PDF- XML EDITOR

PROJECT WORK DONE AT

WIPRO TECHNOLOGIES, BANGALORE.

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

OF BHARATHIAR UNIVERSITY,
COIMBATORE.

SUBMITTED BY: **VASANTH RAM . R**

Reg.No.: **9838MO524**

GUIDED BY

EXTERNAL GUIDE

Rajesh Guptha,
Project Leader,
Wipro Technologies,
Bangalore

INTERNAL GUIDE

Mrs.D.Chandrakala. M.E.,
Senior Lecturer,
Dept of CSE,
Kumaraguru College of Technology,
Coimbatore.

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore – 641 006
MAY 2001

CERTIFICATE

This is to certify that the project work entitled

PDF-XML EDITOR

Submitted to the

Department of Computer Science and Engineering

Kumaraguru College of Technology

in partial fulfillment of the requirements for the award of the degree of Master of Computer applications is a record of original work done by **Mr. R. Vasanthram. Reg. No. 9838MO524** during his period of study in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore under my supervision and this project work has not formed the basis of award of any Degree/Diploma Associateship/Fellowship or similar title to any candidate of any University.

Professor and Head 26/4/01

Staff-in-charge 26/4/2001

Submitted to University Examination held on 11-05-2001

Internal Examiner 11/5/2001

External Examiner 11/5/01

WIPRO
Applying Thought

CERTIFICATE

16th April 2001,
Bangalore.

This is to certify that **Mr. R. Vasanthram** undertook the project entitled **"PDF – XML Editor"** in our organisation from December 2000 to April 2001.

The part of this project he has completed embodies the original work done of the said Module in the partial fulfillment of the requirement for the Degree fo **Master of Computer Applications.**

Mr. R. Vasanthram has worked on this project with dedication towards its sucessful completion. His performance has been very good.

For Wipro Technologies,

Rkpathak

Mr. Rajeev Kumar Pathak,
Project Manager,
Wipro Technologies.

Wipro Technologies
Innovative Solutions. Quality Leadership.
Sri Chamundi Complex, 26, Hosur Main Road, Bommanahalli, Bangalore-560 068. India. Tel : 91-80-5722296 Fax: 91-80-5722696 www.wipro.com
Regd. Office : Wipro Limited

DECLARATION

I hereby declare that the project entitled **'PDF-XML Editor '**, submitted to **Bharathiar University** as the project work of **Master of Computer Applications Degree**, is a record of original work done by me under the supervision and guidance of **Mr. Rajesh Guptha, Project Leader,Wipro Technologies, Bangalore**, and **Mrs. D. Chandrakala, Senior Lecturer, Kumaraguru College of Technology, Coimbatore** and this project work has not found the basis for the award of any Degree/Diploma/ Associateship/Fellowship or similar title to any candidate of any university.

Name of the Candidate	Registration Number	Signature of the Candidate
R. Vasanthram .	**9838M0524**	R. Vasanthram.

Countersigned by:

Staff-in-Charge

Mrs. D. Chandrakala M.E.,

Senior Lecturer,

Dept. of Computer Science & Engineering,

Kumaraguru College of Technology, Cbe

Place: Coimbatore

Date: 26-4-2001

ACKNOWLEDGEMENT

First I woldlike to thank express my sincere thanks to our Head of the Department **Dr. S. Thangaswamy Ph.D.,** who made it possible for me to do my project outside the campus.

My special thanks to my Project Manager **Mr.Rajeev Kumar Pathak ,** and to my Project Leader **Mr. Rajesh Guptha,**who helped me a lot in completing my project sucessfully.

I wish to express my deep sense of gratitude to my internal guide **Mrs. D. Chandrakala M.E.,** Lecturer, Department of computer Science and Engineering, Kumaraguru College of Technology. Her Comments, suggestions help and criticisms have been invaluable to me.

R. Vasanth Ram

SYNOPSIS

This project is intended to the development of an editor to edit the xml file called PDF.xml .This xml file contains model elements for a printer driver that is generated using Printer Driver Framework (PDF) .

The xml file exists in the hard disk .This editor uses a parser to read the xml file.The parser reads the xml file and hands the data off to the application,the application contains the user interface in which the data received from the parser is displayed .The editor allows the user to perform any operations like adding ,deleting or modifying the elements xml file . Since the elements in the xml file are categorised and displayed in group wise the manipulation of the xml file becomes much easier.

This modified PDF.xml file will be used for developing binary model data for the drivers that are developed using PDF. The users of this editor are the customization team developers .They can modify the PDF.xml file for generating a custom driver from PDF.

This editor is developed for editing the xml file PDF.xml. No other xml files can be edited using this editor

CONTENT

1.1 Project Overview

(PDF) Printer Driver Framework

This project is concerned with developing a group of source code and binary deliverables which are designed to be used in the production of printer drivers. The primary goal of a framework is to make the adaptation and reuse of core functionality from product to product go smoothly. These future releases of printer drivers will reuse the PDF technology core, and implement custom features specific to the product targeted as additional software to be integrated with PDF to produce the printer driver of the PDF work..The model information of the printer drivers that are generated using PDF is contained in the file called PDF.xml. PDF makes use of this file to generate a printer driver.

PDF – XML Editor

The objective is to develop an editor to edit the XML file PDF.xml. This PDF.xml file contains the model information for a printer driver generated using Printer Driver Framework (PDF).

User can perform different operations such as :-

- Add,delete or modify an element .
- Modify the value of the attributes of an element.

This modified PDF.xml file will be used for developing binary model data for the PDF. The users of this editor will be the customisation team developers .They can modify the PDF.xml file for generating a custom driver

from PDF using this editor This editor can't be used to edit any xml files other than PDF.xml

1.2 Organisation Profile

Wipro is a leading IT services company providing solutions across the globe to Fortune 500 companies. They provide e-business transformation solutions to their enterprise customers as well as product realization services to our internet infrastructure customers.They have very strong software engineering processes and were the first software services company to be rated at SEI CMM Level-5. They have global operations across North America, Europe and Japan and have over 200 satisfied customers. They provide a full range of services including architecting,Integrating and Managing solutions. These solutions are provided across a wide spectrum of technologies ranging from ecommerce to wireless. Across specific industry groups which they focus on.They provide value to their customers by offering several benefits to them including speed to market, high quality people and processes, cutting edge technology expertise and a full service portfolio . Wipro has a solid track record over the last decade with over 250 satisfied customers across US, Canada, Europe, Japan, and Asia Pacific.

Wipro is:

- Among the top 10 global technology service providers by 2004 offering the complete spectrum of e-business, Internet and communication technology services and components

- Delivering time-to-deploy and quality advantages to our customers with the world's best processes and people

- In an environment of empowerment, intellectual challenge and wealth sharing

Wipro's Quality Management System has a 3 Tier hierarchy structure as follows

I Tier (Policies)

The policies are derived from the SEI CMM and the ISO 9001 standards as applicable to the software development lifecycle within the scope of services of Wipro Technologies .

II Tier (Procedures & Forms)

Procedures and Forms are derived and developed to meet the requirements and objectives of the Policies stated in the Quality Manual. Policy objectives are met and satisfied by implementing our tasks as per the documented procedures

III Tier (Work Instructions)

Guidelines are used to tailor the actual practices as documented in QMS to meet the specific requirements of different projects

[redacted]

2.1 Existing system – Limitations

There are lot of xml editors available on the market either free of cost or for a cost that is affordable by any concern.These editors normally used to edit all xml files. These xml editors however when used to edit a big xml file like PDF.xml fail to satisfy certain requirements of their users.For example in these kind of xml editors all the elements that are present in the xml file are displayed

in alphabetic order or in some other order that is not convenient for the user to search for an element. So when the user wants to add or delete or modify an element in the PDF.xml file he has to search through the list of all elements to find the element. This task is indeed cumbersome and time consuming.

Limitations:

The limitations of the existing system is that those editors that are avilable in the market fail to satisfy certain requirements of the PDF.xml users incertain occasions and since those are developed by a third party the users have to get the license from them to use it and also so they have to keep on updating the existing version of the xml editor with the newer one since the license agreement is made only for a specific period.

2.2 Proposed System

The PDF XML editor that we have developed satisfies the requirements of the PDF.xml file users.It facilitates editing the PDF.xml file . The elements in the PDF.xml file are categorised according to their nature and the user interface part of the editor contains a tab for each category.When the user wants to perform any operation on the any of the element that is present in the file he just has to click on the tab that the element belongs to . The editor provides an options for all sorts of operations that the users of the file want to do. This editor can also display the DTD that is associated with the PDF.xml file.This editor is purely meant for editing the PDF.xml file and no xml file other than this can be edited using this editor.

Features of this editor

- This editor will acts as a data interpreter

- The editor will read the XML file and displays it to the user and againit writes the changes back in to the same XML file.
- Application will provide a GUI through which user can perform different operations.
- An interface is developed between the application and parser.

2.3 Requirements on new system

Purpose

In order to overcome the above said difficulties in using the third partyeditors for editing the PDF.xml file we develop a PDF XML editor. The principal requirements consist of facilitating the editing of the PDF.xml file.

Objectives

The objective of this project is to develop an xml editor that Is customised to the requirememts of the PDF.xml file users.

Scope of the project

The main idea of this xml editor is to facilitate the editing of thePDF.xml file . This editor can only be used to edit the PDF.xml file and no other xml file can be edited usign this editor.

Product perspective

The main idea behind the development of this editor is to make the handling of PDF.xml very easier. The customisation team developers who are the only users of this editor can modify the PDF.xml file using this editor to for developing a custom driver from PDF.

2.4 User Characteristics

This is a product . This is used for editing only the PDF.xml file .Thus

the scope of this editor is confined with in the PDF group which makes use of the PDF.xml file. The PDF group makes use of the PDF.xml file for generating a custom driver from the framwork. The users of this editor are the customisation team developers who will be wanting to modify this xml file for generating the customised driver.

3.1 Hardware configuration

Pentium III Processor

64 MB RAM

4 GB Hard Disk

VGA graphic adapter with color monitor

3.2 Software Configuration

Microsoft Windows 98

Microsoft Visual Studio(VC++)

e**X**tensible **M**arkup **L**anguage (XML)

DOM Parser

3.2.1 About Windows

Visual Interface

The Windows operating system offers a 'Visual Interface', which is intuitive and easy to use. With its conventional Graphical user interface windows lures the users and programmers like the proverbial honeybee

Windows is Microsoft's GUI that provides a large, complex and flexible environment for users. The users appreciate the ability of windows OS to execute several programs simultaneously, switching effortlessly by pointing at windows and clicking them with the mouse.

Multitasking

Windows lift many obstacles to the full usage of the PC. It brings the PC user interface close to a Macintosh but nevertheless retains the advantage of the PC massive application base and supercedes Macs with powerful multitasking support features. Under Windows NT, a type of multitasking called 'Sever/Dll' was introduced wherein the programs themselves can split into multiple threads of execution that seem to run concurrently.

Consistent User Interface

The Windows environment offers a consistent user interface and hence helps users to learn how to interact with the program. As the size of the program started increasing, it was realized that most of the programs had some common features, which were getting repeated unnecessarily in every program. Windows eliminates this problem by making each of these common features a part of the Windows OS responsibility.

Types of Classes

Windows provides for three types of window classes:

- System global classes
- Application global classes
- Application local classes
- System Global Classes

Implementation

The current Win32 platforms (Windows NT and Windows 95) keep the system classes separate for each 32-bit process. How the system classes are implemented does not affect application.

Windows implementation

Windows does things differently. There are two parts of the Win32 subsystem in Windows . a server process and a dynamic-link library (DLL) that runs inside each Win32 process. Windows registers the Edit class from the DLL inside each process. This allows all the code that handles edit controls to reside in that DLL. No local procedure calls are needed to handle edit controls, and the system overhead caused by applications' heavy use of these controls is avoided. Because edit controls manipulate only their own data, the impact on system robustness is minimal.

The server process keeps information for each Win32 application, including a list of the application's public and private classes, on the system. When a queue is created for a Win32 thread, which occurs when the thread calls a USER or graphics device interface (GDI) function, USER checks to see if this is the first thread for the process. If it is, USER registers the rest of the system classes. When a class is registered for any process except for the server process itself (USER keeps track of when it is registering classes for itself), the class is added to either the public list or the private list for the process. Effectively, Windows registers the system classes for each process and stores a copy of each system class for each Win32 application.

In Windows 16-bit applications share the same process, so they share all the system global classes, subclasses and all.

Event-Driven Programming

Windows provide an event-driven programming that is similar to the UNIX X-WINDOW system and the Apple Macintosh system for the user and developer. It also provides a common interface across applications that follow the style guides and convention created by Windows. In event-driven paradigm, instead of creating an application waits for the system to call upon it with a message.

All hardware is accessed through Windows. So just reading the keyboard or updating the display requires calls to MS windows system; the developer can no longer access the hardware directly. The application should also be a "good citizen" in the environment and cooperate with the other application by giving up the CPU after completing the response to the event that triggered the instance of code execution.

The Windows system receives all the events. These events include a keystroke, mouse movements, or a system timer event. The Windows system converts each event into message those represents the type of the event has occurred. The message also carries with it the type of event.

3.2.2 Microsoft Visual Studio C++ 6.0

Visual C++ is the best tool for the creation of user interfaces for the system side programming. The developer Studio is very user-friendly. Visual C++ compiler supports 32 bit programming. Since the OS is also 32 bit, the programming can be done using this software. Visual C++ is the software through which the programming can be done close to the kernel of the operating system. The programming can be done either using SDK or MFC.

The advantage of using SDK is that the memory usage can be well controlled. MFC is a work in progress. It's nigh on impossible for the

ordinary person to determine all its features and pitfalls using the documentation that Microsoft provides, let alone to follow its evolution; it changes quarterly. That's why MFC developers learn MFC by trial and error.

3.2.3. eXtesible Markup Language

A markup language does not worry about how the content it describes is formatted but is, instead, concerned with accurately describing its content. In short, XML is a markup language that can run on any platform, operating system, or environment and is designed to provide developers with a mechanism to better describe their content. In a nutshell, XML provides both a more extensive means for describing document content and a mechanism for describing metadata, using a method that will work on all computers, regardless of platform or operating system. XML takes Web data to the next level, using vocabularies defined by DTDs tailored to specific kinds of content. But unlike HTML, XML tags tell you what the data means, rather than how to display it.

The Standard Generalized Markup Language (*SGML*), from which XML is derived, was born out of the basic need to make data storage independent of any one software package or software vendor. SGML is a powerful language for describing the content and structure of electronic information. Basically, SGML is just too expensive and complicated for Web use on a large scale.

The document type definition (DTD) is an SGML or XML document that describes the elements and attributes allowed inside all the documents that can be said to conform to that DTD. However a DTD is optional for an XML

Features of XML

- XML can be used with existing Web protocols (such as HTTP and MIME) and mechanisms (such as URLs), and it does not impose any additional requirements.
- XML supports a wide variety of applications. HTML is simply too specific.
- XML is compatible with SGML.
- It is easy to write programs that process XML documents.
- XML documents are reasonably clear to the layperson. - Given the time, you can print out any XML document and work out its meaning—but it goes further than this. A valid XML document:
  - Describes the structural rules that the markup attempts to follow.
  - Lists any external resources (external entities) that are part of the document
  - Declares any internal resources (internal entities) that are used within the document
  - Lists the types of non-XML resources (notations) used and identifies any helper applications that might be needed.
  - Lists any non-XML resources (binaries) that are used within the document and identifies any helper applications that might be needed.
- The design of XML is formal and concise. The Extended Backus-Naur Format (EBNF) was used as the basis of the XML specification (a method well understood by the majority of programmers).
- XML documents are easy to create.

Benefits of using XML

XML is structured

By allowing you to define your own tags and create the proper structural relationships in your information (with a DTD), you can use any XML parser to check the validity and integrity of the data stored in your XML documents. This makes it very easy to validate the structure and content of your information when you use XML.

Here are some benefits of the structured nature of XML:

- XML parsers make your application code more reliable and quick to develop by providing validity checking on your XML documents (if you use a DTD).
- XML allows you to easily generate XML documents (that contain your information), since it is so structured.
- XML parsers allow you to code faster by giving you a parser for your all your XML documents (with and without DTDs).

XML documents are easily committed to a persistence layer

XML documents may be stored in files or databases. When stored in files, XML documents are simply plain text files with tags (and possibly DTDs). It is very easy to save your XML documents to a text file and pass the text file around to other machines, platforms and programs (as long as they can understand the data).

XML is platform independent, textual information

Information in an XML document is stored in plain-text. This might seem like a restriction if were thinking of embedding binary information in an XML document. There are several advantages to keeping things plain text. First, it is easy to write parsers and all other XML enabling technology on different

platforms. Second, it makes everything very interoperable by staying with the lowest common denominator approach.

XML is an open standard

By making the W3C the keeper of the XML standard, it ensures that no one vendor should be able to cause interoperability problems to occur between systems that use the open standard.

XML is language independent

By being language independent, XML bypasses the requirement to have a standard binary encoding or storage format. Language independence also fosters immense interoperability amongst heterogeneous systems. It is also good for future compatilbilty.

XML is web enabled

XML is derived from SGML, and so was HTML. So in essence, the current infrastructure available today to deal with HTML content can be re-used to work with XML. This is a very big advantage towards delivering XML content using the software and networking infrastructure already in place today. This should be a big plus in considering XML for use in any of your projects, because XML naturally lends itself to being used over the web.

XML is totally extensible

By not predefining any tags in the XML Recommendation, the W3C allowed developers full control over customizing their data as they see fit. This makes XML very attractive to encoding data that already exists in legacy databases (by using database metadata, and other schema information).

XML supports shareable structure (using DTDs)

Since the structure of the XML document can be specified in DTDs they provide a simple way to make it easier to exchange XML documents that conform to a DTD. DTDs are a simple way to make sure that 2 or more XML documents are of the same "type".

XML enables interoperability

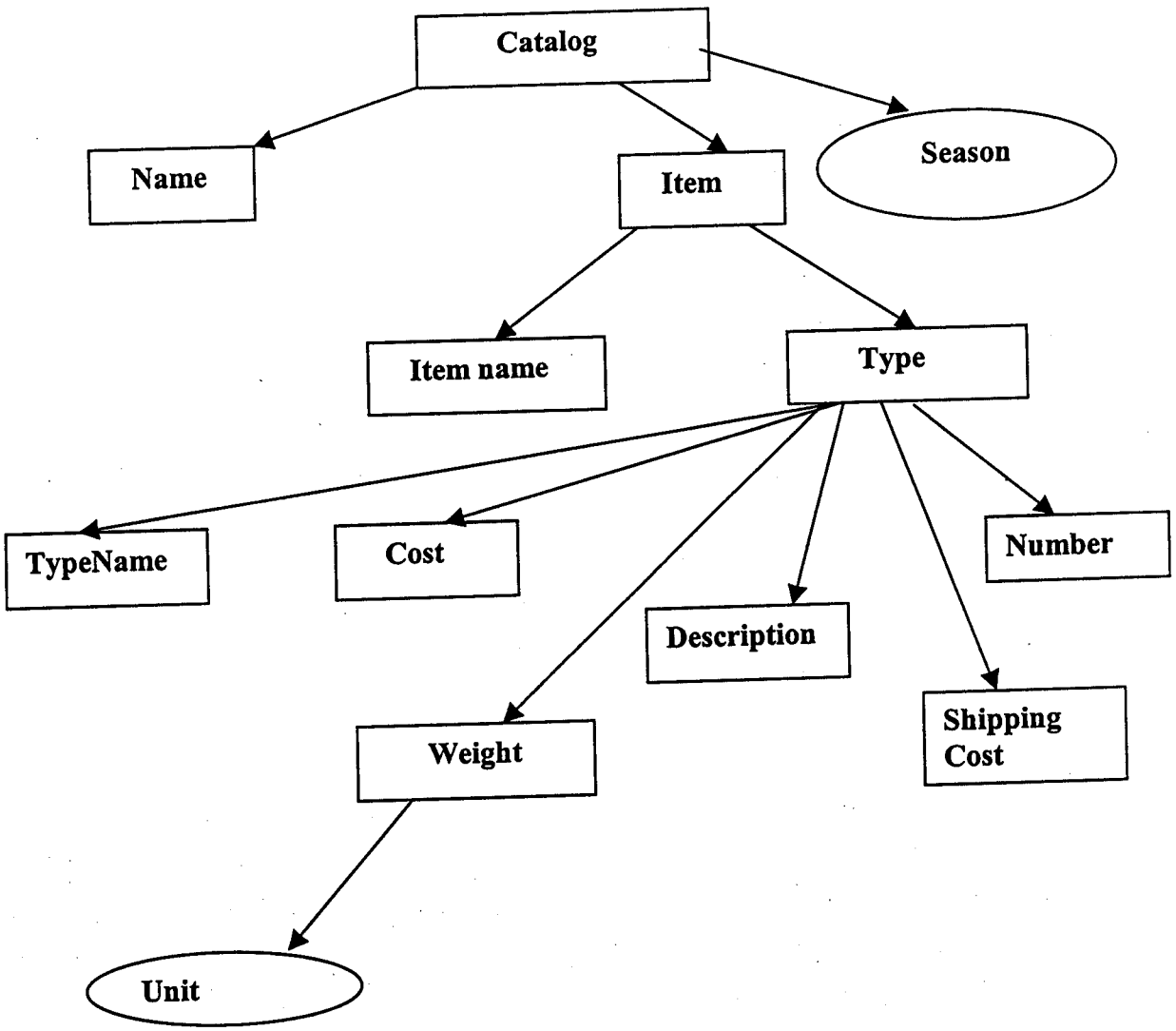
All of the advantages of XML outlined so far all make interoperability possible. This is one of the most important requirements for XML, to enable disparate systems to be able to share information easily.

3.2.4 Parsers

XML documents are processed by applications called ***parsers***. The parser reads the document and generates output based on the document's content and the markup used to describe that content. In some instances, the document must be compared to and abide by the rules specified in its DTD. When properly constructed, these documents are called *valid*. Parsers that have the ability to compare a document to its DTD and determine whether the document is valid are called *validating parsers*. Even if a document does not have to be validated, it must still conform to the general rules of document creation established in the XML specification. Documents that obey all the general rules are considered well formed. All parsers check to make sure an XML document is well formed, but only validating parsers also check to see if a document is valid. While all documents must be well formed, not all must be valid.

Specialized browsers are currently being developed that know how to interpret the output from a parser and render a display of the XML document using that output. XML is a new technology, so parsers and browsers are just beginning to appear. A parser can process any XML document and a validating parser can

The DOM Tree



In the above diagram the elements that are enclosed in a ellipse represents an attribute and others represents the elements of the xml file.This tree is generated by the xml parser as it reads the xml file .

This parsing example accomplishes the following activities:

- Create the DOM tree for the XML document
- Print the complete tree including attributes
- Find and replace the specified element value passed as parameters
- Print the complete tree.

DOM features :

Benefits

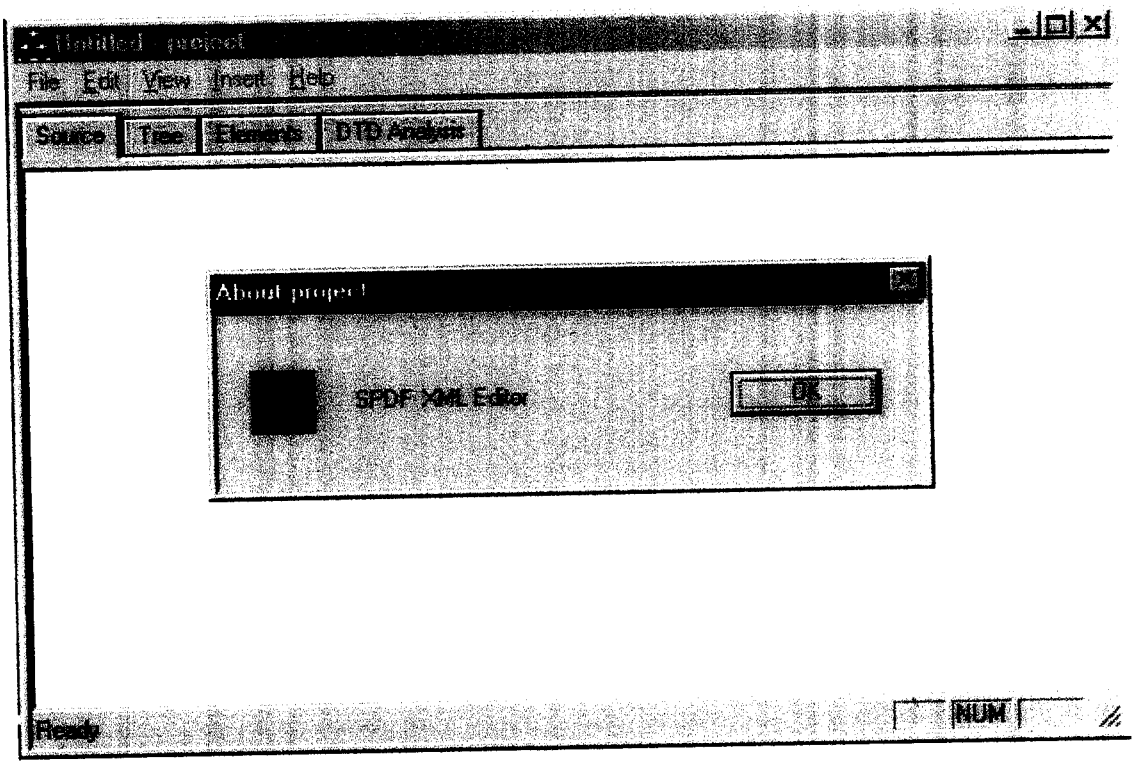
- The DOM ensures proper grammar and well-formedness.
- The DOM abstracts content away from grammar.
- The DOM simplifies internal document manipulation.
- The DOM closely mirrors typical hierarchical and relational database structures.

Drawbacks

- If the file is too big then it is very difficult to handle .

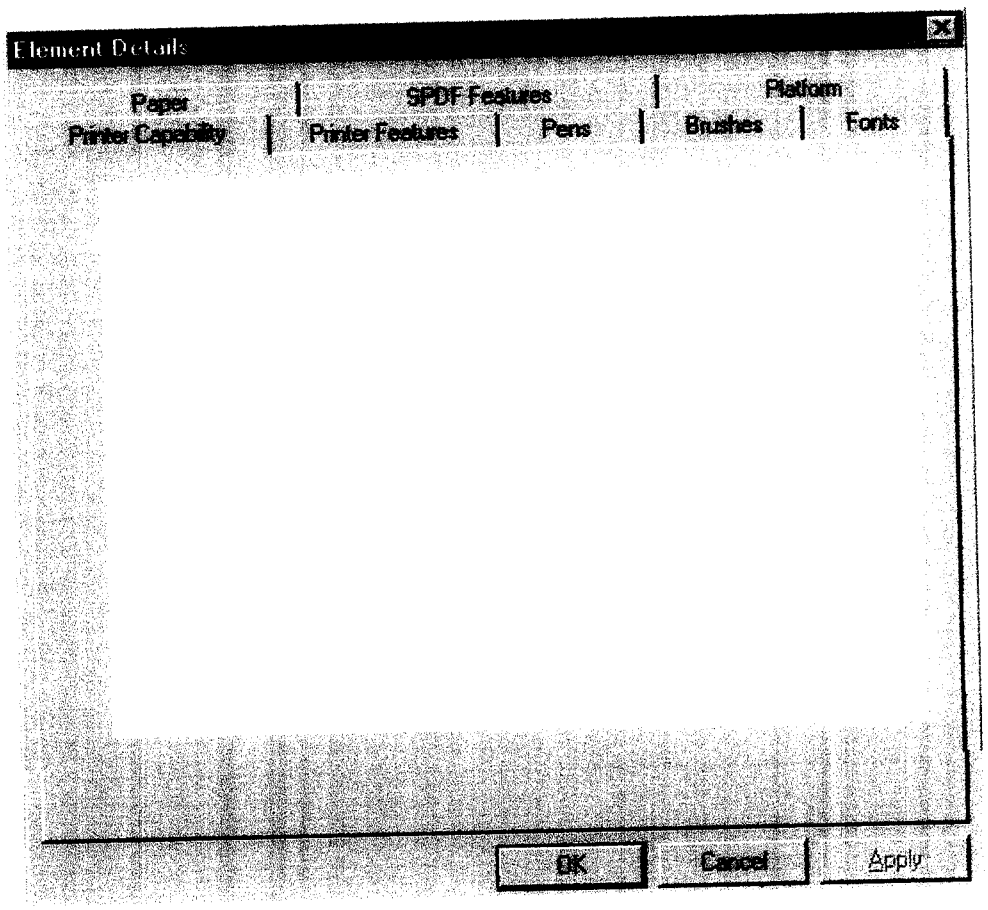
The User will be presented with the following layout whenever he starts the XML Editor.

Input Screen:



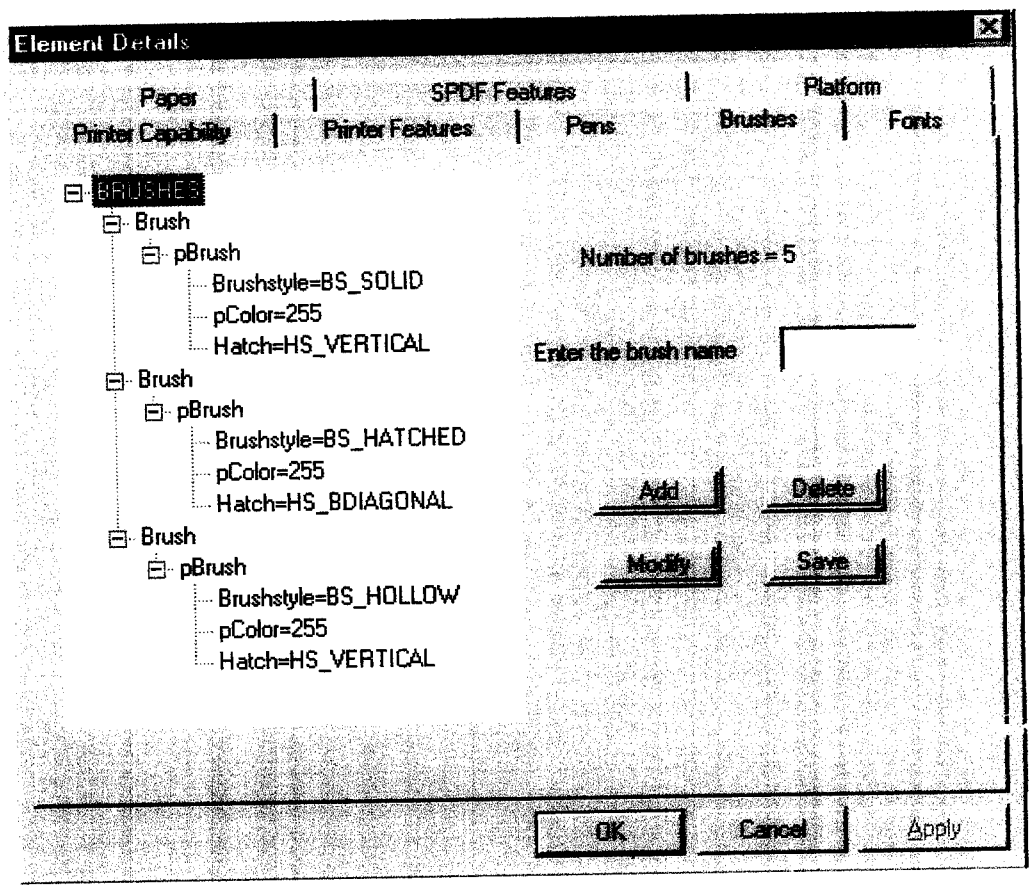
Here whenever the source tab is clicked the whole xml file will be shown to The user. When ever tree tab is clicked the xml file will be shown in a hierarchical Structure. When the user clicks the element button the names of all category of elements that are present in the xml file will be displayed as follows

Element details



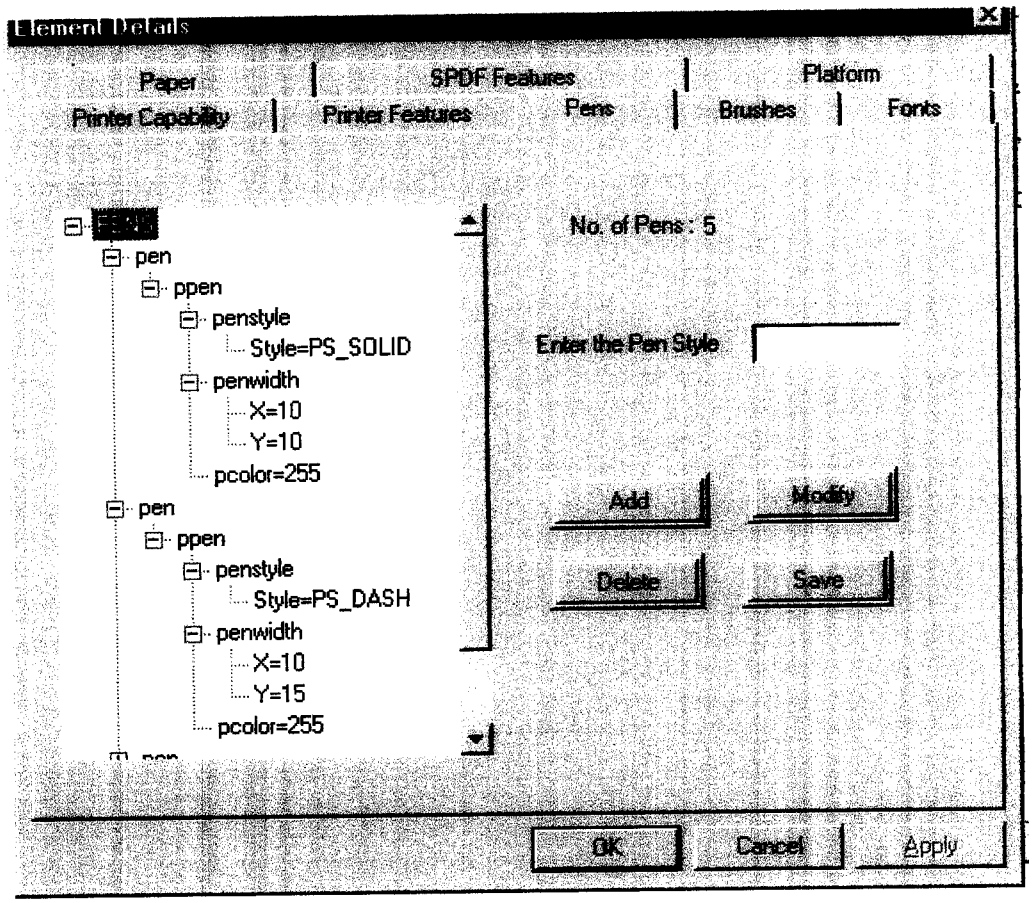
If the user wants to modify any element then he has to click the tab that the element belongs to.Then the elements will be shown in an hierarchical structure.The user Can add or delete elements and he can change the attributes of the elements .After Performing the operation he has to click the appropriate button.

Brush



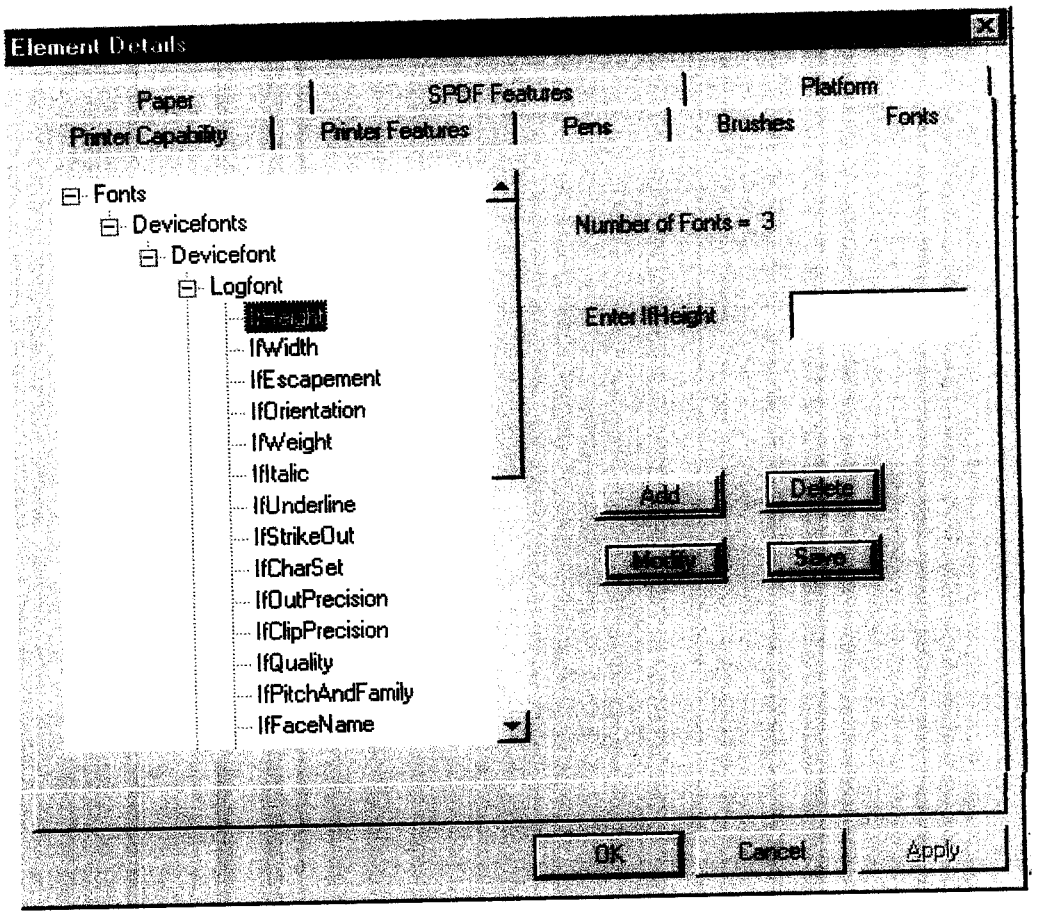
The Brush element and all of it's attributes are shown in a tree structure whenever the user clicks the brush tab.He can add or delete the elements in this new screen.

Pen



The user can traverse in the tree to perform whatever the modification he wants to perform.The user is also prompted to enter the value for the element that he has selected.

Font

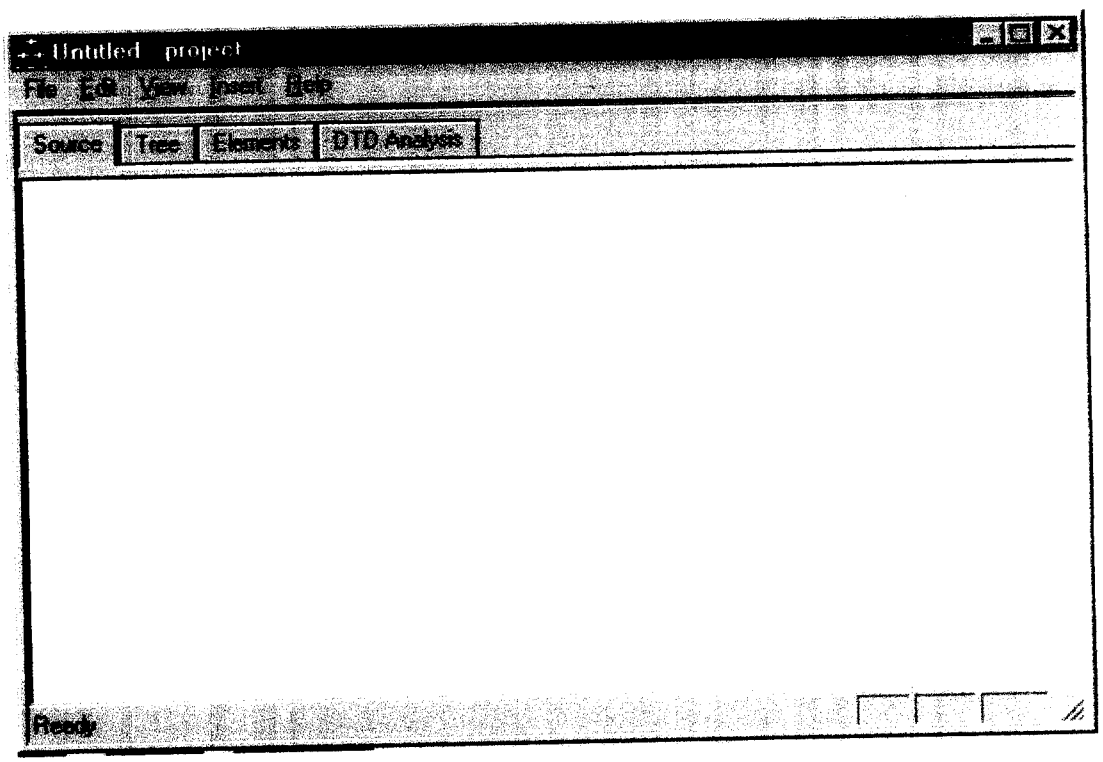


This Layout shows different types of fonts that are present in the xml file and their attributes.The user can add,delete,modify any font , their attributed by traversing through this tree.

4.2 Output Design

The output is displayed in the same screen that is displayed to the user for entring the data.The modifed elements and the new xml are displayed in the following screen.

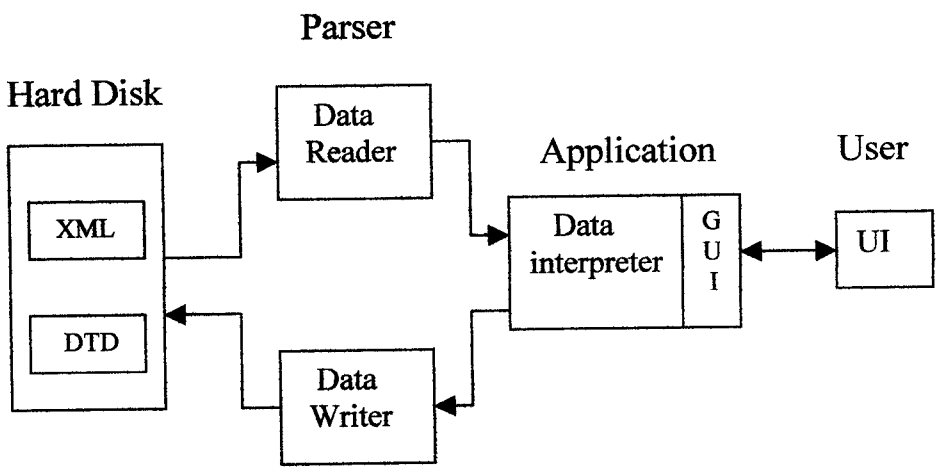
Diagram:



The modified xml file will be displayed when the user click the source tab.The modified DTD and tree structure are also displayed in this layout when their Respective tabs are clicked.

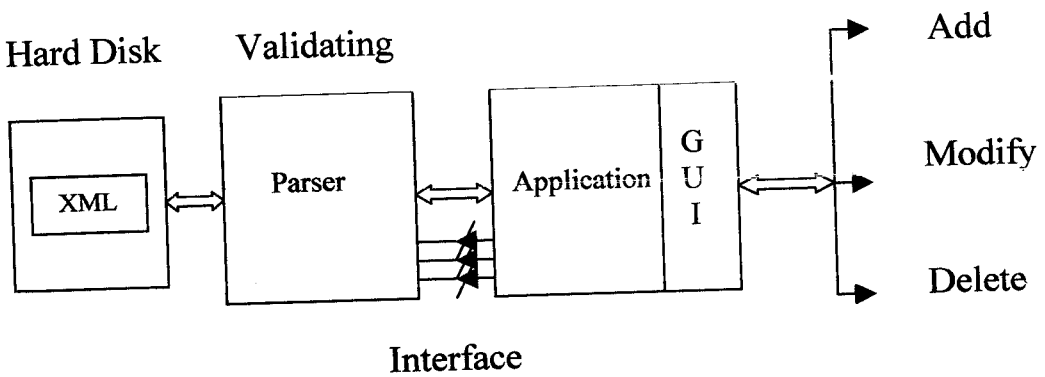
4.3 Process Design

The conceptual design of this project is shown in the following diagram.



The xml file exists in the Hard disk. A parser is used to read the xml file from he hard disk.The parser reads the xml file and generates a hierarchical structure and hands it off to our application.Our application acts as an Data interpreter the GUI part of the application displays the xml file that is received the data interpreter and also allows the user to edit the xml file after all modifications are done the changes are written back to xml file using parser.

The fuctional design of this project is shown in the following diagram.



The application uses the API's provided by the Parser to interact with the xml file that exists in the hard disk. Then the elements of the xml file are displayed to the user through the GUI part of the application. Once the modifications are done the elements are written back to the disk using the parser.In the GUI part provides the users options for performing various operations on the xml file.Few of the interfaces that are provided by the DOM parser and are called by our application are IXMLDOMDocument,IXMLDOMNode,IXMLDOMNodelist.

These interfaces provide lot of functions that are used by the application to interact with the tree that is generated by the parser.

5.1 System Implementation

In the implementation part all the list of methods and the property of the interfaces are mentioned. Each method Name, Description of the method, parameter for the method and the return type are listed in this section.

DOM Interfaces

The DOM level 1 core defines a basic set of interfaces that allow the manipulation of XML documents. It provides methods for the access and population of the document. These methods are encapsulated in two sets of interfaces: the fundamental core interfaces and the extended interfaces.Here is a basic presentation of the main interfaces. For a complete description and all the methods, you will need to download a DOM library. Again, xml4c2 is a good choice because of its excellent documentation. These interfaces can be called by the using the API's of any applications .Several applications provide API's that can be used to call the interfaces of DOM.For example Microsoft VC++ comes with lot of MFC classes that are specially built for calling the interfaces of the DOM.

Fundamental Interfaces

Interface	Description
Node	This interface is the primary datatype for the entire Document Object Model. It represents a single node in the document tree. This is the base interface for everything in the model—therefore all objects implementing the Node interface expose the methods defined by it. One should be careful about this because some derivatives of node, like the text node, expose some Node methods they don't really support like "get children," which results in an exception since a text node cannot have children.
Document	Class to refer to XML Document nodes in the DOM. Conceptually, a DOM document node is the root of the document tree, and provides the primary access to the document's data.
Document Fragment	DocumentFragment is a "lightweight" Document object. This object encapsulates a portion of the document, which is very useful in applications that need to rearrange or modify portions of the tree, for example an editor doing a cut/paste. Note that the fragment contained is not (necessarily or even often) a valid XML document.
Element	The majority of objects, apart from text, that one may find in the

	DOM Tree are DOM Element nodes. They represent elements in the document object model, and since they can have other Element nodes as children, their structure reflects the arrangement of the original XML document.
Other interfaces	The rest of the fundamental interfaces are: DOMImplementation, NodeList, NamedNodeMap, CharacterData, DOMException (which in IDL is not an interface but an exception), Attr, Text, Comment. Again, for more details on these, you are encouraged to download the complete documentation included in toolkits like xml4c2

Extended Interfaces

The extended interfaces also form part of the core DOM. These interfaces are:

- CDATASection
- DocumentType
- EntityReference
- ProcessingInstruction

Other XML DOM Interfaces

IXMLDOMDocument

Represents the top node of the XML DOM Tree.

IXMLDOMDocument:: createAttribute (name)

Creates a new attribute with the specified name.

IXMLDOMDocument:: createElement (name)

Creates a new element with the specified name.

IXMLDOMDocument:: createNode(Type, name, namespaceURI)

Creates a node using the supplied type, name, and namespace.

IXMLDOMDocument:: createCDATASection(data)

Creates a CDATA section node that contains the supplied data

IXMLDOMNode

Represents a single node in the DOM tree.

IXMLDOMNode :: appendChild(newChild)

Appends a new child to a list of nodes

IXMLDOMNode :: firstChild()

Contains the first child fo the node.

IXMLDOMNode :: firstChild()

Returns the last child fo the node.

IXMLDOMNode :: insertBefore(newChild,refChild)

Inserts a child node to the left of the specified node.

IXMLDOMNodeList

Supports iteration and indexed access operations on the live IXMLDOMNode collection of objects.

IXMLDOMNodeList :: Item(index)

Allows random access to individual objects in the collection.

IXMLDOMNodeList :: length()

Indicates the number of items in the collection

IXMLDOMNodeList :: nextNode()

XMLDOMNodeList

Supports iteration and indexed access operations on the live collection of XMLDOMNode objects

5.2 System Testing

Software testing is a critical element of software assurance and follows the ultimate review of specification, design and coding. In addition, data collected during testing is used to provide a good indication of software reliability and some indication of software quality as a whole. Some severe errors that require design modification are encountered with regularity, software functions appear to be working properly and the encountered are easily corrected.

The objectives of testing are as follows:

- Testing is the process of executing a program with the indent of finding an error.
- A good test case is one that has a high probability of finding as as-yet undiscovered error.
- A successful test is one that uncovers an as-yet undiscovered error.

The testing phase involves the testing of developed system using various kinds of data. The following tests were conducted and desirable results were noted.

BLACK BOX TESTING

In Black Box testing, the following were tested.

- Incorrect or missing functions
- Interface errors.
- Errors in data structures or external database access.

- Performance errors.
- Initialization and termination errors.

WHITE BOX TESTING

In white Box testing, the following were tested successfully for this application.Checking whether al independent paths within each module have been exercised at least once.

- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their bounds.
- Exercise internal data structure to ensure their validity.

UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design. Important controls paths were tested to uncover errors within the boundary of the module. The relative complexity of tests and errors detected as a result is limited by the constraints. The local data structures were examined to ensure that data stored temporarily maintains its integrity during all steps of execution. This testing was carried out during programming stage itself. In this testing step each module is found to be working satisfactorily as regard to the expected output from the module.

INTEGRATION TESTING

Data can be lost across an interface: one module can have an adverse effort on another; sub functions, when combined, may not produce the desired major functions. Integration testing is a systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the structure. All the modules are combined and tested as a whole. Here

correction is difficult because the vast expenses of the entire program complicate isolation of causes. Thus in the integration testing step, all the errors uncovered are corrected for the next testing steps.

VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of testing begins – validation test begins validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists.

The function or performance characteristics confirm to specification and are accepted.A deviation from specification is uncovered and a deficiency list is created.Proposed system has been tested by using validation testing and found to be working satisfactorily.

Graphical user interfaces (GUIs) present interesting challenges for software engineers. Because of reusable components provided by VC++ development environment, the user interface becomes less time-consuming and more precise. The GUIs in this system were tested as follows.

- The window open properly based on related type commands.
- The window can be resized, moved and scrolled
- All data contained within the window can be properly
- Addressable with a mouse, function keys, directional arrows and keyboard.
- The window properly regenerated when it is overwritten and then recalled

All the functions that related to the window are available when needed.

All the functions that related to the window is operational

All the relevant scrollbars, dialog boxes, buttons and other controls are properly displayed in the window

Multiple windows are displayed, the name of the window is properly represented

The window is properly closed

The mouse operations are properly recognized throughout the interactive context

Alphanumeric data entry is properly echoed and input to the system

Invalid data is properly recognized

Data input messages are intelligible

5.3 Refinements based on feedback

The list of bugs/enhancements was reported by the Customisation and developers are Fixed/Improved.

- The Platform tab was not working properly.
- Add button in the printer features screen was disabled.

[redacted]

Thus the "PDF-XML Editor has been completed sucessfully . After this editor is tested it is delivered to customisation team developers who are the only users of this editor.This editor is used by the cutomisation team developers for editing the PDF.xml file.The performance of this editor is significantly good.

This product PDF-XML Editor is always a upgradable product. Whenever the structure of the PDF.xml file is modified the editor will have to be modified to include the new element in the category.If an element is added to the PDF.xml file that doesn't belong to any of the category that if present in the user interface of the editor the editor is refined to include the new element also.

References

- **Microsoft Visual C++**

  * Michael Young "Mastering Visual C++" 5th Edition 1999

  * "The Complete Reference Visual C++" Tata McGraw Hill 3rdEditon

  * Charles Petzold "Windows Programming" 3rd Edition 2000.

- **Web Sites :**

  * www.xmltutorials.cm

  * www.aplhanetworks.com

  * www. Ibm .com

Printer Driver Framework

Conceptual Architecture

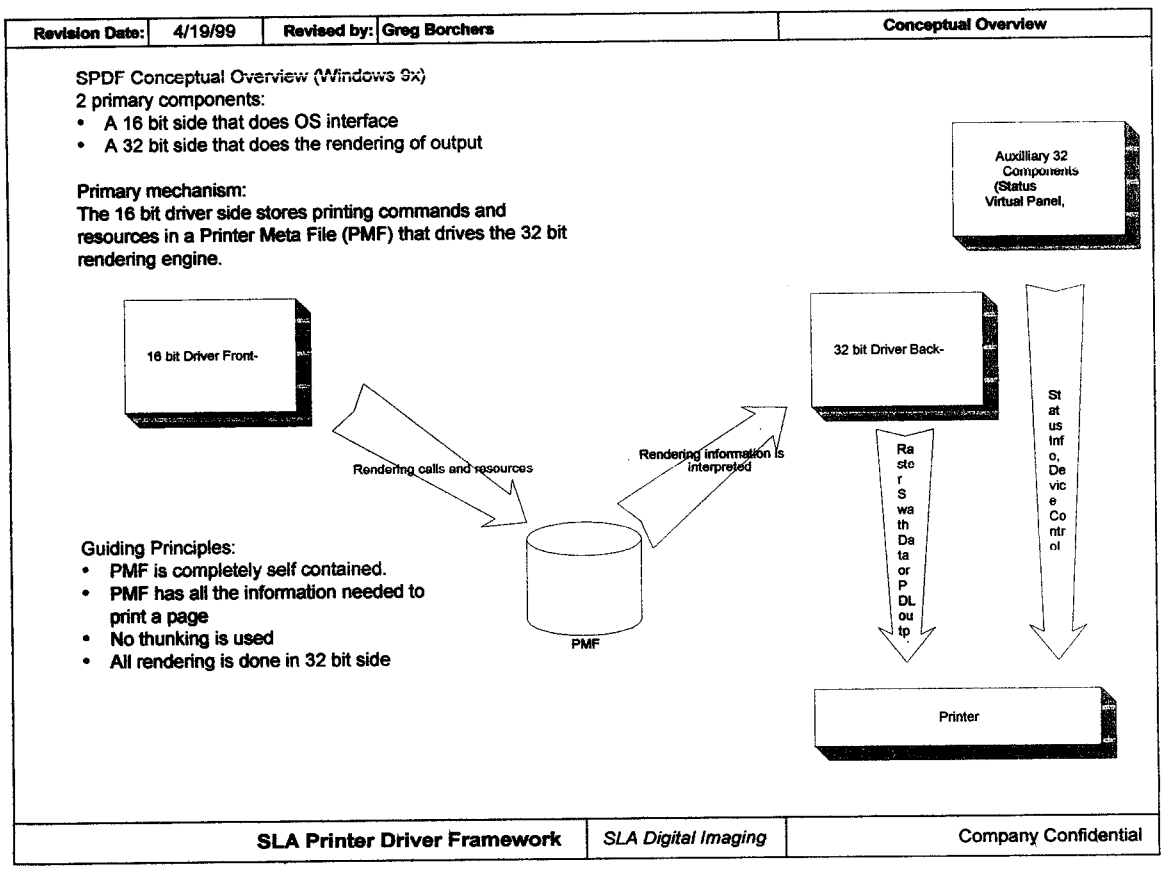


Figure 1 Conceptual Overview

The primary concept for this architecture is that the 16-bit driver interface is the front end for the driver, and the rendering is done in the 32-bit backend of the

Driver .PMF is the data format that transfers the rendering commands from the driver front-end to the back end renderer.

The driver module exports the standard Windows printer driver interface.The renderer has the standard Windows printer processor interface.

Defenitions and Acronyms:

Following is the list of terms that have been used and their meanings

XML – Extensible Markup Language.

PDF – Printer Driver Framework

DTD –Document Type Defenition .

DOM – Document Object Model.

GUI-Graphical User Interface

The User Interfaces in PDF –XML Editor

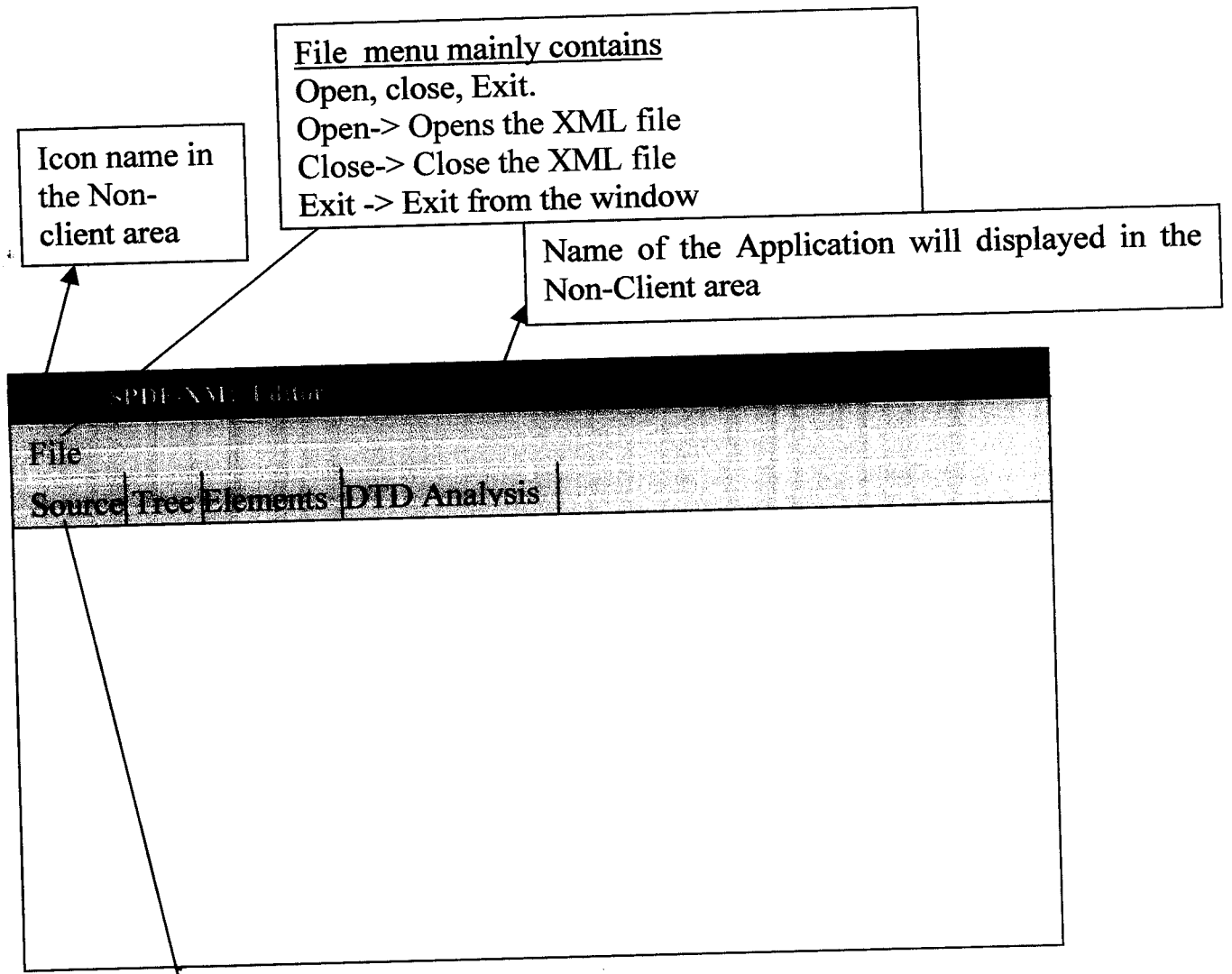


Fig 1.

When the XML file is chosen from File menu it will be displayed in the Source client area.

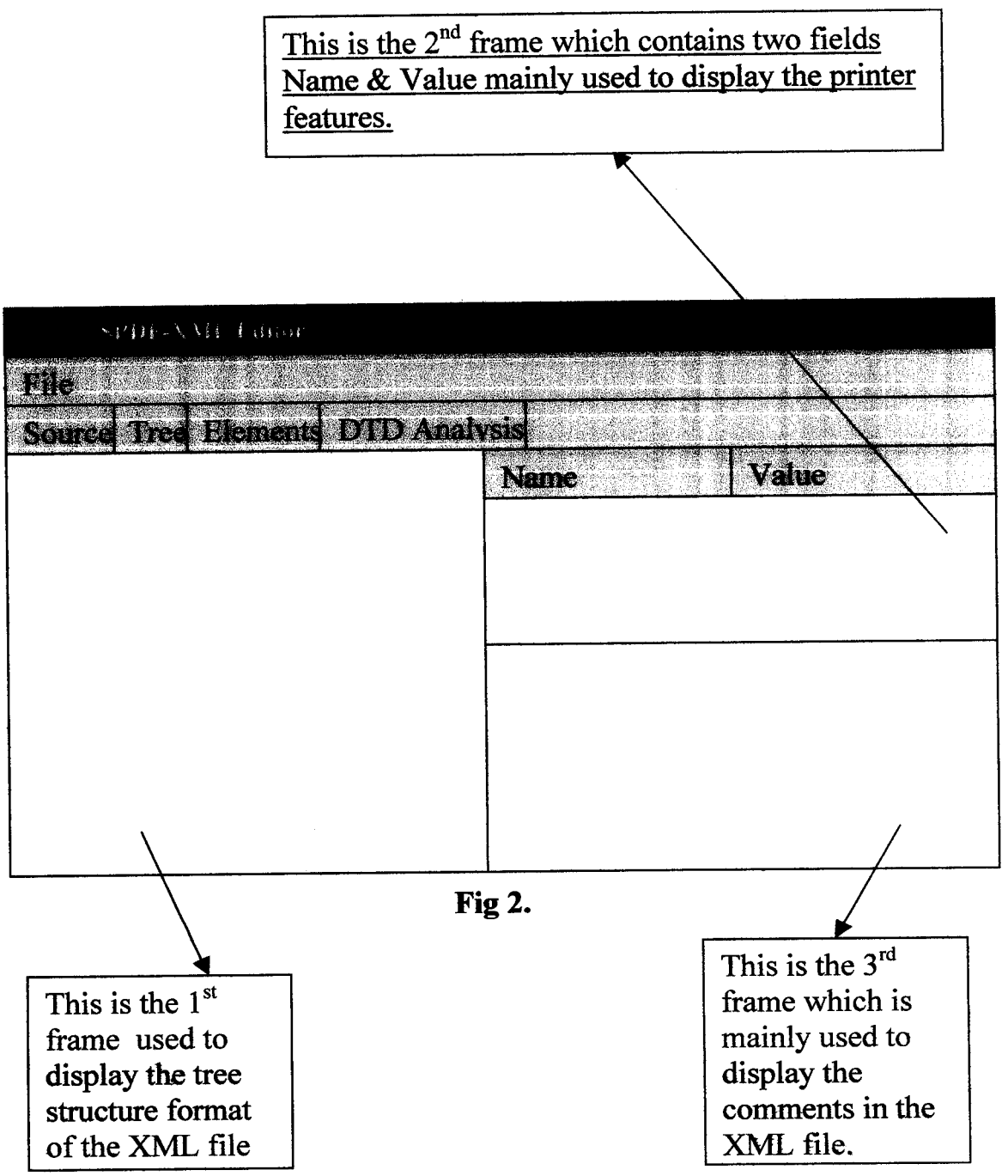

This is the 2nd frame which contains two fields Name & Value mainly used to display the printer features.
File
Source Tree
Elements
DTD Analysis
Name
Value
This is the 1st frame used to display the tree structure format of the XML file
This is the 3rd frame which is mainly used to display the comments in the XML file.

Fig 2.

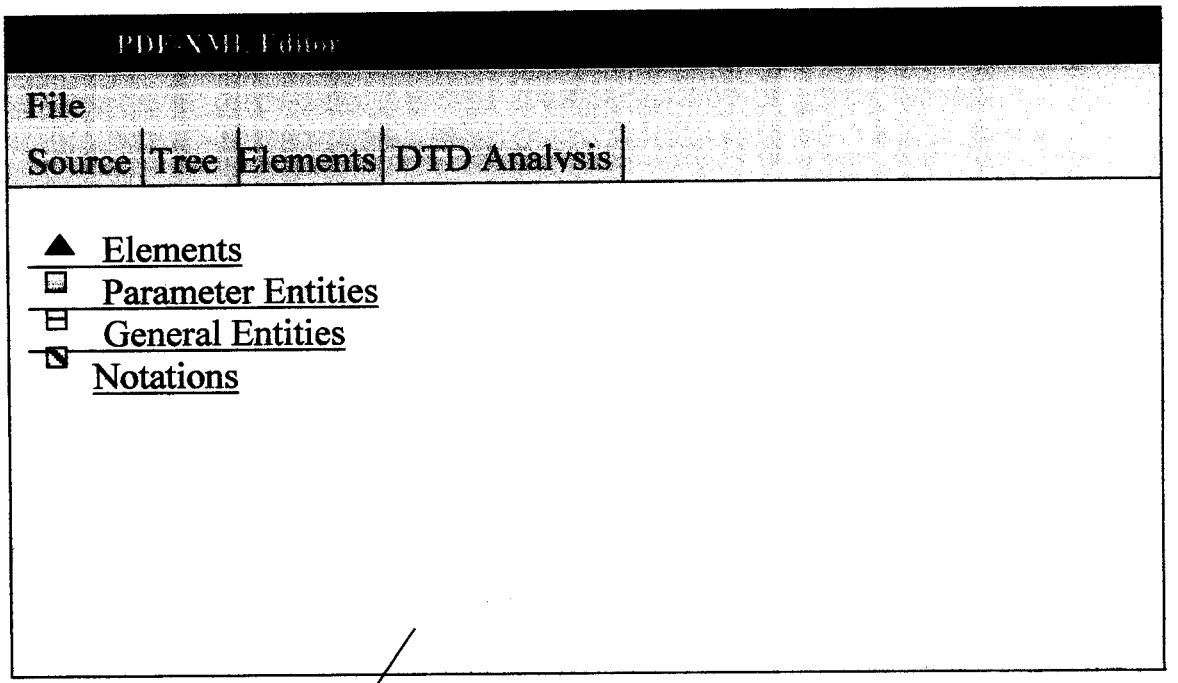


Fig 4.

When this tab is selected different feature like

- Elements in the DTD
- Parameter entities used in the DTD
- General Entities ued in the DTD
- Notations used in the DTD