-- --

IEEE P1003.0 Draft 13 – September 1991

program interface of the TP resource manager protecting that resource could include the ability to reduce the amount of money in the drawer (by pushing it out of the machine). A transaction application program using two TP resource managers (a conventional database manager that keeps track of the balance in accounts, and the teller machine's cash drawer TP resource manager) would want to insure that the two TP resource managers decrement both the cash and the balance of the correct account in the context of a single transaction (i.e., with the ACID properties.)

The TP Resource Manager API, then, generally provides the following services:

(1) Ability to increment or decrement a valuable resource by a certain amount.

(2) Ability to determine the amount of a valuable resource that remains.

Specific capabilities for the very wide variety of specific TP resource managers, cannot, of course, be documented here.

4.10.4.2 External Environment Interface Services

When two or more machines are involved in the same transaction, the following service is required:

— The ability for two application platforms to interoperate with each other (pass along global transaction identifiers, participate with each other in commitment process, participate with each other in recovery).

4.10.4.3 Interapplication Software Entity Services

4.10.4.4 OLTP Resource Management Services

The services listed in this subclause are not provided by application program interfaces or external environment interfaces.

— Management Services — Ability to control the operation of the transaction processing services, including the ability to assign dispatching priorities to individual transaction application programs.

— Monitoring Services — Ability to collect data on resource utilization for purposes such as performance analysis and accounting (data on utilization of the transaction processing services resources: processes, connection pools, . . .).

— Modeling Services — Ability to predict the system resources needed to process a given transaction processing workload.

— Directory/Namespace Services — Ability to map names to addresses.

— Recovery/Restart Services — Ability to recover and restart transactions involving one or more transaction application programs using one or more TP Resource Managers.

— Test Services — Ability to automatically generate tests for workload simulation, etc.

— System Configuration Services — Ability to replace or add transaction application programs without the need to shut down the execution environment.

4.10.4.5 Services Provided at the TM/TPRM SII

HLJ : RJG asked for this to be an ‘‘informative subclause.’’ The only way to do that is to make it a NOTE:

NOTE: For application portability it is not required that the application platform actually consist of Transaction Managers and TP Resource Managers, but in the new age of Open Systems, there are clear advantages in doing so. Two advantages seem obvious: the ability to ‘‘mix and match’’ Transaction Managers and TP Resource Managers from different vendors; and the ability of a user to construct his/her own TP Resource Manager to manage particular critical resources. A market has already developed for ‘‘plug compatible’’ TMs and TPRMs. All TPRMs at this printing are Database type TPRMs. It is expected that a market will also develop for Communications type TPRMs. It is not at all clear that the industry will develop other types of widely applicable TPRMs, thus forcing users to develop their own. Users could use the interface described here to do such development work.

This NOTE very briefly describes the services that should be provided at such an interface.

The TM/TPRM interface must provide the ability of TMs and TPRMs to: register with each other; obtain recovery status information; pass along transaction identifier information; rollback, prepare to commit, and commit the transaction. The interface must provide for the needs of the full range of transaction processing including distributed transaction processing with multiple TPRMs.

Finally it should be noted that the industry recognizes the need for standardization of components as well as the standardization of portability and interoperability. At least one industry group is standardizing and several vendors are implementing products utilizing an interface as described here.

4.10.5 Standards, Specifications, and Gaps

There are currently three transaction processing standards development activities, either completed or in the draft stage. Table 4-11 summarizes the service requirements provided by the various standards.

Table 4-12 summarizes the applicability of the various standards to the various programming languages supported by the POSIX Open System Environment.

4.10.5.1 Current Standards in the POSIX OSE

4.10.5.1.1 International Standards

None.

4.10.5.1.2 Regional Standards

None.

Table 4-11 – Transaction Processing Standards Activities

Service	Specification	Subclause	
Transaction Demarcation	Emerging (P1003.11 {*ref*})	4.10.5.?	D
Communications	Gap	4.10.5.?	D
Terminal Communications	Gap	4.10.5.?	D
Program Scheduling	Gap	4.10.5.?	D
Message Queuing	Gap	4.10.5.?	D
EEI Services	Emerging (ISO/IEC DIS 10026-1,2,3 {*ref*})	4.10.5.?	
Management	Gap	4.10.5.?	D
Monitoring	Gap	4.10.5.?	D
Modeling	Gap	4.10.5.?	D
Directory/Namespace	Gap	4.10.5.?	D
Recovery/Restart	Gap	4.10.5.?	D
Test	Gap	4.10.5.?	D
System Configuration	Gap	4.10.5.?	D
TM/TPRM SI	Emerging (P1003.11 {*ref*})	4.10.5.?	D

Table 4-12 – Transaction Processing Standards Language Bindings D

Standard	N/A	LIS	Ada	APL	BASIC	C	C++
ISO/IEC DIS 10026-1 {*ref*}	•						
ISO/IEC DIS 10026-2 {*ref*}	•						
ISO/IEC DIS 10026-3 {*ref*}	•						
POSIX. 11 {*ref*}		•				•	

Standard	COBOL	C-LISP	Fortran	Pascal	PL/1	Prolog	
ISO/IEC DIS 10026-1 {*ref*}							D
ISO/IEC DIS 10026-2 {*ref*}							D
ISO/IEC DIS 10026-3 {*ref*}							D
POSIX. 11 {*ref*}							D

NOTE: N/A — Language bindings are not appropriate for this standard.

LIS — Language-independent specification is available.

Ada, APL, BASIC, — Language-dependent specifications exist.

4.10.5.1.3 National Standards

None. D

4.10.5.2 Emerging Standards in the POSIX OSE

OSI Distributed Transaction Processing (DTP)

ISO/IEC DIS 10026-1 {*ref*}
ISO/IEC DIS 10026-2 {*ref*}
ISO/IEC DIS 10026-3 {*ref*}

These standards, developed by ISO/IEC JTC 1/SC21/WG5, deal expressly with the OSI services and protocols for transaction mode communications in an OSI environment.

These standards provide for some of the communications services described in 4.10.4.1.

POSIX.11 POSIX Transaction Processing

POSIX.11 {*ref*}

The POSIX.11 working group, formed in 1989, is chartered to work on a profile for Transaction Processing within the POSIX OSE. In the process of developing that profile, it has identified a number of gaps in the standards coverage and is in the process of proposing base standardization activities to address those gaps. Specifically, P1003.11 is currently working on the following services identified earlier:

— Transaction Manager (TM) Services provided at the Transaction API.

— Services provided at the Transaction Manager/TP Resource Manager (TM/TPRM) SII. A typical TPRM is a database manager (e.g., SQL).

RJG: The above list should be adjusted later to reflect the then current state of .11's activities (in terms of approved PARs.) If no PAR is approved by the beginning of formal ballot, I'll fix this prose and move it to the Gap section.

POSIX.11 is working to assure that POSIX Transaction Processing work is consistent with the emerging work of OSI DTP (cited above), certain ongoing work of X/Open TP, several related POSIX activities (POSIX.1 {2}, POSIX.4 {*ref*}, POSIX.8 {*ref*}) and the work of ANSI X3T5.5 (RPC).

4.10.5.3 Gaps in Available Standards

4.10.5.3.1 Standards/Specifications Outside the POSIX OSE

Existing standards and emerging standards do not adequately address all the requirements identified earlier. While POSIX.11 is addressing some of the gaps, there are many other gaps still not being addressed by formal standards committees. Most notable is the work of X/Open TP. While not formally a standards making body, it is addressing most of the gaps, and its output will be potentially useful as the basis of a formal standard.

X/Open TP

This group published an ''Online Transaction Processing Reference Model'' in 1987 and in 1990 published ''Preliminary Specification—Distributed Transaction Processing: The XA Specification.'' The group is studying the use of OSI DTP, two-phase commit, and global transaction identifiers. The group is also actively exploring APIs in support of peer-to-peer distributed transactions.

The work of this group addresses several of the services addressed in this clause, including transaction demarcation and conversation services.

Consideration is also being given to allowing alternative interoperability standards including proprietary mechanisms. Additional APIs are being defined by X/Open TP to facilitate this.

4.10.5.3.2 Unsatisfied Service Requirements

Other than the work of X/Open TP, the following areas are not currently being addressed by standardization activities: communications, terminal communications, program scheduling, message queueing, management, monitoring, modeling, directory/namespace, recovery/restart, test, and system configuration.

4.10.6 OSE Cross-Category Services

Not applicable.

4.10.7 Related Standards

CCR

The following standards relating to commitment are related to the ISO/IEC DIS 10026 series and are referenced in DIS 10026:

ISO/IEC DIS 9804-3 {*ref*}
ISO/IEC DIS 9805-3 {*ref*}

See 4.3 for more information.

Remote Procedure Call

ECMA-127 {*ref*}

See 4.3 for more information.

SQL Standard Database Language

The following standards for SQL also provide transaction demarcation services for relational database access:

ANSI X3.135.1 (ISO 9075, FIPS 127)
ANSI X3.168

See 4.4 D

4.10.8 Open Issues

None.

4.11 Software Development Environments

Responsibility: Don Folland

4.11.1 Overview and Rationale

Software Development Environments are dealt with as a particular application area needing special attention for the following reasons:

— The domain of Software Development Environments is one of prime importance. Software development is a major area of expenditure for government and large commercial organizations.

— The need for standardization is being driven not only by the SDE vendors and users, but also by the Independent tool developers who want to get their tool products on as many vendor platforms as possible.

— The SDE domain calls not only for portability, but also for particular integration and interoperability requirements.

— The domain is primarily of interest to that user community that has large complex software development requirements, but it is also of interest to all application areas as software development is an enabling technology for all applications.

Software engineers seek more powerful assistance to improve productivity and quality in the software development process. Considered opinion currently favors Project Support Environments (PSE) underpinned in such a way that the facilities are capable of being implemented on different machines. A PSE needs a base holding information such as specifications, designs, code, schedules, configuration plans, tests, etc., to support the developers. The interface between the base and the tools must ensure portability of the tools. Again, these tools will be supported by relevant language standards.

Certain design methodologies themselves have been modeled formally to establish their degree of rigor and self-consistency. Function Point Analysis is one method of measuring software systems and computing productivity that is increasing in use. It measures inputs, outputs, and entities accessed to determine transaction size; it gauges technical complexity by reference to 19 characteristics. These are combined to give a measure of systems size. Productivity is the ratio of system size in function points to the effort required to produce or maintain the system.

Generally, software support for the development process is in its infancy and effective metrics have not yet been developed.

4.11.2 Scope

The problem domain is complex software development, from the generation of an idea to the delivery and ongoing support of a solution product set.

Thus, an SDE may include some or all of the following:

(1) Software Development Life Cycle
   (a) Requirements analysis
   (b) Logical design
   (c) Physical design
   (d) Functional and interface specification
(2) Activity support
   (a) Prototyping
   (b) Program development and testing
   (c) Quality assurance and regression testing
   (d) Generation of user documentation
   (e) User training
   (f) Problem report tracking and maintenance
   (g) Maintenance and tracking of schedules
(3) Configuration Management
   (a) Automatic version management
   (b) Integrity management
   (c) Traceability
(4) Project Management
(5) Data Administration
   (a) Access control

In the context of developing software for a POSIX Open System Environment, design will take account of portability and interoperability requirements. The SDE tools themselves should be portable. The software development activities may be provided with a large set of tools and applications. The SDE must provide the necessary support for the integration of all of these tools.

4.11.3 Reference Model

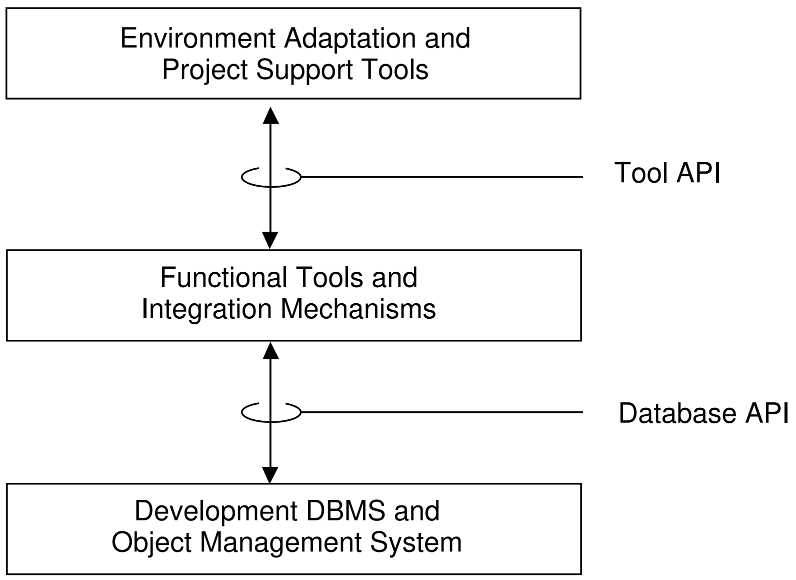

Figure 4-22 – Software Development Model

In this clause the conceptual view of software development is related to the POSIX Reference Model (Figure 3-1). The software developer's view is shown in Figure 4-22. The tools used to develop software can be viewed as applications in their own right in the context of the POSIX Reference Model, requiring the same services from the platform as for Database Management.

In the Software Development Model, the Environment Adaptation and Project Support Tools ''layer'' provides the essential link between the programmer, designer or analyst, the design method, and the development infrastructure. At this level are provided the tools and applications that are unique to the project or methodology; e.g., project management workbench. It requires support from a consistent human-computer interface to the Functional Tools.

The Functional Tools and Integration Mechanisms embrace the essential tool set to enable software developers to build software. It includes simple tools such as editors, tools for tool-building, and integration mechanisms. There will be tools for Configuration Management, Version Management, and System Administration. It is not within the scope of this guide to discuss these in detail.

The whole software development environment is underpinned by essential management systems, such as object management system, a data dictionary, a user interface management system, and environment management. A database will frequently be established to hold specifications, designs, configuration plans, etc.

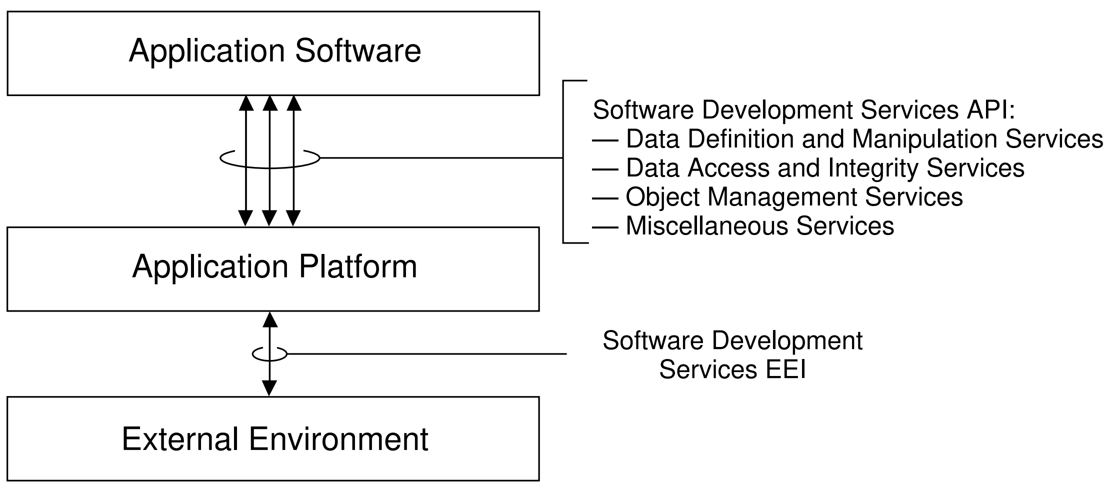

Figure 4-23 – Software Development Reference Model

In the POSIX Open System Environment, the software development model can be incorporated into the POSIX Reference Model as in Figure 4-23. The model shows that the tools and services required by the software developer are part of the POSIX Open System Environment and are available through the POSIX OSE API.

4.11.4 Services Requirements

Software developers, i.e., designers, analysts, and programmers, use software applications to facilitate the complex task of software development. A tool will require services from the application platform and will frequently require support from another application in the application set. There are many possible implementations of tool sets. Descriptions of these are beyond the scope of this guide.

4.11.4.1 Application Program Interface Services

The services required at the API are essentially similar to those described for Database Management in 4.4.4.1; i.e., Data Definition and Manipulation, Data Access, Data Integrity, and such Miscellaneous Services as Data Dictionary.

4.11.4.2 External Environment Interface Services

A consistent human-computer interface to the tool set is required. Some of the
programmer's tool set will be explicitly focused on windowing services (such as 4.6 4
and 4.7) and provide assistance to develop software with improved usability.

Resource data formats must be specified in order to ensure effective information interchange [for example, CASE Data Interchange Format (CDIF)], for which standards are currently under development under the aegis of the CDIF Technical

Committee (see also 4.11.5.2 and 4.5).

Protocol services are required for the transport of data (see 4.3).

4.11.4.3 Interapplication Software Entity Services

Many of the tools depend for interface between one another upon the data dictionary/repository, which is a key software component and which may conceptually be regarded as part of the Applications Platform. Included in this category will be utilities for servicing the DBMS, such as recovery, reorganization, and restructure:

— Object management system

— User interface management system

— Database management system

— Transaction processing management system

Details of these management systems may be recorded in the data dictionary/repository.

4.11.4.4 Software Development Resource Management Services

These services are generally not visible to the programmer or software developer at the Tools API, usually being provided by the tool building and other software development utilities.

4.11.5 Standards, Specifications, and Gaps

This subclause describes current accepted standards that are relevant to this area in addition to the language standards in 4.1.5 and the database standards in 4.4.5.

4.11.5.1 Current Standards in the POSIX OSE

This subclause briefly identifies the current standards in this area.

DEF: The following provides place holders for further text to be inserted – assistance required please.

4.11.5.1.1 International Standards

Labeling and File Structure of Magnetic Media

The following standards refer to the labeling of magnetic media and for the file structure on such media to facilitate information interchange:

Labeling of magnetic tape	ISO 1001 {*ref*}
Labeling of cassette and cartridge	ISO 4341 {*ref*}

Table 4-13 – Software Development Standards Activities

Service	Specification	Subclause	
Miscellaneous Services:			
Labeling of magnetic tape	ISO 1001 {*ref*}	4.11.5.?	D
Labeling of cassette and cartridge	ISO 4341 {*ref*}	4.11.5.?	D
Labeling of flexible disks	ISO 7665 {*ref*}	4.11.5.?	D
Volume and file structure for flexible disks	ISO 9293 {*ref*}	4.11.5.?	D
Volume and file structure for CD-ROM	ISO 9660 {*ref*}	4.11.5.?	D
Documentation symbols and flowchart conventions	ISO 5807 {*ref*}	4.11.5.?	D
Documentation of applications	ISO 6592 {*ref*}	4.11.5.?	D
Program flow for sequential files	ISO 6593 {*ref*}	4.11.5.?	D
Data descriptive file for information interchange	ISO 8211 {*ref*}	4.11.5.?	D
Program constructs and conventions	ISO 8631 {*ref*}	4.11.5.?	D
User documentation	ISO 9127 {*ref*}	4.11.5.?	D

Labeling of flexible disks	ISO 7665 {*ref*}
Volume and file structure for flexible disks	ISO 9293 {*ref*}
Volume and file structure for CD-ROM	ISO 9660 {*ref*}
Data descriptive file for information interchange	ISO 8211 {*ref*}

DEF: The above-mentioned standards might be more suitably called out in Richard Scott's section 4.5.

Software Documentation

There are several standards dealing with documentation to assist with the task of software development, and therefore potentially facilitating programmer and designer portability, as well as user documentation.

Documentation symbols and conventions for data, program and system flowcharts, program network charts, and system resources charts	ISO 5807 {*ref*}
Guidelines for the documentation of computer-based application systems	ISO 6592 {*ref*}
Program flow for processing sequential files in terms of record groups	ISO 6593 {*ref*}
Program constructs and conventions for their representation	ISO 8631 {*ref*}
User documentation and cover information for consumer software packages	ISO 9127 {*ref*}

4.11.5.1.2 Regional Standards

ECMA has approved ECMA-149 as the standard for the Portable Common Tool Environment (PCTE).

4.11.5.1.3 National Standards

To Be Provided

4.11.5.2 Emerging Standards in the POSIX OSE

This subclause describes the activities currently in progress to further standardize this area.

4.11.5.2.1 International Standards

To Be Provided

4.11.5.2.2 Regional Standards

To Be Provided

CASE Data Interchange Format (CDIF)

The CDIF Technical Committee is developing a data interchange format to serve as an industry standard for exchanging information between Computer-Aided Software Engineering (CASE) tools. CDIF is an EIA-endorsed initiative. It assumes that two or more tools may interface asynchronously with each other and will transfer information from one to another via ''CDIF files.'' The types of information that may be contained in these files is defined by the CDIF Conceptual Models.

Portable Common Tool Environment (PCTE)

ECMA TC33 has responsibility for the development and maintenance of PCTE. The committee formed a Task Group in 1988 to develop a Reference Model which would assist the standardization process. Such a model has been developed totally independent of PCTE, and is described in ECMA Technical Report 55. The model provides a way to describe, compare, and contrast CASE environment frameworks.

4.11.5.2.3 National Standards

To Be Provided

4.11.5.2.4 National Standards

To Be Provided

4.11.5.3 Gaps in Available Standards

4.11.5.3.1 Standards and Specifications Outside the POSIX OSE

To Be Provided

4.11.5.3.2 Unsatisfied Service Requirements

To Be Provided

4.11.6 OSE Cross-Category Services

Not applicable. D

4.11.7 Related Standards

To Be Provided

4.11.8 Open Issues

— Relationship between methodology and formats

Section 5: POSIX OSE Cross-Category Services

Responsibility: Fritz Schulz

The POSIX reference model defines a set of conceptual system building blocks that collectively describes the Open System Environment. Each building block provides a specific set of interfaces for access to their associated facilities and services. There is another class of services and requirements, however, that may influence and/or impact the basic architectural building blocks; these are referred to as OSE Cross-Category Services.

An OSE Cross-Category Service is a set of tools and/or features that, when applied, may have a direct affect on the operation of one or more of the Open System Components, but it is not in and of itself a standalone OSE component. Examples of OSE Cross-Category Services include internationalization, security and privacy, administration, etc. Internationalization has a number of attributes that influence multiple OSE components; supporting multiple coded character sets, for example, will affect end-user interfaces, operational message input and output, screen display, data collating sequences in programming languages and database systems, etc.

This section will deal with the general characteristics of OSE Cross-Category Services as applied to the OSE architectural components and to the profiles and domains that characterize application environments. The specific impact/influence of an OSE Cross-Category Service will be described in the appropriate subclause of Section 4 that deals with individual OSE Components.

Initially, this section will address Internationalization, Security and Privacy, and System Administration; however, it is anticipated that other OSE Cross-Category Services will be identified as the concept is applied to the model.

This section describes issues that should be considered in writing profiles, and is organized so that subclauses for each OSE Cross-Category Service points to, and addresses issues adjacent to each of the service categories identified in Section 4.

These issues defined areas that need to be traded off to arrive at balanced solutions for a specific profile. It is expected that the specific trades would be made by the profiler, but that this clause could give guidance for trading and could also be used to accumulate lessons learned.

5.1 Internationalization

Responsibility: Ralph Barker D

5.1.1 Overview and Rationale

Historically, information systems intended for use within a particular national or cultural market have been designed specifically for the requirements of that market. If the vendor or developer was based in a country other than that of the target market, this was typically accomplished through substantial re-engineering the features of an existing system designed for some other country, and doing so at considerable cost. As the developer desired to market the system in additional countries, the process of re-engineering was repeated for each new national or cultural market. Application software developers were faced with the same problem. The very nature of this style of development produced little concern for portability across national or cultural boundaries, or interoperability between them. Users or organizations that needed to operate in multiple national or cultural markets typically did so with multiple, generally incompatible, information processing systems.

The interfaces provided by the POSIX Open Systems Environment (POSIX OSE) can be generalized, however, through the use of internationalization, to extend across national and cultural boundaries. Such a model provides the foundation for international portability of application software, increased user portability, and enhanced interoperability and data exchange capabilities. The task of internationalization is to ensure that the services provided by the POSIX OSE, and the interfaces between such services, are specified in such a way that they can be easily used all over the world. Additionally, as the user is likely to require services from any or all of the service categories of the POSIX OSE, internationalization impacts all areas of the POSIX OSE, and should be viewed as an OSE Cross-Category Service. Since the internationalization aspects of general OSE services and application program interface (API) services are similar for all of the POSIX OSE service categories, they are discussed here rather than repeating them in each of the services sections within this guide.

The ability of the service categories of the POSIX OSE to support multiple natural languages, and the underlying cultural conventions, is a two step process. These two steps are generally referred to as ‘‘internationalization’’ and ‘‘localization.’’ First, the interfaces between the service categories are generalized, so that they are not oriented to the requirements of any particular natural language or set of cultural conventions (internationalization). Then, facilities are provided by the POSIX OSE that allow the user to select the desired natural language and cultural conventions (localization). Tools are provided to facilitate this process.

Within this context, cultural conventions, while discussed more fully later in this clause, may be viewed as various aspects of how information is presented to the user. Different cultures, for example, use different formats for dates and numeric values and use different currency symbols. The interfaces provided by the POSIX OSE should allow the information to be presented to the user in the appropriate format as well as the appropriate natural language.

5.1.2 Scope

The POSIX OSE provides services that are necessary to support users, irrespective of their particular natural language or cultural conventions. While it is not expected that every implementation of the POSIX OSE would provide support for all possible natural languages and cultural conventions, the specification of the services and the interfaces within the POSIX OSE should not preclude such support. In addition to the service and interface requirements described here, it should be noted that internationalization is affected by a number of elements that are beyond the scope of this guide. Actual implementations of the internationalized POSIX OSE, for example, may need to consider the impact of multiple sets of governmental and regulatory agencies, international data communication standards and other elements which are presently not specified within the POSIX OSE, such as data portability between localized information processing systems.

Service requirements differ from country to country and even between users within one country. Many users, for example, may require the simultaneous support of multiple natural languages and cultural convention sets. Therefore, the basic internationalization requirement within the POSIX OSE is to provide a set of services and interfaces that allow the user to define, select, and change between different culturally related application operating environments supported by the particular implementation. Specifically:

— The POSIX OSE must provide the means of adjusting the output of specific functions and utilities to support different natural languages, cultural conventions and character sets as may be required by the supported natural languages.

— A user should have the capability to select an internationalized user environment that specifies a particular set of data presentation characteristics, including cultural conventions, character sets and native language.

— An implementation of the POSIX OSE should be able to concurrently support different applications functioning in different internationalized user environments, supplying different sets of natural languages, cultural conventions and character sets for different users.

— The capability of supporting different internationalized user environments, and the associated natural languages, cultural conventions and character sets, should not require any changes to the logic of existing application programs.

— The effect of the user selecting a new internationalized user environment, and its associated natural language, cultural conventions and character set, should be transparent to application programs.

— The model should be flexible, to support future extensions and requirements.

5.1.3 Reference Model

Internationalization is an OSE Cross-Category Service, spanning all OSE service categories. While various reference models have been used in published technical papers to depict internationalization issues, the internationalization services described in this clause conform to the POSIX OSE Reference Model.

5.1.4 Service Requirements

The POSIX OSE must provide services on different levels: general service requirements to be satisfied for any requesting program; API service requirements to be satisfied at the application program interface for a specific program; and a set of tools to support the localization of systems and applications. This subclause (5.1.4) will discuss these different service requirements in detail. In examining these service requirements, it is helpful to draw a distinction between those services which are required to support the portability of an application platform across cultural boundaries, and those services which are required to support the portability of an application across one or more sets of cultural conventions which may be supported on a single application platform.

5.1.4.1 General Service Requirements, Application Platform

Internationalization requirements are focused on support and handling of:

— Character sets and data representation

— Cultural conventions

— Natural language support

5.1.4.1.1 Character Sets and Data Representation

The character set for the English language can easily be satisfied by the standard ASCII character set (American Standard Code for Information Interchange). The ASCII code uses 7 bits to uniquely identify each of the 95 available characters. For European and American languages beside English, the number of local characters is much larger. The far-east requirements for thousands of pictograms add yet another dimension to the coding rules and techniques.

Different standards address the methods by which the local character repertoires can be coded for unique identification. While replacement of seldom-used characters in the 7-bit codings can support a single additional language besides English, 8-bit coding schemes are used to satisfy multiple languages concurrently by assigning an additional 96 graphic characters to the available repertoire. An example is ISO 8859-1 (the extended ASCII code), which can support all of western Europe, America, Australia, and other English speaking countries all over the world. For Eastern Europe, Greece, Russia, Arabia, and many other countries, other 8-bit codes are defined. Japan, China, Korea, and Taiwan have so many characters in their repertoire that 16 bits are needed to identify them clearly. Work is under way to develop a multi-octet character set with up to 32 bits per coded character; this method will allow concurrent use of all possible languages in the same application.

Because different coding schemes are used, it is important that the application platform have the potential capability of supporting all of them. It is also important that the application platform has the capability to represent (display, print) the data correctly. It is also important that an application be able to determine in which coded character set data items are stored on disk or tape. Otherwise, it is impossible for the application to interpret the data correctly. Currently the user must control the consistent use of the same coded character set within an application, but in the future the application platform should be able to provide identification methods for the coded character sets used for data storage, processing, communication, and presentation. It might also be advantageous for the application to be able to prohibit users from updating data stored in one coded character set with data in another coded character set since this would immediately corrupt data bases or flat files. Therefore it may be necessary in the future to provide a method of announcing the coded character set in which data are stored, processed, communicated, and presented.

The general service for support of character sets and data representation in an international environment are:

(1) Coded character set independence: the ability of the application platform to input, store, manipulate, retrieve, communicate, and present data independent from the coding scheme used. This includes 7-bit, 8-bit, 16-bit, and multi-octet coded character sets.

(2) Character set repository: the ability of the application platform to maintain and access a central character set repository. This repository contains all coded character sets used throughout the platform and specifies relevant information about them:

- Code format: the repository contains information, if characters are coded in 7 bits, 8 bits, 16 bits, or any other format.
- Data class definition: the definition that a character is considered numeric, alpha, etc., by the programming languages. This classification can vary for the same character from country to country.
- Collating rules: different character sets have different coding for characters. Thus, comparison of strings of such coded characters must follow rules defined for the specific character set. Culturally dependent additional collating rules are discussed in 5.1.4.1.2.
- Lower- to uppercase mapping: this defines the rules of mapping, if for a specific character no upper- or lowercase is available. Examples are the lower case umlauts which do not have uppercase representations in Switzerland; the uppercase forms are A, O, or U, respectively, followed by a lowercase ''e''.
- Escapement rules: some languages like Hebrew and Arabic are written from right to left; numbers within text in these languages are written from left to right. It is necessary to store these escapement