-- --

IEEE P1003.0 Draft 13 – September 1991

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

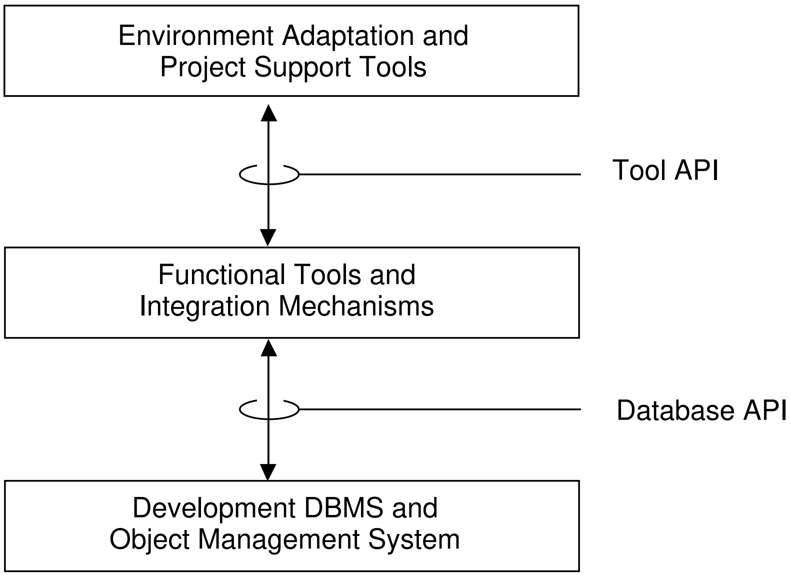

Figure 4-22 – Software Development Model

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.

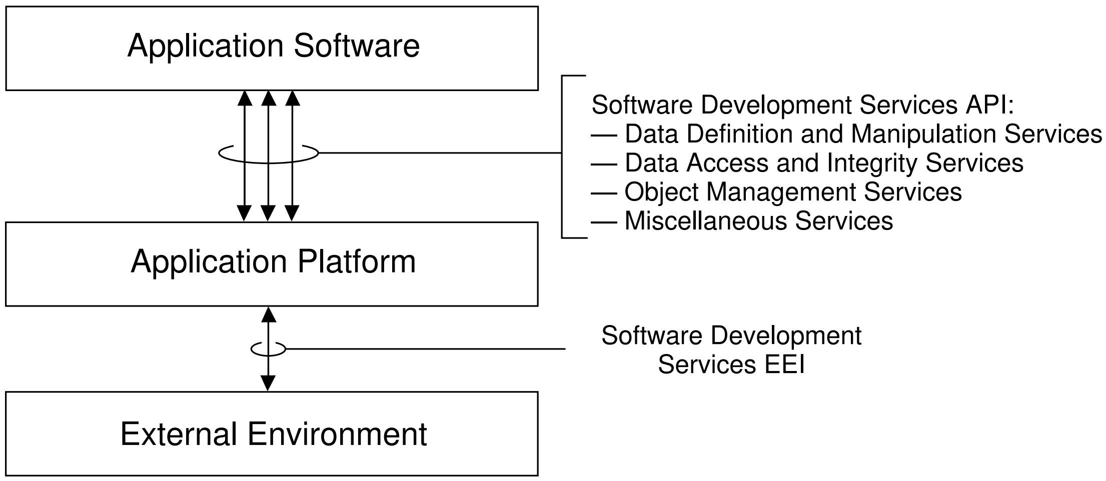

Figure 4-23 – Software Development Reference Model

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.