESS to the physical file. In practice, far more software can be generated automatically from the EXPRESS language, which would result in a pre-processor, post-processor or syntax and semantics checker being constructed very quickly, with little hand-coding.

Given the description of the header section, the juxtaposition of the schema and an instance of it in the physical file, and the mapping rules outlined earlier, it can be seen how instances of the various EXPRESS constructs are realized on the physical file. It should be remembered that this is a clear-text encoding, although it is to be hoped that humans seldom, if ever, need to be able to examine such files directly. A binary-text encoding (which would necessarily require a software interface to read it for display purposes) would provide more compact files and perhaps resolve some of the outstanding concerns over file size, shown by the provision in STEP of short names for entities.

### Physical file printing

Part 21 also includes a section describing how to 'pretty-print' a STEP physical file (bearing in mind that it is a stream of characters with no carriage-return characters). There are commands which can be embedded explicitly in the file, and a set of implicit rules. The flavour of both of these is given below.

- The \F\ directive starts a new page.
- The \N\ directive starts a new line.
- Both \F\ and \N\ can appear where token separators are allowed, in strings and in binaries.
- All lines are left justified, with a maximum of 72 characters.
- A new line is required for each section, header section entity, comment and entity name.
- Tokens (other than strings and binaries) must not be broken.

### Physical file storage

Once a STEP physical file has been written by a pre-processor, it is likely that it will be sent for post-processing into a receiving system, probably at another site. (The exception is that it could be archived.) Although an increasing number of data exchanges are taking place over a network, many still require the transmission of the file on a physical storage medium. In the past, many such exchanges have failed simply because the medium was not labelled with the name of the utility used to write the medium. Part 21 includes a normative annex which defines how a physical file is represented on storage media: currently, on magnetic tape or $3\frac{1}{2}$-inch or $5\frac{1}{4}$-inch floppy disk. Others will no doubt be added as newer technologies become commonplace.

## Part 22: Standard data access interface

The SDAI is a functional specification which may be used as the interface between an application and instances of data in a form

specified by an EXPRESS schema. This is often referred to as the application programmers' interface (API). The data is stored in a repository, which may be a file, a working form (in-memory data structures) or a database. The repository may contain several models, the structure of each of which is defined by a single EXPRESS schema. Each schema is self-contained, as the 'long form' of the application interpreted model in an application protocol (see Chapter 7).

The functional specification includes several classes of operations:

*Environment:* initiation.

*Session:* handles the repositories in the session.

*Repository:* handles models within the repository.

*SDAI model:* creates entity instances, sets desired model access mode (read-write or read-only), checks references in the model, checks EXPRESS rules.

*Type:* checks subtype relationships.

*Application instance:* delete, check, manipulate instances and attributes of application schemata.

*Entity instance:* navigate instances of both SDAI-defined and application schemata.

*Aggregate:* manipulate, examine, modify instances of aggregates by using iterators.

There are also three EXPRESS schemata which form part of the functional specification.

1. Dictionary model: enables instances of application-schema entities, session-model entities and dictionary-model entities themselves to be made available to the application. It comprises the definitions in EXPRESS of those EXPRESS constructs which are relevant to the SDAI.

2. Session model: defines entities of the session that support the SDAI environment, such as its access mode, current state and error log. Instances are created and modified only as side effects of SDAI operations.

3. Abstract data model: this is not part of an implementation, but describes the objects referenced by the SDAI operations.

The functional specification is independent of any implementation language. It is realized (in an implementation) in one of several possible programming languages, in the same manner as GKS. Language bindings are being defined for C, FORTRAN, C++, Ada and Pascal. Further, there are two types of binding: late, which is independent of the EXPRESS schema being implemented, and early, which is dependent upon it. Some of the bindings will be standardized, probably as normative annexes to the part.

# 9

# Conformance testing methodology and framework

Many standards in computing have suffered from the lack of a conformance testing service at the time the standard was published. In the area of product data, CADCAM-Labor (part of Kernforschungszentrum Karlsruhe in Germany) provided one for VDA-FS in 1987, Association GOSET have had one for SET since 1991 and in the UK, CADDETC (part of the University of Leeds) provides a service for IGES. Whilst these services are most welcome, they became available several years after the relevant standard was published. This delay necessarily results in a period when users of CAE systems are reliant on the claims of vendors for their processors; if independent testing is required, the user has to undertake it himself.

This was recognized as a problem, so a conformance testing framework and methodology was designed into STEP from the beginning, in order to enable the timely provision of conformance testing services for STEP. The definition of conformance testing is

> "The testing of a candidate product for the existence of specific characteristics required by a standard in order to determine the extent to which that product is a conforming implementation."

Thus, it uses the standard as a metric against which implementations are measured. It presupposes that the standard itself is correct and useful, given the development and review process. It is also useful to contrast conformance with other types of testing, which can be undertaken once an implementation has been deemed to be conforming:

*Robustness:* the ability to handle files with incorrect entities and files with large models.

*Performance:* the usage of computing resources (memory, disk space, central processing unit ...).

*Interoperability:* the transfer from one system to another.

*User acceptance:* does it satisfy user requirements? (procurement).

The basis for conformance testing is provided by the conformance requirements and test purposes in the application protocol and abstract test suite, plus the conformance requirements in the implementation method. If it is not specified in the standard, it cannot be tested.

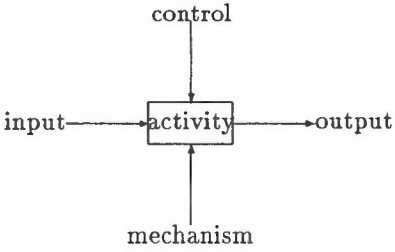
The advantages of conformance testing are twofold: the system users benefit by being able to ask a vendor for a conformance test report which documents the results of independent tests on the latter's processor, which saves tests being repeated by users individually throughout the world. The vendor benefits too, by having a set of standardized tests available either during the software development or its enhancement, and by not having to repeat tests during a user's procurement exercise, because they will already have been documented independently.

## The conformance assessment process

An overview of conformance testing is presented in Figure 9.2 as an IDEF0 diagram, which forms part of an IDEF0 model [13].

IDEF0 is a method for modelling activities. Each activity is represented by a box, which has inputs, controls, outputs and mechanisms (denoted by ICOM codes), as shown in Figure 9.1. Each activity may be decomposed into a number of sub-activities, thus producing a hierarchically organized set of diagrams.

The conformance assessment process comprises four stages. The first, preparation for testing, begins with the production of administrative information: what is to be tested, contact names, the organizations involved, and so on. Given the application protocol and the implementation method, the identification of the abstract test suite and the abstract test method is made. The PICS is used to




During the activity, under control, input is transformed into output by the mechanism.

Figure 9.1. IDEF0: key to layout.

determine which options in the application protocol have been implemented; this is most likely to be an indication of the relevant levels. The PIXIT documents additional information required by the testing laboratory in order to undertake conformance testing, such as the names of specific vendor constructs in the CAE system which correspond to those in the STEP application protocol. A set of abstract test cases is selected by the testing laboratory, parameter values are assigned, and an executable test suite is generated. This is then used by the client—at his own site—in order to prepare the system under test. This is known as prevalidation, and enables the client to test the processor himself. When he is satisfied, the final selection of the abstract test cases is made by the testing laboratory, parameter values (which may be different) are assigned, and the executable test suite is produced. At this stage, the scope of the conformance assessment process is frozen and cannot be changed subsequently. However, the client should be confident of the results because of the prevalidation phase.

During the test campaign, all executable test cases are run, and all inputs and outputs are recorded in the conformance log, for analysis and possible future audit. The analysis phase uses the verdict criteria from the abstract test case; each test is assigned a verdict of *pass*, *fail* or *inconclusive*. The first indicates that *all* criteria have been satisfied, whereas the second that *at least one* criterion has been violated. The third indicates that something unforeseen

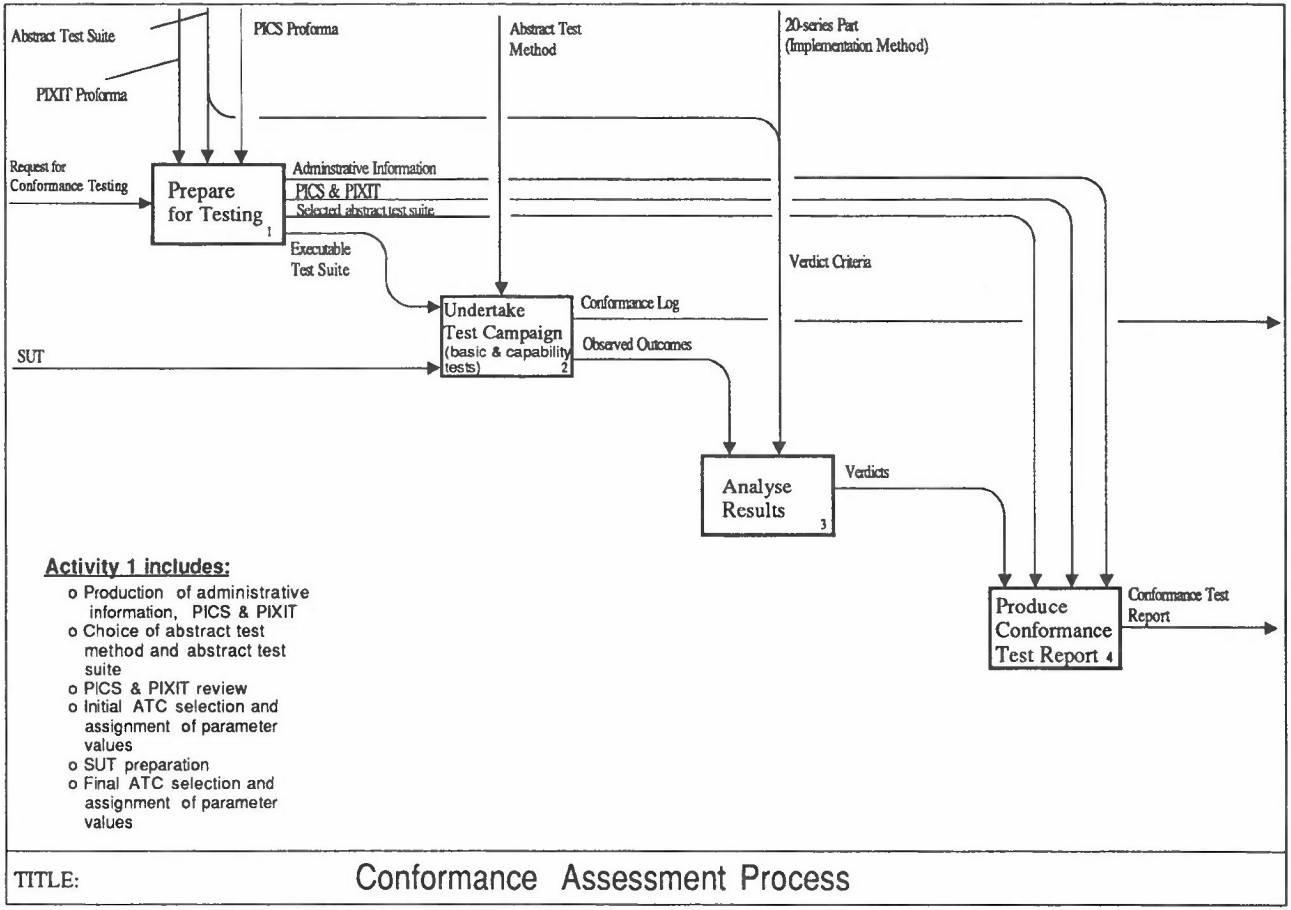



Figure 9.2. Overview of the STEP conformance assessment process.

happened (e.g. an error in the test case) and is used only rarely. A justification is provided with each fail verdict. The results are then synthesized into a conformance test report, the proforma for which is provided in Part 32.

## Enabling technologies

It is imperative that conformance testing is credible. Irrespective of when and where it is undertaken, the results should be consistent. Equally, the process needs to be auditable, to allow a review to demonstrate that procedures have been followed correctly. With this in mind, the key enabling technologies have been standardized for STEP. They cannot be reinvented at individual sites, which in any case would be inefficient. They include:

- The test purposes (in each abstract test suite).
- The PICS proforma (in each application protocol).
- The abstract test suite (in the 1200-series class).
- The abstract test method (in Part 34).
- The conformance requirements (in each application protocol and implementation method).

These enabling technologies provide a firm basis for conformance testing.

There are enabling technologies which it is either undesirable or impossible to standardize. The executable test suite is derived from the standardized abstract test suite by software, the requirements for which are documented in Part 34; the detailed procedures manuals are derived from Parts 32 and 34, and the software tools used during the conformance assessment process from the requirements in Part 34. The PIXIT is developed jointly by the testing laboratory and the client from guidance in Part 32. The entire process takes place under the aegis of a quality system, developed by a testing laboratory and accredited by an independent agency.

It should be noted that all controls in Figure 9.2 are provided directly and standardized by STEP, with the exception of the PIXIT proforma for which there is guidance provided in the standard.

### Overview of the 30-series class of parts

The 30-series of parts, all of which are prefaced by 'Conformance testing methodology and framework', comprise the following:

*Part 31:* General concepts.

*Part 32:* Requirements on testing laboratories and clients.

*Part 33:* Abstract test suites.

*Part 34:* Abstract test methods.

### Part 31: General concepts

Part 31, which serves as an introduction to the class, specifies a general methodology and framework for testing the conformance of an implementation of ISO 10303. It defines the fundamental concepts of conformance testing, provides a standardized terminology which is used throughout the rest of the class, and gives an overview of the conformance assessment process. It introduces abstract test suites and abstract test methods, which are the subjects of Parts 33 and 34 respectively, and describes how the PICS and the PIXIT are used during conformance testing. It also provides a framework for accreditation and certification, which is addressed later in this chapter.

### Part 32: Requirements on testing laboratories and clients

Part 32 defines who does what during conformance testing: personnel from the testing laboratory are responsible for some actions while personnel from the client's establishment are responsible for others. It should be noted that these roles and requirements are the same, irrespective of the nature of the client or the testing laboratory. The client is most likely to be an implementor or a supplier of a STEP software system, but may be a user, procurer, trade association, government body, or any other interested party. Similarly, the testing laboratory may be an implementor, a user or—most likely—an independent organization. Part 32 also provides the proforma for the conformance test report, so that all reports will be in the

same (standardized) format, thus enabling reciprocal arrangements between laboratories.

### Part 33: Abstract test suites

During conformance testing, a set of abstract test cases is used as the basis for testing the software for individual requirements stated in the standard. These abstract test cases are selected from the abstract test suite, defined as one of the 1200-series class of parts, corresponding to the application protocol that has been implemented. Part 33 defines the components of an abstract test suite and how they are used during conformance testing. The use of the language in which abstract test cases are written, EXPRESS–I, is expounded, and the links defined between the PICS questions and the abstract test suite, so that abstract test cases may be selected.

### Part 34: Abstract test methods

Part 34 describes the abstract test method for each of the STEP implementation methods. Some actions (such as information gathering) are common to all of the abstract test methods, but others are very different. For example, the creation of the model for a post-processor physical file implementation entails only reading in the file itself, but for other implementation methods, it requires a series of 'create' instructions in the CAE system. Whenever a new implementation method is standardized, Part 34 will need to be expanded.

### Accreditation and certification

It should be noted that the test suite and test method, both of which are standardized, are prefixed by the word "abstract". The test suite is 'abstract' in that it is used as the basis for generating, automatically by software, the executable test suite for each of the implementation methods. That is, a STEP physical file or a series of SDAI instructions can be generated from the abstract test suite. The test method is 'abstract' in that it describes the procedures and software required to undertake conformance testing, but does not provide either standardized software executables (such as a syntax checker for a physical file) or a detailed procedures manual: these

would be produced by test realisers and licensed by the testing laboratories for use in conformance testing under strict accreditation criteria.

Although ISO can standardize how conformance testing is undertaken, it cannot mandate that it occurs at all. It is up to users—particularly national governments and large consortia—to require that implementors do have their products conformance tested. However, the testing laboratories have to provide a quality, useful and cost-effective service. Conformance testing involves two organizations, the testing laboratory and the client's enterprise, but takes place within an infrastructure which includes several other bodies, as shown in Figure 9.3.

Accreditation is a formal process which ensures that a testing laboratory is competent to carry out specific types of tests. The term "laboratory accreditation" covers the recognition of both the technical competence and the impartiality of a testing laboratory. Accreditation is normally awarded following successful laboratory assessment and is followed by appropriate monitoring.

Certification may take place following conformance testing. A third party issues a certificate based on the conformance test report. This demonstrates that the identified implementation is in conformity with the STEP standards against which the implementation has been tested. (Those standards are usually an application protocol and an implementation method, although EXPRESS is another possibility.) This raises a number of liability issues.

It is important that, once a piece of software has been conformance tested, the results are recognized not only in the country in which conformance testing was undertaken, but worldwide (otherwise, conformance testing would need to be repeated in every country in which the software is sold). Reciprocal arrangements can be effected in a number of ways: unilaterally between testing laboratories, by national accreditation bodies recognizing each other, or by the criteria for issuing a certificate (based on the contents of a conformance test report) being coordinated by trade associations.

## Completeness and subsets

Whereas VDA-IS provides standard conversion rules for a proces-

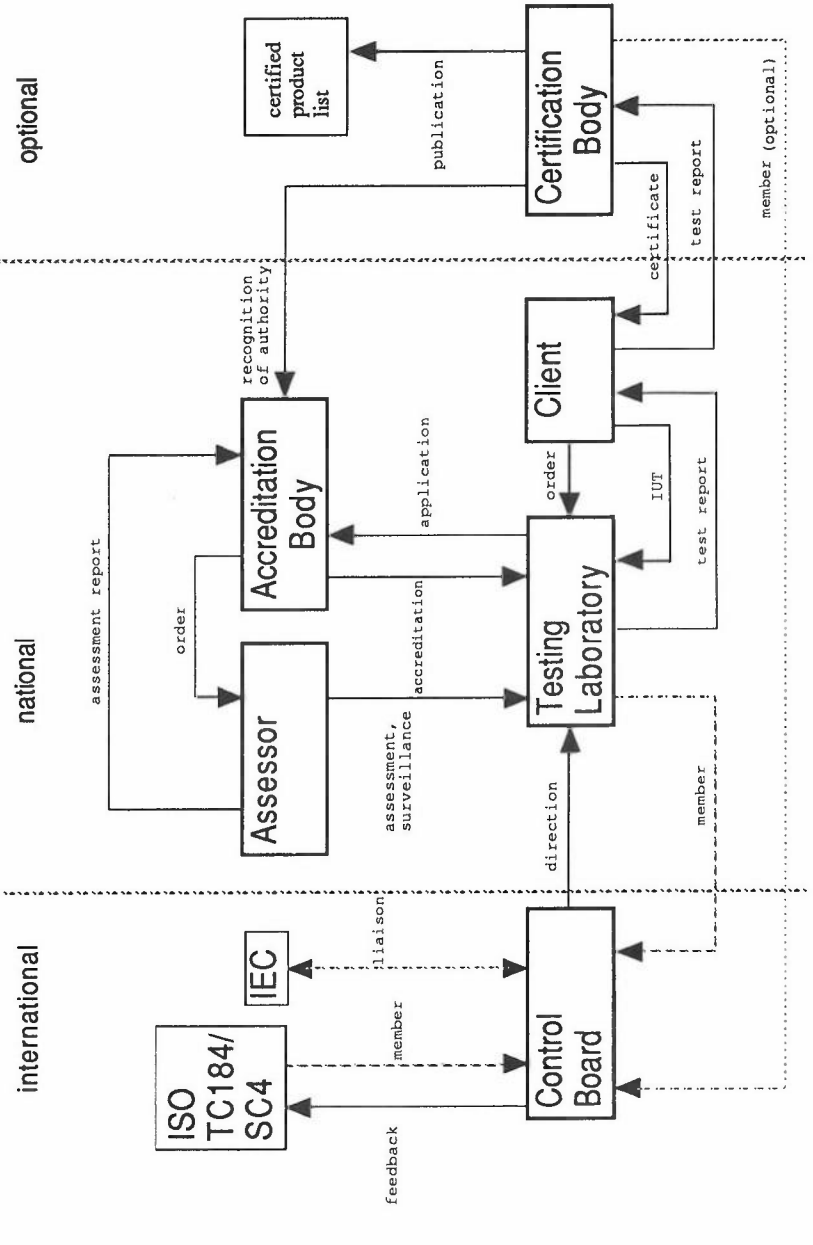



Figure 9.3. Infrastructure for conformance testing.

sor to apply when it encounters an entity from a level it does not support, none of the STEP application protocols currently does so. Consequently, vendor software needs to be able to present information to a user in the form given in the application protocol. This is *not* the same as using the EXPRESS definitions as the basis for data structures in the software, merely the presentation of that information: what is inside the software 'black box' remains the intellectual property of each vendor, and is what makes each system special.

One of the major issues for STEP has been what is usually referred to as 'completeness'. In practice, this is really two issues: not only completeness but also 'conversion'. As described in Chapter 7, the problem of completeness has been resolved by the introduction of levels within an application protocol. A vendor may support one or more of these levels, but in order to be a conforming implementation, each level must be supported completely. This precludes the *ad hoc* choice of individual entities from a particular level.

The issue of 'conversion' is best illustrated by an example. Consider the following representations of a circular arc:

- Centre, radius, start angle, swept angle.
- Centre, radius, start angle, finish angle.
- Centre, radius, start point, end point.
- Centre, radius, bulge factor.
- Ellipse (major axis = minor axis).
- General conic.
- Rational B-spline curve (with control points).
- Rational B-spline curve (flagged as circular).
- Polyline.

All would be acceptable if an application requires only a graphical rendering of the primitive. Other applications will require that the information is preserved in the form of the first representation, or some of the alternatives. All of the alternatives require some

conversion, even if it is only a 'simple' addition, which results in a loss of information. It is worthwhile considering, in a particular application context, what requirement the circular arc construction fulfills and, hence, what is the essence of a particular representation which makes it special or acceptable. If the application does allow any alternatives, then the conversion algorithm must be supplied in the standard; again, if it is not in the standard, it cannot be tested.

# 10

# The future of STEP

Although, at the time of writing, STEP has not yet been ratified as an international standard, many implementors and users are already benefitting from the technology and techniques arising from the standard. This concluding chapter presents some likely future developments, both in the STEP standard itself and its use.

The first area in which STEP is different from its predecessors is in the separation of the implementation method from the information content. This has resulted in a better-defined standard and an efficient physical file. A further key benefit is the development of implementation methods which will support the *same* application protocols. Even while this text has been in preparation, the SDAI has progressed at a fast rate, and it will undoubtedly soon become a standard itself. 'Database' implementations, based on the SDAI, are already becoming available, and it will only be a matter of time before commercially available systems link product information with a knowledge base.

The second area of difference is in the concept of application protocols, which enable application-specific views to be defined. Given the stable set of resource information models now available, and those which will be available shortly, there is potentially a wide range of application protocols which could be developed based on interpretations of these resources. Figures 10.1 and 10.2 give a list of application protocols currently being defined; the sheer diversity is striking.

Further resource models will undoubtedly be developed to support an even wider range of application protocols. The details of conformance testing are currently being documented. A second version of EXPRESS is scheduled for 1996, the requirements for which

*201* Explicit draughting

*202* Associative draughting

*203* Configuration controlled design

*204* Mechanical design using boundary representation

*205* Mechanical design using surface representation

*206* Mechanical design using wireframe representation

*207* Sheet metal die planning and design

*208* Life cycle product change process

*209* Design through analysis of composite and metallic structures

*210* Electronic printed circuit assembly, design and manufacture

*211* Electronics test, diagnostics and remanufacture

*212* Electrotechnical plants

*213* NC process plans for machined parts

*214* Core data for automotive design processes

*215* Ship arrangement

*216* Ship moulded forms

*217* Ship piping

*218* Ship structures

*219* Dimensional inspection process planning for coordinate measuring machines using tactile and video sensors

Figure 10.1. Application protocols with an allocated part number.

- composites – design to manufacturing
- electric/electronic
- product life cycle
- manufacturing process plans
- polymer testing
- sheet metal
- shipbuilding
- life cycle management
- product operation
- product procurement
- near net shape processing
- process plant functional data and its schematic representation
- ships electrical systems
- ships HVAC systems
- ships library parts
- ships outfit and furnishings
- printed circuit assembly manufacturing planning
- exchange of design and manufacturing product information for cast parts
- mechanical products definition for process planning using form features
- constructional steelwork

Figure 10.2. Proposed application protocols.

have already been determined.

There are two areas of concern. The first is the configuration control of application interpreted constructs from which application protocols and abstract test suites are built. The second is the possible inefficiency of the mapping of the conceptual EXPRESS schema directly on to the implementation method, perhaps requiring an so-called intermediate implementation or concrete schema. However, STEP is stable; evidence is provided by the implementations which have appeared already.

How, then, is industry likely to use the standard and its technology, and benefit from them?

The most obvious is the explicit use of STEP. Whereas enterprises currently exchange IGES, VDA-FS or SET physical files, in the future they will exchange STEP physical files. It will be possible to exchange a wider range of data with better-conditioned geometry within a framework of product data. Users and implementors should have greater confidence that transfers will work because of the availability of accredited conformance testing services at the time that STEP is published. Standardized interoperability testing will also become available, which will be able to provide answers to questions about the transfer of information between two specific systems which have already undergone conformance testing.

However, STEP will be adopted only if the benefits to enterprises are perceived. Either STEP will allow information they transfer at present to be transferred more reliably, or else it will facilitate the transfer of information which is not possible currently. In this area, the concept of application protocols is of prime importance.

There will necessarily be several—perhaps many—standard application protocols, usually divided into levels, for which conformance tested processors are available. However, enterprises may wish to exchange information in an area for which such an application protocol is not available. One solution would be for the two enterprises to produce an application protocol themselves, but not necessarily to standardize it (or perhaps to standardize at the national or industrial level, rather than the international). This would allow the automated software techniques discussed earlier to operate upon an EXPRESS schema interpreted from the integrated resources, en-

abling a pair of special-purpose translators to be produced quickly. The only difference would be that the EXPRESS upon which they were based would not be part of an ISO standard. Skill would be needed to determine the scope and requirements of the information to be exchanged, and to capture that in a non-standard application protocol. The majority of this work would have to be undertaken even if conventional hand-coded processors were to be produced and used. This technology can also be used to effect the migration from an existing product data standard to STEP; there are already examples of processors which can read IGES and 'convert' to STEP, in a particular context. Again, EXPRESS is the basis of these processors, with its facilities for derive rules and functions used both to undertake and document the conversions. This migration is essential if STEP is to be used; there is a massive amount of product data stored using existing standards.

Although this discussion has concentrated on the exchange of physical files, the same techniques can be applied to other implementation methods. Given an EXPRESS schema, it is easy to generate data structures and access software written in one of several high-level programming languages, or an object-oriented or relational database schema. Again, such implementations are already being demonstrated.

However, the major influence of STEP may not be its explicit adoption, either in the transfer of STEP physical files or in the use of EXPRESS schemata to drive software development, but in the adoption of techniques used during its development. Even if STEP were not to become an international standard, its impact in the field of computer-aided engineering would still be enormous. Many individuals have benefitted from the understanding gained in developing STEP, which has been reflected in their own enterprises, through activities such as:

- Finding out how and when particular product information is used in its life cycle.
- Using formal techniques to define the scope and requirements of a particular application.
- Defining the activities which take place within an enterprise which help to determine these requirements.

- Defining a framework for product data.
- Separating the application, logical and physical layers.
- Having the possibility of multiple implementation methods for particular information.

The key concept of the application protocol can be used to underpin all of these advances and also as the basis for software generation. It thereby provides 'standard' working practises for product data exchange and management. If these lessons have been learned and these techniques are applied, then STEP will indeed be a giant leap forward for computer-aided engineering.

# A

# Abbreviations

Appendix D, which follows, provides details of national standards bodies; their abbreviations are not included in this appendix.

*AIC:*
application interpreted construct

*API:*
application programmers' interface

*ASCII:*
American Standard Code for Information Interchange

*ATC:*
abstract test case (used in conformance testing)

*CAD:*
computer-aided design

*CADEX:*
CAD Exchange (an ESPRIT-funded project)

*CAD-LIB:*
CAD libraries (a European project)

*CAD*I:*
CAD*Interfaces (an ESPRIT-funded project)

*CAE:*
computer-aided engineering

*CAM:*
computer-aided manufacture

*CD:*
Committee Draft (in ISO)

*CDC:*
Committee Draft for Comment (in ISO)

*CGM:*
Computer Graphics Metafile

*DBMS:*
database management system

*DIS:*
Draft International Standard (in ISO)

*DP:*
Draft Proposal (in ISO—now termed a CD)

*EDIF:*
Electronic Design Interchange Format

*ESPRIT:*
European Strategic Programme for Research in Information Technology

*GKS:*
Graphical Kernel System

*HVAC:*
heating, ventilation and air-conditioning

*ICAM:*
Integrated Computer Aided Manufacturing (a US Air Force programme of work)

*IDEF:*
originally ICAM Definition; now Integrated Definition

*IEC:*
International Electrotechnical Commission

*IGES:*
Initial Graphics Exchange Specification

*IPIM:*
integrated product information model

*IS:*
International Standard (in ISO)

*ISO:*
International Organization for Standardization

*NC:*
numerical control

*NEDO:*
National Economic Development Office (UK)

*NIAM:*
Nijssen's Information Analysis Method

*NIST:*
National Institute of Standards and Technology (USA)

*PDDI:*
Product Definition Data Interface

*PDES:*
originally Product Data Exchange Specification; now Product Data Exchange using STEP (a US programme of work)

*PDES Inc:*
PDES Incorporated (a multinational group of companies contributing to the development of STEP)

*PDTAG:*
Product Data Technology Advisory Group (European Commission)

*PHIGS:*
Programmers Hierarchical Interactive Graphics System

*PICS:*
Protocol Implementation Conformance Statement (used in conformance testing)

*PIXIT:*
Protocol Implementation eXtra Information for Testing (used in conformance testing)

*PMAG:*
Project Management Advisory Group (in ISO)

*SADT:*
Structured Analysis and Design Technique

*SC:*
Sub-Committee (in ISO)

*SET:*
Standard d'echange et de transfert (France)

*SMMT:*
Society of Motor Manufacturers and Traders (UK)

*SPAG:*
Strategic Planning Advisory Group (in ISO)

*SPARC:*
Standards Planning and Requirements Committee (ANSI)

*STEP:*
Standard for the Exchange of Product Model Data

*SUT:*
system under test (used in conformance testing)

*TC:*
Technical Committee (in ISO)

*VDA:*
Verband der Automobilindustrie (Germany)

*VDA-FS:*
VDA Flächen Schnittstelle (Germany)

*VDA-IS:*
VDA IGES Subsets (Germany)

*VDA-PS:*
VDA Programm Schnittstelle (Germany)

*VDMA:*
Verband Deutscher Maschinen- und Anlagenbau e.V. (Germany)

*WD:*
Working Draft (in ISO)

*WG:*
Working Group (in ISO)

*XBF:*
Experimental Boundary File

# B

# ISO structure and development procedure

ISO has approximately two hundred technical committees in which *all* ISO standards are developed and maintained. Each technical committee has a number of sub-committees, in which particular nations participate. Each nation is designated as either a P-member (participating) or an O-member (observing), depending upon their level of activity. The former may vote whereas the latter may not; the status of each nation's membership may be regraded depending on its level of activity.

STEP is being developed in ISO TC184/SC4, to which several working groups and *ad hoc* committees report. The names of these committees are given in Figure B.1, with their conveners.

Note that WG1 was disbanded at the same time as STEP was divided into a series of parts, in June 1990, with the remaining committees being created at that time and subsequently. JWG9 is denoted as a *joint* working group because it is shared between ISO and IEC. The working groups meet three or four times a year at the same location, with the venue being in one of the participating countries.

Within the working groups, experts from the nations represented on SC4 collaborate to produce a document in their particular domain, and to build consensus. All documents are subjected to peer review, which continues until consensus has been reached within the working group. If the experts are producing a resource model, this has to be integrated with existing models, using the procedures developed in WG5. This integration is accomplished by WG4 in conjunction with the domain experts. Documents undergo tech-

| | | |
|---|---|---|
| TC 184 | Industrial Automation Systems and Integration | [François de Belenet (F)] |
| SC 4 | Industrial Data and Global Manufacturing Programming Languages | [Brad Smith (USA)] |
| WG 1 | (disbanded in June 1990) | [Jerry Weiss (USA)] |
| WG 2 | CAD standard parts | [Gerd Ehinger (D)] |
| WG 3 | Product modelling | [Barbara Warthen (USA)] |
| WG 4 | Qualification and integration | [Yuhwei Yang (USA)] |
| WG 5 | STEP development methods | [Bill Danner (USA)] |
| WG 6 | Conformance testing procedures | [Sheila Lewis (UK)] |
| WG 7 | Implementation specifications | [Jan van Maanen (UK)] |
| WG 8 | Industrial manufacturing management data | [Albert Colin (F)] |
| JWG 9 | Electrical and electronic product model data | [(vacant)] |
| Editing Committee | | [Nigel Shaw (UK)] |
| Strategic Planning Advisory Group | | [Jean-Pierre Letouzey (F)] |
| Project Management Advisory Group | | [(vacant)] |
| Application Protocols Co-ordination | | [Mark Palmer (USA)] |

Figure B.1. The ISO committees involved in STEP development.

nical editing and also have to conform to the ISO documentation format. Obviously, the earlier that the authors use this format and are aware of the scope and content of other parts, the easier and quicker this process will be.

When all of the working groups have accepted the document, it is submitted to the parent committee SC4 as a Committee Draft. It is then distributed for a four-month ballot period to those nations which participate on SC4. The national standards body in each nation may register a vote of

- Approve.
- Disapprove with comment (it is not acceptable to disapprove without providing comments).
- Abstain.

Any comments are distributed by the SC4 secretariat to the working group which developed the document, and these are collated and reviewed. The conveners of the working group and PMAG in conjunction with the chairman of SC4 determine if consensus has been reached on the document. If not, the document is returned to the working group, and a second (three-month) CD ballot will follow. It may be that a nation has voted to disapprove and submitted a comment which can be addressed by an editing change (rather than a technical one); alternatively, the comment itself may not be correct or be based on a misreading or misunderstanding of the document. In these cases, consensus is deemed to have been reached.

The document may then undergo editing change, after which it is submitted by SC4 to *its* parent committee, TC184, as a Draft International Standard (DIS). It is copyrighted by ISO and then distributed to *all* national standards bodies represented on ISO for a six-month ballot. At this stage, about ninety countries are involved. As a DIS, the document is considered to be technically stable, and only editorial changes may be made (by the ISO Central Secretariat) before it becomes an International Standard. In order to become an International Standard, the DIS requires a two-thirds majority of those countries voting which are represented as P-members on SC4, and no more than a quarter of the total votes cast may be negative.

Thus, if the P-members on SC4 were to vote by eleven votes to four to approve a STEP part, and there were no other votes cast, the part would pass on the first criterion but fail on the second.

If comments are submitted, the SC4 chairman and secretariat (with the project leader and relevant working group conveners, if required) in consultation with the ISO Chief Executive Officer, decide whether to prepare a new DIS or to refer the document back to the working group for further work. If an amendment is made to the original DIS to reflect a technical comment, then the new version of the DIS is circulated for a two-month ballot.

The history of voting on STEP and the status of the parts is given in Appendix C.

# C

# Summary of voting and current status

This appendix presents the summary of voting on STEP. The key to the symbols used in the voting summaries is in Figure C.1.

| | |
|---|---|
| $\surd$ | approved |
| $X$ | disapproved |
| – | abstained |
| | no information available |
| 0 | did not return a vote |
| $\surd$C | approved and also submitted some comments |

Figure C.1. Voting on STEP parts: key to symbols used.

Figure C.2 gives the results from the ballot of the first Draft Proposal, circulated after the Tokyo meeting in December 1988. Subsequently, ISO changed its procedures so that documents which had reached this level were termed 'Committee Drafts'. The results of voting on such CDs are summarized in Figure C.3 and those on Draft International Standards in Figure C.4. [1] Finally, a summary of the STEP parts and their status is provided in Figure C.5.

Note that a majority of nations voting to approve a CD as a DIS does not necessarily result in consensus being reached; this is determined as described in Appendix B.

It should also be noted that the names of the nations are given which were historically correct at the time of their membership of

[1] This last figure cannot be completed at the time of writing.

SC4: East Germany, West Germany, Czechoslovakia and the USSR have all undergone transformation since that time. This also applies to the details of the national standards bodies in Appendix D.

*Notes for first Draft Proposal ballot results:* Votes from Canada, France, West Germany, Japan, Netherlands, Norway, Sweden, the UK and the USA are documented. Each of the other nations represented on SC4 (Australia, Belgium, Brazil, Czechoslovakia, East Germany, Hungary, Italy, Poland, Switzerland and USSR) did not register a vote. A total of 1568 comments were submitted. Annexes A, B and C and clauses 4.5, 4.6 and 4.7 were circulated as SC4 documents N31 to N34 on 6th July 1988, in order to obtain advance comments. This material was modifed as a result and then distributed as part of the first Draft Proposal.

*Notes for* CD *ballot results:* Both East and West Germany registered $\surd$ for N64; after this time, Germany registered a single vote. N114 was circulated as volume two of Part 201 after N104 (11/10/91) was circulated as volume one; the balloting period commenced when N114 was available. N160 (16/10/92) was circulated as the corrected clause 6 of N152. Results given in *italics* indicate that consensus was reached and a DIS produced after the editorial comments were included.

*Note for* DIS *ballot results:* All nations which do not participate on TC184/SC4 but which are represented on ISO are also permitted to vote; these are shown as a summary only.

*Notes on summary of parts:* The critical parts (i.e. those intended for the initial release of STEP), as determined at the Paris WG and Gothenburg SC4 meetings in 1990, are marked with a †; those indicated form the original set of nine documents. Those marked with a ‡ are now also included in the minimum set required for the initial release. The class of abstract test suites is not shown; parts are numbered one thousand higher than their corresponding application protocol.

See next pages for the following information:

- Results of voting on the first STEP Draft Proposal.
- Results of voting on STEP Committee Drafts.
- Results of voting on STEP Draft International Standards.

| title | clause |
|---|---|
| scope | 1 |
| normative references | 2 |
| definitions | 3 |
| introduction | 4.1 |
| resource schema | 4.2 |
| types and functions | 4.3 |
| miscellaneous resources | 4.4 |
| geometry | 4.5 |
| topology | 4.6 |
| shape representation | 4.7 |
| features | 4.8 |
| shape representation interface | 4.9 |
| tolerancing | 4.10 |
| materials | 4.11 |
| presentation | 4.12 |
| product life cycle | 4.13 |
| applications | 4.14 |
| product manifestion | 4.15 |
| product structure | 4.16 |
| AEC applications | 4.17 |
| ship models | 4.18 |
| electrical applications | 4.19 |
| analysis applications | 4.20 |
| data transfer applications | 4.21 |
| conformance | 5 |
| EXPRESS | A |
| physical file structure | B |
| mapping to physical | C |
| | overall |
| | total comments |

Figure C.2. Results of voting on the first STEP Draft Proposal.

| CAN | FRA | FRG | JAP | NDL | NOR | SWE | UK | USA |
|---|---|---|---|---|---|---|---|---|
| √ | √C | X | X | X | √C | √C | X | X |
| – | √C | X | √ | X | X | X | √ | X |
| √C | √ | X | X | X | X | X | – | X |
| √ | X | √ | – | X | √C | √C | X | X |
| √ | √C | √ | – | X | √C | √C | – | X |
| √C | – | √C | – | X | X | X | X | X |
| √ | √C | X | – | X | X | X | X | X |
| √C | X | √C | √C | X | X | X | X | X |
| √C | √C | √C | √C | X | √C | √C | X | X |
| √ | X | √C | √C | X | √C | √C | X | X |
| X | X | √C | X | X | X | X | X | X |
| √C | X | X | – | X | X | X | X | X |
| √ | – | √C | X | X | X | X | – | X |
| √ | √C | X | X | X | – | – | X | X |
| – | X | √C | – | – | X | X | – | X |
| – | X | X | – | X | √C | √C | X | X |
| – | X | X | – | X | √C | √C | – | X |
| – | X | X | X | X | X | X | X | X |
| – | – | X | – | X | – | – | X | X |
| – | – | X | X | X | X | X | X | X |
| – | – | – | X | X | X | X | – | X |
| – | – | X | – | X | – | – | X | X |
| – | X | – | X | X | X | X | X | X |
| – | – | X | √C | X | X | X | – | X |
| – | √C | X | X | X | √ | √ | X | X |
| √C | √ | √C | – | X | √ | √ | X | X |
| – | – | √C | √C | X | X | X | X | X |
| – | X | X | √C | X | X | X | X | X |
| – | X | X | X | X | X | X | X | X |
| 23 | 36 | 77 | 137 | 82 | 200 (jointly) | | 232 | 781 |

| SC4 N#; date | | part | AUS | BEL | BRA | CAN | CHZ | FRA | GER | HUN |
|---|---|---|---|---|---|---|---|---|---|---|
| 64 | 13/07/90 | 11 | | √ | 0 | √ | 0 | √ | √ | 0 |
| 75 | 14/01/91 | 31 | √ | 0 | √ | √ | √ | √C | √C | √C |
| 78 | 25/03/91 | 21 | 0 | 0 | √ | √ | 0 | *X* | √C | √C |
| 83 | 31/05/91 | 11 | √ | 0 | √ | √C | 0 | √C | √C | √C |
| 87 | 21/06/91 | 42 | 0 | 0 | √ | √ | 0 | *X* | √C | √C |
| 93 | 08/08/91 | 43 | √ | √ | √ | √C | 0 | *X* | √C | 0 |
| 95 | 16/08/91 | 44 | √ | √ | √ | – | 0 | 0 | √C | 0 |
| 97 | 19/08/91 | 101 | √ | √ | √ | – | 0 | √ | √C | 0 |
| 99 | 20/09/91 | 203 | √ | 0 | 0 | √C | 0 | *X* | *X* | 0 |
| 102 | 11/10/91 | 46 | √ | 0 | √ | – | 0 | *X* | √C | √ |
| 105 | 11/10/91 | 41 | √ | 0 | √ | √C | 0 | *X* | *X* | √ |
| 111 | 14/01/92 | 31 | 0 | 0 | √ | 0 | | √ | √C | 0 |
| 114 | 12/03/92 | 201 | √ | 0 | 0 | √C | 0 | *X* | *X* | 0 |
| 134 | 15/05/92 | 1 | √ | 0 | 0 | 0 | 0 | *X* | √C | √C |
| SC4 N#; date | | part | AUS | BEL | BRA | CAN | | FRA | GER | HUN |
| 139 | 25/08/92 | 41 | 0 | 0 | 0 | √ | | *X* | √C | 0 |
| 141 | 25/08/92 | 42 | √ | 0 | 0 | √C | | *X* | √C | 0 |
| 143 | 25/08/92 | 43 | √ | 0 | 0 | *X* | | *X* | √C | 0 |
| 145 | 25/08/92 | 44 | √ | 0 | 0 | √C | | – | √C | 0 |
| 147 | 25/08/92 | 46 | √ | 0 | 0 | √C | | *X* | √C | 0 |
| 149 | 25/08/92 | 101 | √ | 0 | 0 | – | | √C | √C | 0 |
| 152 | 14/09/92 | 203 | √ | √ | 0 | *X* | | *X* | √C | 0 |
| 154 | 15/09/92 | 1 | 0 | √ | 0 | √C | | √C | √C | 0 |
| 167 | 23/11/92 | 201 | | | | | | | | |

Figure C.3. Results of voting on STEP Committee Drafts.

| ITA | JAP | NDL | NOR | POL | SWE | SWI | UK | USA | USSR | total |
|---|---|---|---|---|---|---|---|---|---|---|
| X | X | X | X | 0 | X | √ | X | X | 0 | 6-7 |
| X | √C | √C | X | 0 | X | 0 | √C | √C | 0 | 11-3 |
| √ | X | √ | 0 | 0 | √C | 0 | √C | X | 0 | *8-3* |
| √C | √C | X | √C | | √C | | X | √C | 0 | *11-2* |
| 0 | √ | √C | √C | √ | √C | 0 | √C | X | 0 | *10-2* |
| – | √C | X | √C | 0 | √C | 0 | X | X | | 8-4 |
| – | √C | X | √ | 0 | X | 0 | √C | X | | 7-3 |
| – | √C | X | √ | √ | √C | 0 | √C | X | √ | 11-2 |
| – | √C | X | 0 | √ | X | 0 | X | X | √ | 5-6 |
| – | √ | √ | 0 | √ | √C | 0 | X | X | √ | 9-3 |
| – | √C | X | √C | √ | X | 0 | X | X | √ | 8-6 |
| – | √C | 0 | √C | | √C | 0 | √ | √C | 0 | *8-0* |
| – | √C | X | 0 | √ | X | 0 | X | X | √ | 5-6 |
| √ | 0 | √C | √ | | √C | X | X | √C | 0 | 8-3 |

| ITA | JAP | NDL | NOR | RUS | SWE | SWI | UK | USA | | total |
|---|---|---|---|---|---|---|---|---|---|---|
| √ | √C | √C | √C | 0 | √C | X | X | √C | | 8-3 |
| √ | √C | √C | √C | 0 | √C | X | √C | √C | | 10-2 |
| √ | √C | √ | √C | 0 | √C | X | X | √C | | 8-4 |
| √ | √C | √ | √C | 0 | √C | X | X | √C | | 9-2 |
| √ | X | | √C | 0 | √C | X | X | √C | | 7-4 |
| √ | √C | √ | √C | 0 | √C | X | √C | √C | | 10-1 |
| – | √C | X | X | 0 | X | X | X | √C | | 5-7 |
| – | √C | √C | √C | 0 | √C | √C | √C | √C | | 11-0 |

| SC4 N#; date | part | AUS | BEL | BRA | CAN | FRA | GER | HUN | ITA |
|---|---|---|---|---|---|---|---|---|---|
| N151 27/08/92 | 11 | √ | √C | √ | | √C | √C | √ | √ |
| N157 15/09/92 | 31 | √ | √ | √ | | *X* | √ | √ | √ |
| N193 | 1 | | | | | | | | |
| N194 | 44 | | | | | | | | |
| N195 | 43 | | | | | | | | |
| N196 | 101 | | | | | | | | |
| N197 | 41 | | | | | | | | |
| N198 | 42 | | | | | | | | |
| N200 | 203 | | | | | | | | |
| N204 | 21 | | | | | | | | |
| N217 23/08/93 | 46 | | | | | | | | |
| | 201 | | | | | | | | |

Figure C.4. Results of voting on STEP Draft International Standards.

| JAP | NDL | NOR | ROM | RUS | SWE | UK | USA | SC4 total | non-SC4 |
|---|---|---|---|---|---|---|---|---|---|
| √ | √ | √ | – | | √C | √C | √C | 13-0 | 2-1 |
| √ | √ | √ | √ | | √ | √C | √C | 13-1 | 3-0 |

| part | title | status |
|---|---|---|
| † 1 | Overview and fundamental principles | DIS; SC4 N193; ballot closes 20/11/93 |
| | **Description methods:** | |
| † 11 | The EXPRESS language reference manual | DIS ballot closed 19/05/93; 15-1 |
| 12 | The EXPRESS–I language reference manual | SC4/WG5 N40 11/92 |
| | **Implementation methods:** | |
| † 21 | Clear text encoding of the exchange structure | DIS; SC4 N204; ballot closes 29/01/94 |
| 22 | Standard data access interface | CD ballot expected mid/93 |
| | **Conformance testing methodology and framework:** | |
| † 31 | General concepts | DIS ballot closed 19/05/93; 16-1 |
| 32 | Requirements on testing laboratories and clients | SC4/WG6 N61 01/07/93 |
| 33 | Abstract test suites | SC4/WG6 N64 23/08/93 |
| 34 | Abstract test methods | SC4/WG6 N65 27/08/93 |
| | **Integrated resources:** | |
| † 41 | Fundamentals of product description and support | DIS; SC4 N197; ballot closes 03/12/93 |
| † 42 | Geometric and topological representation | DIS; SC4 N198; ballot closes 03/12/93 |
| ‡ 43 | Representation structures | DIS; SC4 N195; ballot closes 20/11/93 |
| ‡ 44 | Product structure configuration | DIS; SC4 N194; ballot closes 20/11/93 |
| 45 | Materials | WD; SC4/WG3 N241 25/06/93 |
| † 46 | Visual presentation | DIS; SC4 N217; 23/08/93 |
| 47 | Shape variation tolerances | WD; SC4/WG3 N247 12/07/93 |
| 48 | Form features | WD; SC4/WG3 N102 02/01/92 |
| 49 | Process structure, property and representation | CD; SC4 N223 24/08/93 |
| † 101 | Draughting | DIS; SC4 N196; ballot closes 03/12/93 |
| 102 | (Ship structures) | (now deleted) |
| 103 | Electrical applications | |
| 104 | Finite element analysis | CD; SC4 N192; 15/03/93 |
| 105 | Kinematics | SC4/WG3 N204; 28/01/93 |

| Application protocols: | | |
|---|---|---|
| † 201 | Explicit draughting | CD; SC4 N167; ballot closed 23/02/93 |
| 202 | Associative draughting | CDC; SC4 N189; period closed 08/04/93 |
| ‡ 203 | Configuration controlled design | DIS; SC4 N200; ballot closes 15/01/94 |
| 204 | Mechanical design using boundary representation | SC4/WG3 N209; 28/01/93 |
| 205 | Mechanical design using surface representation | SC4/WG3 N190; 15/10/92 |
| 206 | Mechanical design using wireframe representation | SC4/WG3 N114 12/06/92 |
| 207 | Sheet metal die planning and design | CDC; SC4 N159 16/10/92 |
| 208 | Life cycle product change process | CDC |
| 209 | Design through analysis of composite and metallic structures | CDC expected soon |
| 210 | Electronic printed circuit assembly, design and manufacture | CDC; SC4 N218; period closes 01/12/93 |
| 211 | Electronics test, diagnostics and remanufacture | |
| 212 | Electrotechnical plants | CDC; SC4 N201; period closed 15/07/93 |
| 213 | NC process plans for machined parts | CDC; SC4 N203; period closed 25/07/93 |
| 214 | Core data for automotive design processes | CDC expected soon |
| 215 | Ship arrangement | |
| 216 | Ship moulded forms | |
| 217 | Ship piping | |
| 218 | Ship structures | WD; 06/01/93 |
| 219 | Dimensional inspection process planning for coordinate measuring machines using tactile and video sensors | |

Figure C.5. Status of STEP parts.

# D

# Standards bodies

## Australia:

**SAA**
Standards Australia
PO Box 458
AUS - North Sydney
NSW 2059
Australia

**phone:** + 61 2 963 4111
**fax:** + 61 2 959 3896

## Belgium:

**IBN**
Institut belge de normalisation
Avenue de la Brabançonne 29
B - 1040 Bruxelles
Belgium

**phone:** + 32 2 734 92 05
**fax:** + 32 2 733 42 64

## Brazil:

**ABNT**
Associação Brasileira de Normas Técnicas
Av 13 de Maio, $n^o$13 – 28$^o$ andar
Caixa Postal 1680
CEP: 20.003 - Rio de Janeiro - RJ
Brasil

**phone:** + 55 21 210 31 22

## Canada:

**SCC**
Standards Council of Canada
350 Sparks Street, Suite 1200
CDN - Ottawa, Ontario K1P 6N7
Canada

**phone:** + 1 613 238 3222
**fax:** + 1 613 995 4564

## Czechoslovakia:

**CSN**
Urad pro normalizaci a mereni
Federal Office for Standards and Measurement
Václavské náměsti 19
CS - 113 47 Praha 1
Czechoslovakia

**phone:** + 42 2 235 21 52
**fax:** + 42 2 26 57 95

## France:

**AFNOR**
Association française de normalisation
Division Informatique
Tour Europe
Cedex 7
F - 92049 Paris la Défense
France

**phone:** + 33 1 42 91 55 55
**fax:** + 33 1 42 91 56 56

## Germany:

**DIN**
Deutsches Institut für Normung
Burggrafenstraße 6
Postfach 11 07
Berlin
Germany

**phone:** + 49 30 26 011
**fax:** + 49 30 260 12 31

## Hungary:

**MSZH**
Magyar Szabványügyi Hivatal
Pf 24
H - 1450 Budapest 9
Hungary

**phone:** + 36 1 118 30 11
**fax:** + 36 1 118 51 25

## ISO:

**ISO**
International Organization for Standardization
Case Postale 56
CH - 1211 Geneva 20
Switzerland

**phone:** + 41 22 749 01 11
**fax:** + 41 22 733 34 30

## Italy:

**UNI**
Ente Nazionale Italiano di Unificazione
Piazza Armando Diaz 2
I - 20123 Milano
Italy

**phone:** + 39 2 72 00 11 41
**fax:** + 39 2 869 01 20

## Japan:

**JISC**
Japanese Industrial Standards Committee
Standards Department
Agency of Industrial Science and Technology
Ministry of International Trade and Industry
1-3-1 Kasumigaseki Chiyoda-ku
J-Tokyo 100
Japan

**phone:** + 81 3 501 92 95 or 96
**fax:** + 81 3 580 14 18

## Netherlands:

**NNI**
Nederlands Normalisatie Instituut
Kalfjeslaan 2
Postfach 50 59
NL - 2600 GB Delft
Nederlands

**phone:** + 31 15 69 03 90
**fax:** + 31 15 69 01 90

## Norway:

**NSF**
Norges Standardiseringsforbund
Postboks 7020
Homansbyen
N - 0306 Oslo 3
Norway

**phone:** + 47 22 46 60 94
**fax:** + 47 22 46 44 57

## Poland:

**PKNMiJ**
Polish Committee for Standardization, Measures and Quality Control
Ul. Elektoralna 2
PL - 00 - 139 Warsawa
Poland

**phone:** + 48 22 20 54 34
**fax:** + 48 22 20 66 46

## Sweden:

**SIS**
Standardiseringskommissionen i Sverige
Box 3295 Tégnergatan 11
S - 10366 Stockholm 6
Sweden

**phone:** + 46 8 613 52 00
**fax:** + 46 8 11 70 35

## Switzerland:

**SNV**
Swiss Association for Standardisation
Kirchenweg 4
Postfach
CH - 8032 Zürich
Switzerland

**phone:** + 41 1 384 47 47
**fax:** + 41 1 384 47 74

## UK:

**BSI**
British Standards Institution
2 Park Street
London
W1A 2BS
United Kingdom

**phone:** + 44 71 629 9000
**fax:** + 44 71 629 0506

## USA:

**ANSI**
American National Standards Institute
1430 Broadway
New York
NY 10018
USA

**phone:** + 1 212 354 3300
**fax:** + 1 212 302 1286

## USSR:

**GOST**
USSR State Committee for Product Quality Control and Standards
Leninsky Prospekt 9
SU - Moskva 117049
USSR

**phone:** + 7 095 236 40 44

# Bibliography

[1] G. Baker, J.C. Kelly and W. Conroy (1992) "IGES 5.1 recommended practices guide". Iges/Pdes Organization, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.

[2] K. Blacker, ed. (1989) "Advancing the application of IGES". Published by the Society of Motor Manufacturers and Traders Ltd, London.

[3] M.S. Bloor, J.R. Dodsworth and J. Owen (1984) "CAD interchange of data: guidelines for the use of IGES. Phase 1: two dimensional scale and non-scale drawings with text". Published by NEDO in July 1984. Available from the National Economic Development Office, Millbank Tower, Millbank, London SW1P 4QX.

[4] M.S. Bloor, J.R. Dodsworth, J. Owen and P.F. Stewart (1986) "CAD interchange of data: guidelines for the use of IGES. Phase 2: two dimensional drawings with associated data". Published by the CAD-CAM Data Exchange Technical Centre in July 1986.

[5] CGI (1991) "Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — functional specification". ISO 9636. Part 1: Overview, profiles and conformance. Part 2: Control. Part 3: Output. Part 4: Segments. Part 5: Input and echoing. Part 6: Raster.

[6] CGM (1987) "Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information (CGM)". ISO 8632. Part 1: Functional specification. Part 2: Character encoding. Part 3: Binary encoding. Part 4: Clear text encoding.

[7] EDIF (1992) "Electronic Design Interchange Format, version 2 9 0". Electronic Industries Association, Engineering Department, 2001 Eye Street N.W., Washington DC 20006, USA.

[8] GKS (1985) "Information processing systems — Computer graphics — Graphical Kernel System (GKS) functional description". ISO 7942.

[9] GKS-3D (1988) "Information processing systems — Computer graphics — Graphical Kernel System for three dimensions (GKS-3D) functional description". ISO 8805.

[10] R.J. Goult and P.A. Sherar, eds. (1990) "Improving the performance of neutral file data transfers". Esprit Project 322 (CAD*I) Volume 6. ISBN 3-540-53427-X, published by Springer-Verlag, Heidelberg, Germany.

[11] H. Grabowski, ed. (1991) "Advanced modelling for CAD/CAM systems". Esprit Project 322 (CAD*I) Volume 7, published by Springer-Verlag, Heidelberg, Germany.

[12] K. Higa, M. Morrison, J. Morrison and O.R.L. Sheng (1992) "An object-oriented methodology for knowledge base / database coupling". *Communications of the ACM*, Volume 35, Number 6, (99–113), June 1992.

[13] IDEF0 (1981) "ICAM Architecture Part II, Volume IV — Function Modeling Manual (IDEF0)". Report number AFWAL-TR-81-4023, Volume IV, June 1981. Available from: Mantech Technology Transfer Center, WL/MTX, Wright-Patterson Air Force Base, OH 45433-6533, USA.

[14] IDEF1X (1985) "Integrated Information Support System (IISS). Volume V — Common data model subsystem. Part 4

— Information Modeling Manual — IDEF1 Extended.". Report number AFWAL-TR-86-4006, Volume V, Part 4, November 1985. Available from: Mantech Technology Transfer Center, WL/MTX, Wright-Patterson Air Force Base, OH 45433-6533, USA.

[15] IGES (1991) "The Initial Graphics Exchange Specification (IGES) version 5.1". Iges/Pdes Organization, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.

[16] ISO (1987) "ISO Technical Report 9007: Information processing systems — concepts and terminology for the conceptual schema and the information base". ISO Central Secretariat, Geneva, Switzerland.

[17] U.I. Kroszynski, B. Palstroem, E. Trostmann and E.G. Schlechtendahl (1989) "Geometric data transfer between CAD systems: solid models". *IEEE Computer Graphics and Applications*, Volume 9, Number 5, (57–71), September 1989.

[18] L. Lamport (1986) "LaTeX: a document preparation system". Published by Addison-Wesley Publishing Company, Reading, Massachusetts, USA.

[19] M.H. Liewald (1985) "Initial graphics exchange specification: successes and evolution". *Computers and Graphics*, Volume 9, Number 1, (47–50), 1985.

[20] MIL-D-28000A (1992) "Digital representation for communication of product data: IGES application subsets and IGES application protocols". Available from: Standardised documents order desk, Building 4D, 700 Robbins Avenue, Philadelphia, PA 19111-5094, USA.

[21] T.W. Olle, H.G. Sol and A.A Verrijn-Stuart, eds. (1982) "Information systems design methodologies: a comparative review". Published by North-Holland.

[22] J. Owen and M.S. Bloor (1987) "Neutral formats for product data exchange: the current situation". *Computer-Aided Design*, Volume 19, Number 8, (436–443), October 1987.

[23] M.E. Palmer (1993) "Guidelines for the development and approval of STEP application protocols, version 1.1". ISO TC184/SC4/WG4 N66.

[24] PDDI (1984) "Product Definition Data Interface". Report number SS 5601 30200. July 1984. Available from: Materials Laboratory (AFWAL/MLTC), Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, OH 45433, USA.

[25] PHIGS (1989) "Information processing systems — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS)". ISO 9592. Part 1: Functional description, Part 2: Archive file format, Part 3: Clear text encoding of archive file.

[26] PHIGS-PLUS (1992) "Information processing systems — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS)". ISO 9592. Part 4: Plus lumière und surfaces.

[27] G. Pierra (1990) "An object oriented approach to ensure portability of CAD standard parts libraries". Proceedings of EuroGraphics 90, (205–214). September 1990.

[28] M.J. Pratt (1985) "IGES — the present state and future trends". *Computer-Aided Engineering Journal*, Volume 2, Number 4, (130–133), August 1985.

[29] M. Raflik and B. Pätzold, eds. (1990) "An approach to an engineering database (version 4.0)". Esprit Project 322 (CAD*I) Volume 5. ISBN 3-540-53383-4, published by Springer-Verlag, Heidelberg, Germany.

[30] P. Rehwald (1985) "VDA-FS — an interface to transfer surface description data between CAD systems". *Computers and Graphics*, Volume 9, Number 1, (69–70), 1985.

[31] E.G. Schlechtendahl, ed. (1988) "Specification of a CAD*I neutral file for CAD geometry, wireframes, surfaces, solids (version 3.3)". Esprit project 322 (CAD*I) Volume 1. ISBN 3-540-50392-7, published by Springer-Verlag, Heidelberg, Germany.

[32] E.G. Schlechtendahl, ed. (1989) "CAD data transfer of solid models". Esprit project 322 (CAD*I) Volume 3. ISBN 3-540-51826-6, published by Springer-Verlag, Heidelberg, Germany.

[33] R. Schuster and R. Anderl (1986) "CAD-Interfaces: a tool for integrating production processes". Proceedings of MICAD'86, (722–741).

[34] R. Schuster, D. Trippner and M. Endres, eds. (1990) "CAD*I drafting model". Esprit project 322 (CAD*I) Volume 4. ISBN 3-540-52051-1, published by Springer-Verlag, Heidelberg, Germany.

[35] SET: z 68-300 (1989) "Industrial automation — external representation of product definition data — data exchange and transfer standard specification version 89-06". L'association française de normalisation (afnor), Tour Europe Cedex 7, 92080 Paris-la-Défense, France.

[36] B.M. Smith (1983) "IGES: a key to CAD/CAM systems integration". *IEEE Computer Graphics and Applications*, Volume 3, Number 8, (78–83), November 1983.

[37] B.M. Smith et al (1986) "A reporting of the PDES initiation activities". National Bureau of Standards, Gaithersburg, MD 20899, USA.

[38] D. Thomas, J. van Maanen and M. Mead, eds. (1989) "Specification for exchange of product analysis data". Esprit project 322 (CAD*I) Volume 2. ISBN 3-540-51579-8, published by Springer-Verlag, Heidelberg, Germany.

[39] D. Tsichritzis and A. Klug, eds. (1978) "The ANSI/X3/-SPARC DBMS framework: report of the study group on database management systems". *Information Systems*, Volume 3, (173–191), published by Pergamon Press Ltd, Oxford, UK.

[40] VDA-FS (1986) "DIN 66301". Deutsches Institut für Normung e.V., Burggrafenstraße 6, D1000 Berlin 30.

[41] VDA-FS (1987) "VDA surface interface version 2.0". Verband der Automobilindustrie e.V (VDA), W-6000 Frankfurt am Main, Westendstraße 61, Germany.

[42] VDA-IS (1989) "VDA IGES subset version 2.0". Verband der Automobilindustrie e.V (VDA), W-6000 Frankfurt am Main, Westendstraße 61, Germany.

[43] VDA-PS (1987) "VDA Programm Schnittstelle". Verband der Automobilindustrie e.V (VDA), W-6000 Frankfurt am Main, Westendstraße 61, Germany.

[44] G.M.A. Verheijen and J. van Bekkum (1982) "NIAM: an information analysis method". See: T.W. Olle, H.G. Sol and A.A Verrijn-Stuart (1982), (537–589).

[45] J. Weiss (1986) "STEP functional requirements". ISO TC184/SC4 N30 (2nd May 1988).

[46] P.R. Wilson (1987) "A short history of CAD data transfer standards". *IEEE Computer Graphics and Applications*, Volume 7, Number 6, (64–67), June 1987.

[47] P.R. Wilson (1987) "Information and/or data?". *IEEE Computer Graphics and Applications*, Volume 7, Number 11, (58–61), November 1987.

[48] P.R. Wilson (1987) "Information modelling". *IEEE Computer Graphics and Applications*, Volume 7, Number 12, (65–67), December 1987.

[49] P.R. Wilson (1989) "PDES STEPs forward". *IEEE Computer Graphics and Applications*, Volume 9, Number 2, (79–80), March 1989.

[50] P.R. Wilson (1990) "STEP ballot results". *IEEE Computer Graphics and Applications*, Volume 10, Number 3, (79–82), May 1990.

[51] P.R. Wilson, I.D. Faux, M.C. Ostrowski and K.G. Pasquill (1985) "Interfaces for data transfer between solid modeling systems". *IEEE Computer Graphics and Applications*, Volume

5, Number 1, (41–51), January 1985. See also: Volume 5, Number 3, (83), March 1985.

[52] N. Wirth (1977) "What can we do about the unnecessary diversity of notation for syntactic definitions?". *Communications of the ACM*, Volume 20, Number 11, (822–823, November 1977.

# Index