XSLT Custom File Format Generator

Done at

Lucid Technologies Pvt. Ltd.

PROJECT REPORT

Submitted in partial fulfillment for award of Degree of
Master of Science in Software Engineering

Done by

Krishna Prathab R V

9837S0053

P- 935

Guided by

Mrs.S.Devaki B.E., M.S.,

Assistant Professor
Department of Computer Science & Engineering
Kumaraguru College of Technology
Coimbatore
&

Mr.B.Sriram Balasubramanian B.E., M.S.,

Project Leader
Lucid Technologies Pvt. Ltd., Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

COIMBATORE – 641006

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

COIMBATORE – 641006

P-935

BONAFIDE CERTIFICATE

This is to certify that this is the Bonafide Project Record Work done by **Krishna Prathab R V**, Reg.No **9837S0053** in Partial fulfillment for the award of Degree of M.Sc. [SOFTWARE ENGINEERING] , during the academic year 2002 –2003.

31/3/03

Prof & HOD

Guide

SUBMISSION

This Project entitled "XSLT Custom File Format Generator " is submitted for the X semester of M.Sc. [SOFTWARE ENGINEERING] for Bharathiar University Project Viva-voce examinations held on ....

Internal Examiner

External Examiner

Microsoft

Lucid Technologies Pvt. Ltd.

Bonafide Certificate

Certified that this thesis on "XSLT Custom File Format Generator" is the bonafide work of **Mr. Krishna Prathab R V, Register Number:** 9837S0053 and **Roll Number**: 98SE13, who carried out the project in our organization during the period December 2002 to March 2003. Certified further that to the best of my knowledge, the work reported there-in does not form part of any other thesis or work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Srinath Jagannathan
Director

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

To add meaning to the perception , it is my indebtedness to honor a few who had helped me in this endeavor, by placing th..n on record.

With profound gratitude, I am extremely thankful to **Dr.K.K.Padmanaban B.Sc (Eng),M.Tech,Ph.D.**, Principal, Kumaraguru College of Technology, Coimbatore for providing me an opportunity to undergo the M.Sc. [SOFTWARE ENGINEERING] Course and thereby this project work also.

I extend my heartfelt thanks to my CSE department head **Prof.Dr.S.Thangasamy B.E.(Hons),Ph.D.**, for his kind guidance and encouragement to complete this project successfully.

It's my privilege to express my deep sense of gratitude and profound thanks to **Mr.Srinath Jaganathan, Managing Director**, and **Mrs.Kalyani Jaganathan,Technical Director** Lucid Technologies Pvt Ltd, Chennai for having allowed me to do my project work in his esteemed team and for helping me in all means in successful completion of this project work.

Gratitude will find least meaning without a mention for **Asst Prof.Mrs.S.Devaki B.E., M.S. Mr.B.Sriram Balasubramanian, Project Leader** who have guided me with their valuable suggestions and consistent Motivations during my project work. I'm happy that I was able to give shape to their novel ideas to an extent.

Words are boundless for me to express my deep sense of gratitude and profound thanks to **Mr. Abhishek Sarkar and Mr. Karthik B** and all my associates at Lucid Technologies, for all their kind guidance and encouragement towards my project work. I've learnt quite a lot and acquired enough technical skills from their guidance without which, this project work could have not reached its successful completion.

Finally, this acknowledgement will not achieve its complete form if I don't remember my parent's sacrifices. Without their constant moral support, motivations and kind encouragements, I could have not channelised my career in the field of Computer science.

Above all, I dedicate this project to the Lord Almighty, who has been miraculously leading me throughout my life and career.

- Krishna Prathab R V

PROJECT ABSTRACT

This Project named "XSLT Custom File Format Generator" is done at Lucid Technologies Pv Ltd, Chennai, and the purpose of this project is to develop a system that will provide wireless service' based on the requirement of the client. The product will be customized to match the client's requirements This product entitled "Style Studio" is developed using Java , XML , Stylesheet and XSLT . Oracle i used as the backend.

The Style Studio(SS) is a XML based network application. This product is unique of its kin and has not yet been attempted to the best of our knowledge. So far XML parsers and transformers hav been existing as separate entities, in this product we have made an attempt to combine the features of bot of these technologies to develop a product which will address far beyond the needs of either the XML parse or the transformer. The system facilitates generation of custom files based on the request of the client th key feature of this product is that, it is WAP enabled which enables the client to access information from hi mobile device. The system contains modules for handling customer requests and generates appropriat responses based on the request. As a request from a customer is sent, the system identifies the type o request and generates a custom file format.

There are two main modules in the application, namely XML Generator and XSLT Custon File Format Generator. First module is meant to be at the back end wherein it will produce the appropriat XML response based on the request received. The second module will translate the XML and generate th output in the form requested by the user. The two modules operate independent of each other. The mai reason for such a separation is that each of the two modules after customization can be used as a plug-in int some other system with a similar requirement.

Though this product can find its use in various areas of Information Processing and e-commerce th current requirement emphasizes on the banking sector wherein based on the request from the client th system would generate custom files to address the customer need. The product as of now would be used b financial institutions to provide information on their bank accounts and would facilitate online transaction over the internet. The product will initially target the banking sector where the financial institution wi provide wireless services.

CONTENTS

CONTENTS

INTRODUCTION

1.1 Organization Profile - Lucid Technologies Pvt. Ltd.

Lucid Technologies was set up in 1997 with a single minded mission: To provide world-class IT services that enable clients to develop and market innovative products and services that rely on next generation technologies.... and win.

Our world class development centre in Chennai, India, is equipped with the latest technologies and solutions for enterprise networking, office productivity and collaborative software engineering. We are a Microsoft Certified Partner and a part of the Sun Developer Partner Program.

1.2 Project Overview - XSLT Custom File Format Generator

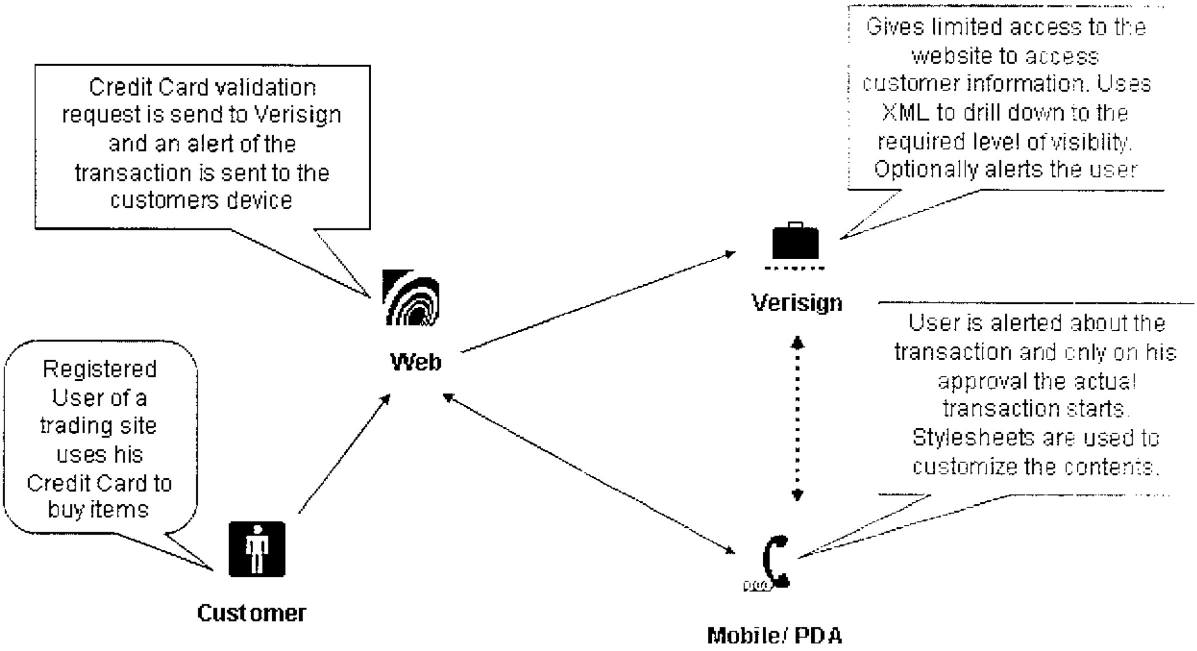
The XSLT Custom File Format Generator is a component of Style Studio (SS) which is a XML based network application. The system facilitates a client to provide internet based services which are WAP enabled wherein the customer will be able to use his browser or his mobile device/PDA to access information. The two main features of this product is the XML Generation and the XSLT engine which transform the XML document into various forms which is suitable for the end-user.

The system contains modules for handling various types of customer request's based on the client who is purchasing this product. There are mainly two modules namely XML Generator and XSLT Custom File Format Generator . First module is meant to be at the back end wherein it will produce the appropriate XML response based on the request received. The second module will translate the XML and generate the output in the form requested by the user. The two modules' operate independently of each other and can be used as independent entities and plugged-in into another application in need of a similar technology.

This product is unique of its kind and has not yet been attempted to the best of our knowledge. So fa XML parsers and transformers have been existing as separate entities, in this product we have made ar attempt to combine the features of both of these technologies to develop a product which will address fa beyond the needs of either the XML parser or the transformer. The key feature being the using of wireles technologies enables the user to perform secure transactions and hence paws the way for a reliable Credi Realization and Authentication for Online Transactions.

1.3 Business Scenario

The illustration below depicts a specific business scenario of an online transaction. In th scenario depicted below Verisign is used as an example , in the real application the client who holds financial institution would serve as a verifying authority and he provides limited access to th information requested by the web service.


Credit Card validation request is send to Verisign and an alert of the transaction is sent to the customers device
Gives limited access to the website to access customer information. Uses XML to drill down to the required level of visiblity. Optionally alerts the user
Web
Verisign
User is alerted about the transaction and only on his approval the actual transaction starts. Stylesheets are used to customize the contents.
Registered User of a trading site uses his Credit Card to buy items
Customer
Mobile/ PDA

This diagram is a self-explanatory illustration of an actual instance of the use of this application i real-time. The key feature of this product is that it is Machine-independent , cross-platform compatible , ha the power to transform XML data and data's from other sources into secure formats for the web such as XSI

, WAP and SVG. The product though initially aims at addressing mobile and wireless application can b extended for other media types such as media streaming supporting aural, print and display.

The product is hence totally customizable and addresses superior level of compression. A concep called BizCodes is used in XSLT Custom File Format Generator module which enables the same data to b disturbed to various clients but still can provide a layer of separation between the entities. Data hiding an encapsulation are used to its fullest power to provide customization.

SYSTEM ANALYSIS

2.1 Existing System

To the best of our knowledge there doesn't exist a similar system like this. With the implementatior of this system successfully the client would no longer need to depend on his financial institution nor his computer to bank and also transact money. He can bank and transact through his mobile device or his hand-held device with just a click of a button.

2.2 Need for the new System

To consider a simple scenario, consider a customer needs to view his account balance when he wants to write out a check to see if he has the required balance. As of now he should either approach the bank or login to the website to view his account balances but with this new system he can request for his balances from his mobile phone.

The traditional methods followed as of now posses security threats and a lot of dependencies exists which makes transactions insecure and non-reliable. Access is also restricted to the web and persona banking and globalization has not yet been fully implemented in either cases. To consolidate the requirements of the new system , the new system should address the following issues

1. **Content Customization** – Since this application mainly focuses on WAP enabled devices there should be different and separate models of information for various devices. Stylesheets should be used to customize information fro various type of devices and various models of mobile devices.

2. **Security** - Since the whole system revolves around transaction of money the system should be secure to the maximum possible extend. In a case where user request information from his mobile device there should be at least two checkpoints to ensure that the request is genuine. The first one of which should be the validation for his mobile number from which the request is sent and then his private PIN.

3. **Availability** – Availability turns out to be yet another important factor in the system the application should be available to the user at any point of time. A server uptime of at least 99% should be made possible to avoid delay in transactions and information.

4. **File Format Support** – The final system would handle queries from different types of sources. If a user is requesting information through his browser then the XSLT engine would transform the required data into a format suitable for displaying in the client's browser in this case the format would be an HTML file. If his request is from his mobile device then the XSLT engine would transform the data into WML format and send back it to the client's mobile device.

5. **Limited Access to Information** – The system should be modeled in such a fashion that there exists a single data store and multiple methods of retrieval based on the user. By this a big financial institution can have affiliates all over the world and can provide limited access to information from a single data store. The system would retrieve data selectively based on the BizCode and format data using stylesheets to the targeted receiver edge. Each of the requests should carry a BizCode to uniquely identify the request.

2.3 Proposed System - XSLT Custom File Format Generator

The proposed system would address all the above listed features with main emphasis on Security and Content Customization. The main system can be split into many individual modules each of which can be considered as plug-in modules for the other modules. The main core forms the XML Generator and XSLT Custom File Format Generator modules these are the two modules that handle the user requests, all the other modules form a support for these two modules and are built around these modules.

DETAIL DESCRIPTION :

The final system will incorporate two main modules and many sub-modules they are.

- **XSLT Custom File Format Generator**
  - XSLT Engine
  - HTML Transformer
  - Stylesheet Repository
  - WML Transformer
  - Content Customization Module
- **XML Generator**

The two modules will function independent of each other and will be plug-in modules fo each other. The other sub modules will support these main modules additional features such a PDA Transformers can be plugged-in at a later stage to this module based on the requirement therein.

XSLT Custom File Format Generator :

This is the core system which takes care of all the transformation process it receives reques from the client in form of HTTP requests if the request is from a web browser or WAP request if its from a WAP Browser or a mobile device , send the request to the XML Transformer. The XSL Transforme generates a XML file based on the request or generates an error code in an XML file. This file is ther converted into the clients target format and sends to the client. The description and functionality of the individual modules are listed below :

- **XSLT Engine :**

  This module contains the XSLT Transformer Engine which uses XSLT to structure data anc extract data from the XML file. It take as input the generated XML file and expels as output an XSL file which then includes a stylesheet for data structuring.

- **HTML Transformer :**

  This module contains the code which transforms the XML document into an HTML document. Before the transformation is done the target request is identified and the target device is identified. This module then pulls in the stylesheet which is specific to the requirement anc structures data based on the type of request. If the request is from Internet Explorer a differen stylesheet is pulled in and if it's from a Netscape Navigator then a different stylesheet is pulled in.

- **Stylesheet Repository :**

  This module is a virtual repository of stylesheets it contains within it all the types of stylesheets in variations for various type of browsers , various types of mobile devices and hand held devices When a transformation is needed stylesheets are pulled out of this repository. Almost all the stylesheets within this module are static entities and this contains two general templates which are called when the device or the browser type is unknown or new.

- **WML Transformer :**

  This module is similar to that of the HTML Transformer except for the fact that this module is responsible for handling WAP requests. Before the transformation is done the target request is identified and the target device is identified. This module then pulls in the stylesheet from the repository and structures data based on the type of device. If the request is from a Nokia device it gets the Model number and based on the model number and the manufacturer specification a stylesheet is pulled in from the repository . Supposing if the device is a Motorola or Siemens made then the respective model number is identified from the request and the corresponding stylesheet is pulled in.

- **Content Customization Module :**

  The main functionality of this module is to retrieve customized content for particular type of user. BizCodes are used to identify the financial institution and by using the BizCodes the data is structured to suit that particular client. Say if an associate of the main financial client has restricted access only to view the name of a customer then this property is identified using the BizCodes and the data is structured in a way that it contains only the name of the customer and not any other information.

XML Generator :

This module lies in the backend and is mainly responsible for all database related activities like entry, retrieval and maintenance. This module is made up of many sub modules that will be used for various operations. The main function of this module is to get the request and process it to check its validity Then based on the request the database will be queried or the error handler will be called a response either a normal response or a error response which will be in an XML Format will be generated
which will then be sent to the XSLT Custom File Format Generator Engine.

Java Engine :

This forms the core of the XML Generator this is the module which will interact wit all the other modules other than the data population modules. This will first get the request then i will submit the request to the error handler which will check the validity of the request anc authenticate the request. Then the request will again be processed by the engine which will ther process the nature of the request and call appropriate DB Handlers to get the required information Then this information extracted will then be used to generate an appropriate response which will be an XML which will then be sent to the XSLT custom file format generator.

DB Handler :

This module will be called by the engine in case of a database operation this module is mainly composed of beans which are used for database queries. The engine will call the appropriate DB Handler based on the nature of the request .

Error Handler :

This module is used for error handling and request authentication. Three main case may occur the first is that the database or the server may fail. In the second case the user may senc an invalid request. In the third case an invalid user may send a request. Such errors are to be handle and appropriate error responses are to be generated as XML's and the response is to be sent to th user.

Manual Data Population :

For all this requests the responses are generated by extracting the data from database The data in the database are to be constantly updated. For this we provide two means one is th manual data population where we provide a user interface for the client to add ,edit and delete th database entries .For this we need to provide security where unauthorized database access will b restricted.

Harvester :

This module is for automated data population. This is specifically for all banking sector clients where the account details will be extracted from the specific bank websites. This is an automated process which will simulate a user login into his financial website and scrape the data from the html presented there and place those account details in the database.

3.1 Computing Environment :

Server:

Web Server Software	Iplanet 4.1
Application Server Software	Weblogic 5.1
Server Operating System	Sun Solaris 2.8
Database Software	Oracle 8 RDBMS
Hardware	Sun SPARC Server ES 3000
Web Security Software	Web Security Management System

Client (Intranet):

Hardware	IBM PC Compatible Pentium III or later
Client Operating System	Windows 2000, Windows NT Workstation and Windows 2000 Server
Web Browser	Internet Explorer 5.0, Netscape Communicator 7.0 or later

Development:

Hardware	IBM PC Compatible Pentium III or later
Client Operating System	Windows 9x, Windows 2000, Windows NT Workstation
Web Browser	Internet Explorer 5.0, Netscape Communicator 4.6 or later
Development Language/ Tools	JAVA 2, Java Server Pages (JSP), Enterprise Java Beans (EJB) , JavaScript, Servlets, HTML

3.2 Technologies - Quick Reference

The following are the main factors in choosing the specific technologies and environments fo developing the product.

SUN SOLARIS

An Operating system's primary goal is to use the system's resources and hardware in a efficient manner and it's secondary goal is to make the system convenient to use. Solaris satisfie the both factors and hence it's preferred in all over the world and in this project also. It's mai salient features includes the following : (a) Multi tasking (b) Multi User and (c) System Portability 'Multitasking' means multiple tasks can be carried out by placing other tasks in the background while the user work on one task at a time. A multi user operating system permits several users to us the same computer to carry out their computing jobs. One of the outstanding features that Solari posses is the ability to port itself to another installation without the need to incorporate any majo changes.

HYPER TEXT MARKUP LANGUAGE (HTML)

HTML is basically a scripting language that's mainly used to display static contents on th Internet and Intranet applications, using Browsers. As a formatting language, HTML utilize SGML(Standard General Markup Language) declarations and the document type declarations (DTD). SGML document has three main parts. The first part defines the character set to be used and tells whic characters in that set distinguish text from markup tags. Markup tags specify how the viewer application, o browser, should present the text to the user. The second part of an SGML document specifies the documen type and states which markup tags are legal. The third part of an SGML document, called the documen instance, contains the actual text and markup tags. Because there is no requirement that the three parts of a SGML document reside in the same physical file, we can concentrate on the document instance. The We pages you create are document instances.

Most HTML browsers assume a common definition about the character set used, and about whicl characters distinguish text from markup tags. They also generally agree about a core set of legal markuj tags. They then diverge on which additional new markup tags to permit. In terms of universality, HTMI along with Hyper Text Transfer Protocol (HTTP) , is a base scripting language that's mainly used fo creating a Client – Server applications in Internet / Intranet arenas.

WEB LOGIC SERVER 5.1

As the industry leading e-commerce transaction platform, Web Logic server provides a number o features critical to developing and deploying mission-critical e-commerce applications across distributed heterogeneous computing environments. These include:

1. Standards leadership-Comprehensive Enterprise Java support, including EJB and JMS, to ease th implementation and deployment of application components.

2. Enterprise e-business scalability-WebLogic Server's software clustering of dynamic web pages (Servlets JavaServer Pages)

3. EJB business components, coupled with client connection sharing and database resource pooling maximize efficiency in the use of critical resources.

4. Robust administration-WebLogic Server offers a comprehensive pure-Java console operation, Zer Administration Client (ZAC) for managing the distribution of applications to remote users, and dynami application partitioning and cluster membership.

5. E-commerce-ready security-WebLogic Server features Secure Sockets Layer (SSL) support fo integration of encryption and authentication security into e-commerce solutions.

6. Maximum development and deployment flexibility-WebLogic Server features tight integration with an support for leading databases, development tools, and other environments.

WebLogic Server operates at the center of a multitier architecture. In this architecture, busines logic is executed in WebLogic Server, rather than in client applications. The resulting "thin" client, thre tier architecture allows the client to manage the presentation layer, the application server to manage th business logic and the back end data services manage the data. This makes WebLogic Server the idea platform for web-enabled e-commerce applications.

XSLT

XSLT's central advantage is that it was designed for use with XML. XSLT is the only programmin language standardized specifically for processing XML. More and more processing can be moved into th XSLT domain as more and more data is represented or transferred as XML. The specific advantages o XSLT are listed below :

Reliable XML Model :

XSLT has a well defined XML data model that requires all compliant XSLT engines to parse XMI in exactly the same way. Scripting languages tend to depend on the DOM or on a non-standard API. Eve when they depend on the DOM, they do so in a variety of ways. Some normalize text nodes when th document is loaded. Some do not. Some support CDATA nodes. Some do not. Some discard entit boundaries. Some preserve them.

Although XSLT processors are much more predictable and reliable in their parsing, there are two variants o XSLT processors. Those that use validating parsers can differ somewhat from those that use non-validatin ones. Scripting languages and traditional programming languages also have this problem. In fact th problem stems from XML itself. Portable XML-processing code should not rely on the existence of th DTD.

Tree Walking :

XML documents are trees. XSLT is organized around the notion of walking down trees from th root nodes to the leaves and from the start of the document to the end. XSLT is therefore very convenien for processing documents from top to bottom as is often necessary for conversion and formatting tasks.

The default behavior of even an empty XSLT transformation is to walk down the XML tree from th start of the document to the end, examining each element and outputting each text node. Many simpl transformations consist merely of adding some functionality to that process. Because XSLT by default doe the document traversal that most people want, traversal code is minimized.

On the other hand, if you want to do some form of document traversal that is not like the one buil into XSLT, your task will get harder. For instance if you wanted to process the document backwards o traverse from link references to their referents, this would be a little more difficult.

Templates:

XSLT documents are XML. This means that there is a very natural syntax for representing the XMI elements to be generated in the output: XML elements! An XML-generating program in a scriptin language template would typically look something like this:

```
$var = setup_some_variable();
output("""
<somexmlelement attribute="$var">
   <etcetc>
 """)
do_some_processing()
output("""
   </etcetc>
</somexmlelement>
 """)
```

XSLT is designed to move back and forth between code and data more seamlessly. For example computing an attribute value can be done right in a template for the attribute value. As another example XSLT has a fairly natural built-in way of representing namespaces. It just uses the XML namespace declared in the XSLT document. An XSLT equivalent of the code above is all in XML syntax.

```
<somexmlelement attribute="{calculate/some/value}">
  <etcetc>
     <xsl:apply-templates/>
   </etcetc>
</somexmlelement>
```

With one important exception, scripting languages are not as well set up to do templating. Th exception is PHP. PHP has excellent support for templating. It was designed as an HTML templatin language but also works fine for XML. Most other scripting languages do have special templatin extensions that can be used to approximate the templating ease of PHP.

XSLT's XML heritage is not a pure advantage, however. There are times when it can b inconvenient. For one thing it is always somewhat verbose. For another, controlling the generation o whitespace can be somewhat tedious for output formats where that matters. XSLT has no direct way t generate entity references, including named character references.

Pattern Matching :

XSLT templates are triggered based on patterns expressed as XPaths. XPath is an extremely ric and sophisticated language for node matching and selection. Scripting languages can be extended wit libraries that incorporate XPath engines. But it is not common for scripting languages to have XPath engine (or anything of equivalent power) until somebody has implemented an XSLT engine in that language. XSL seems to always be the impetus to improve a language's XML processing power to the point where th language competes with XSLT.

Optimization Opportunities :

A language like XSLT offers both opportunities and difficulties in terms of performance. Th primary opportunity is that XSLT is much more constrained and less flexible than a general purpos language (especially a scripting language). For instance XSLT disallows one template from overwriting variable that is potentially in use by another template. That means that a variable can be looked up once an have its value cached somewhere (even in compiled code for the template itself) rather than having th value be looked up over and over again in some lexically scoped namespace. Also, XSLT is much easier t implement than a scripting language because its domain of application is so much more constrained. Tha leaves an implementer more time to pay to attention to optimization techniques.

Scripting languages can also have performance advantages in some circumstances. Where XSL puts performance tweaking into the hands of the engine implementer, scripting languages can give th transformation programmer more leeway for performance optimizations in a particular transformation. Al else equal, it is better to do performance optimizations once in an XSLT engine instead of over and ove again in each individual transformation. Practically speaking, however, the transformation programmer ha more knowledge about the performance requirements of the code they are writing than the engin implementer does. It is therefore appropriate in some cases for the transformation programmer to want t take on responsibility for performance.

Ubiquity :

Being embedded in an important system is a surefire way for a language to become a success Languages like DOS batch files, Unix shell scripts, JavaScript and Microsoft's Basic variants surely did no become popular based purely on their technical merits. It was through being integrated into an importan system (DOS, Unix, Netscape and Visual Basic) that these languages became important.

XSLT is similarly embedded throughout the XML infrastructure. Databases embed XSLT, Interne Explorer and Windows XP ship with XSLT implementations. Most Linux distributions probably ship wit at least one by now. This ubiquity is a remarkable achievement that has not yet been matched by any genera purpose scripting language (which is to exclude JavaScript).

WAP :

The Wireless Application Protocol (WAP) is an open, global specification which gives mobile user with wireless devices the opportunity to easily access and interact with information and services. Th protocol is developed by WAP Forum http://www.wapforum.org/, an organization of some of the mos powerful Internet and telecom companies.

The advantages of WAP are that if you have a mobile phone you will probably have noticed tha your phone is a more "private" device than your computer. Your phone is always within reach. you can ge hold of all your friends using it and you have a lot of personal information in store. As opposed to your PC the WAP phone will always be within range. It invites you to order something wherever you are, and yo can search for information in the middle of a discussion around the café table. You can check your e-mail o the run, and you can be in touch with the Intranet in your office anytime.

XML

XML stands for the eXtensible Markup Language. It is a markup language, developed by the W3C (World Wide Web Consortium), mainly to overcome limitations in HTML. The first version of HTML ha a dozen tags; the latest version (HTML 4.0) is close to 100 tags (not counting browser-specifi tags).Furthermore, a large set of supporting technologies also has been introduced : JavaScript. Java, Flash CGI, ASP, streaming media, MP3, and more. Some of these technologies were developed by the W3C whereas others were introduced by vendors.

However, everything is not rosy with HTML. It has grown into a complex language. At almost 10 tags, it is definitively not a small language. The combinations of tags are almost endless and the result of a particular combination of tags might be different from one browser to another. Finally, despite all these tag already included in HTML, more are needed. Electronic commerce applications need tags for produc references, prices, name, addresses, and more. Streaming needs tags to control the flow of images an sound. Search engines need more precise tags for keywords and description. Security needs tags for signing The list of applications that need new HTML tags is almost endless. However, adding even more tags to a overblown language is hardly a satisfactory solution.

XML was developed to address these shortcomings. It was not introduced for the sake of novelty XML exists because HTML was successful. Therefore, XML incorporates many successful features o HTML. XML also exists because HTML could not live up to new demands. Therefore, XML breaks nev ground where it is appropriate. It is difficult to change a successful technology like HTML so, no surprisingly, XML has raised some level of controversy.

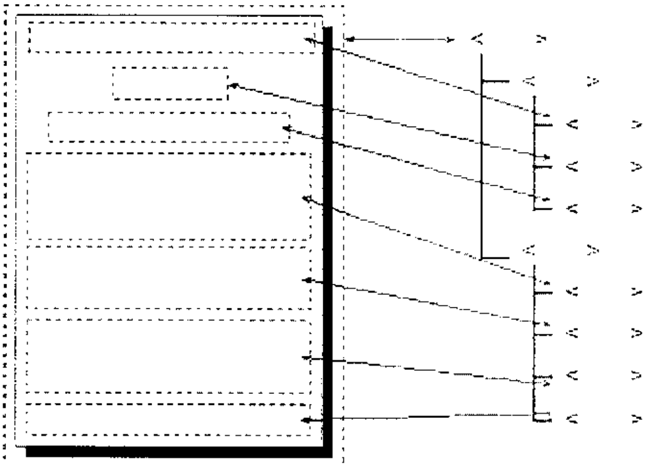
Some of the areas where XML will be useful in the near-term include:

- Large web site maintenance. XML would work behind the scene to simplify the creation of HTML documents

- Exchange of information between organizations
- Offloading and reloading of databases
- Syndicated content, where content is being made available to different Web sites
- Electronic commerce applications where different organizations collaborate to serve a customer
- Scientific applications with new markup languages for mathematical and chemical formulas
- Electronic books with new markup languages to express rights and ownership
- Handheld devices and smart phones with new markup languages optimized for these "alternative" devices

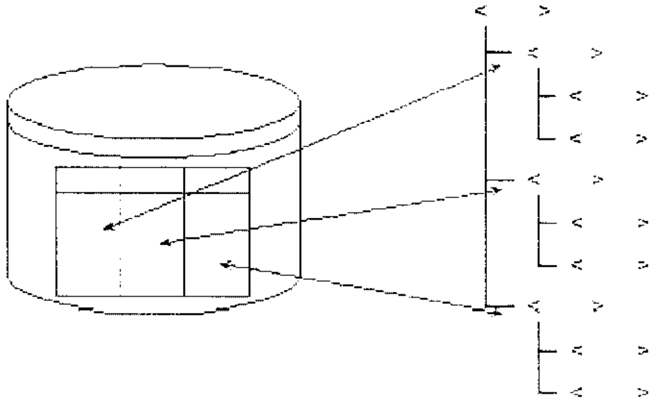
Document Applications

The first application of XML would be document publishing. The main advantage of XML in this arena is that XML concentrates on the structure of the document, and this makes it independent of the delivery medium. XML is independent from the medium.Therefore, it is possible to edit and maintain documents in XML and automatically publish them on different media. The operative word here is automatically.The ability to target multiple media is becoming increasingly important because many publications are available online and in print. Also, the Web is changing very rapidly. What is fashionable this year will be passé next year so one needs to reformat his site regularly.



The structure of a document in XML

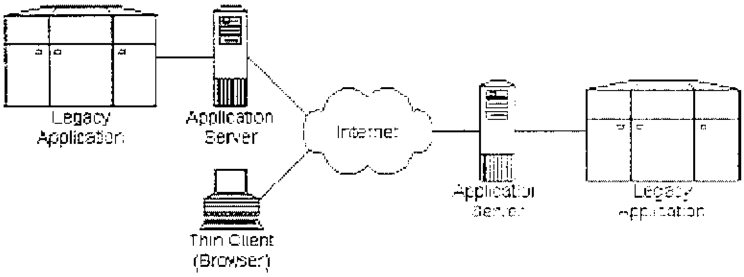
Finally, some Web sites are optimized for specific viewers, such as Netscape or Internet Explorer This often leads to the development of two or more versions of the same site: one generic version and one optimized for some users. If done manually, this is very costly. For all these reasons, it makes sense t maintain a common version of the documentation in a media-independent format, such as XML, and t automatically convert it into publishing formats such as HTML, PostScript, PDF, RTF, and more. O course, the more media we need to support and the larger the document, the more important it is tha publishing be automated. Data Applications One of the original goals of SGML was to give documen management access to the software tools that had been used to manage data, such as databases. With XML the loop has come to a full circle because XML brings a publishing kind of distribution to data. This lead to the concept of "the applications the document" where, ultimately, there is no difference betwee documents and applications.



The structure of a database in XML

XML is used to exchange information between organizations.The XML Web is a large database o which applications can tap. This can be viewed as an extension for extranets. The idea behind an extranet i that one organization publishes some of its data on the Web for its partners. For example, an organizatio will publish its price list on its Web site. In some industries, such as electronics, the price list is ver dynamic. Prices can change several times during a month. If the information is available on a Web site customers can always access the latest, most up-to-date information. Currently, the price list is published i

HTML—that is, intended for viewing by a human. This is acceptable if you have few providers with fev products but as soon as you have many providers or many products, you want an automated solution. Witl XML, software can automatically visit the price list, extract the price, and update the information in you own database. This is shown in the Figure. It requires a markup language that does not concentrate o appearance but on structure.

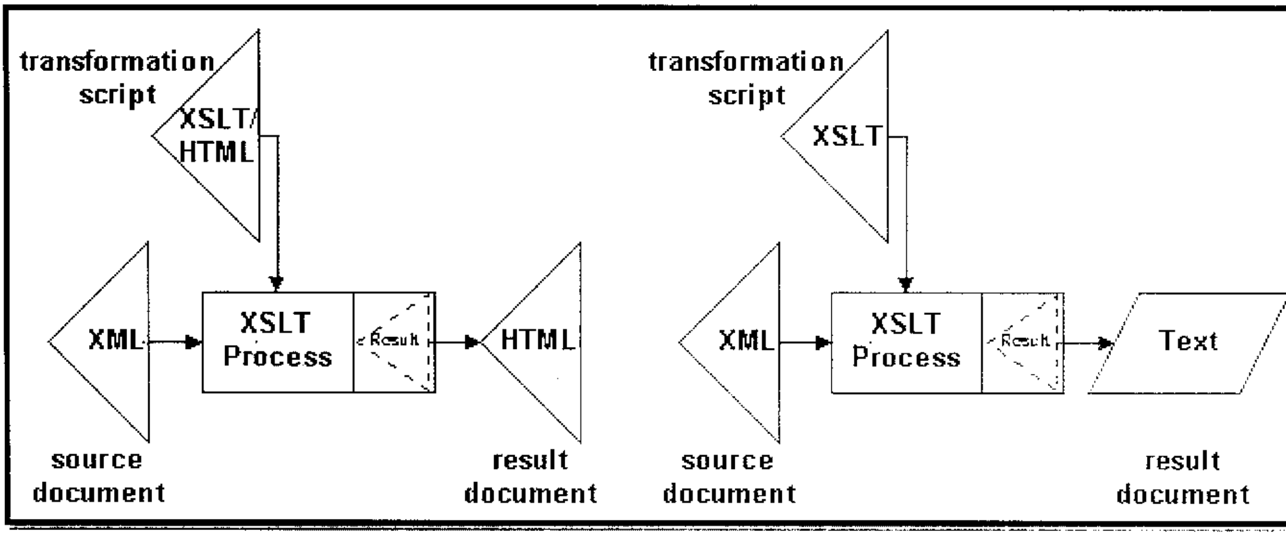


Applications exchanging data over the Web

SYSTEM DESIGN

4.1 Design Architecture :

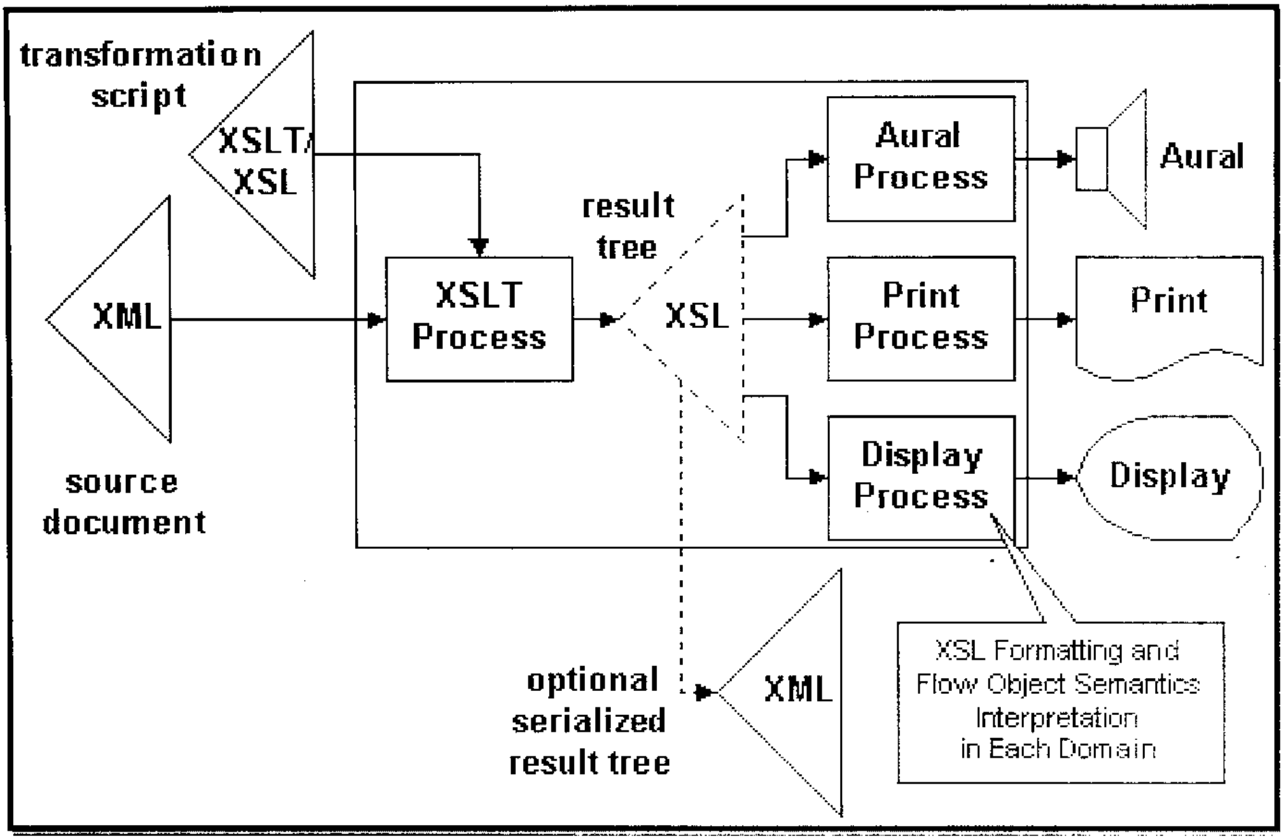
Simple Transformation Architecture :



There are two types of transformation architecture that can be adopted in this project. The XSL Custom File Format Generator is made of the following components namely the XSLT Engin /XSLT Process, the transformation script and the result set or the Content Customization Module. Th XML file generated is taken as the input in the XSLT Process the request type is identified and th transformation script i.e., WML Transformer or HTML Transformer is invoked based on the request type.

The Transformer then identifies the device/browser and pulls-in the respective stylesheet an produces the XSL file. This XSL file is then used to restructure the data in the XML file and finally th Content Customization Module customizes the content based on the user requirement and generates th HTML/WML file. A plain text format generation is also possible through the simple transformatio architecture.

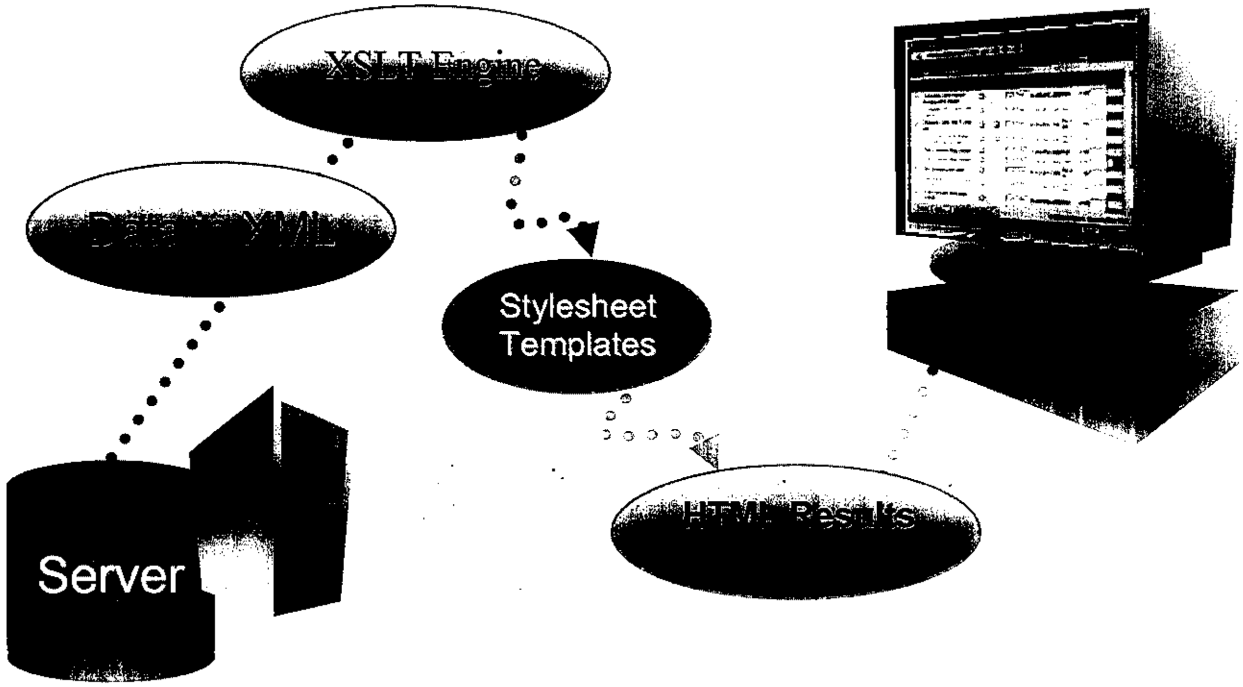
Advanced Transformation Architecture :



The diagram above depicts the advanced transformation architecture which is planned to be done i the future. Though some components of this architecture are built already the aural process would requir more effort. In this model all the procedures that apply for the simple transformation architecture are th same except that this system can generate an optional serialized XML result tree which may be then use as a mature data store. The result tree is also an XSL and the XSL file is then used to model data for th various media types such as Aural, Print and Display.

With more effort and time put in to the existing system. The existing system can be made suitabl for aural data transmission with the emergence of concepts such as VoiceXML the process becomes easie as everything in the system is manipulated in terms of XML. This would be the future modification to th system though this is not build up in the existing system.

General Architecture :



4.2 Screen Design

There are three main screens in this project , one for loading the generated XML and converting it into HTML/WML another for previewing the result and the last for applying the template to resultant HTML / WML file and generating the target file after getting the stylesheet from the repository.

HTML/WAP Transformer :

This screen is used to load an XML/XSL and transform it into the target format namely WML HTML. The conversion at this point of time does not take into consideration the transformation process attaching the stylesheet this screen is used only for the conversion into HTML/WAP format. When an XS file is loaded instead of the XML file this screen would convert it into the respective format. Since XSL fil have in them already applied styles the conversion is direct.

The user can perform the following functions using this screen:

- Load an XML file and convert it into the resultant format
- Load an XSL file and convert it into the resultant format

WAP Previewer :

As the name implies this screen is used to preview the output in WML format for various mobil devices. As of now only Nokia and Alcatel models of mobile output can be previewed in this screen. Th main use of this screen is that the user can preview the output in real-time. Instead of using a real mobi device this screen simulates a mobile device by showing the user how the WML file generated would loo like in the mobile device. Since Nokia mobiles turns out to be the most widely used mobile bran throughout the world Nokia is considered to be base model for development and all the general template are based on the Nokia model. The screen also contains hot spots in the page which simulates the navigatio in the mobile device. The user can press the navigation keys on the simulated mobile phone and can t navigating as he would do through his mobile device. He can scroll through the text and press on th hyperlinks to navigate to the next level.

Stylesheet Embedder :

In this screen the user can select the stylesheet from the repository and try applying it to the XM file and can generate the resultant HTML/WML file. The user loads the generated XML file and the choose the stylesheet that he wants to apply to the XML file and on pressing the translate button the resultan HTML/WML file is given as the output after the application of the stylesheet in the document. Say fo example if need to generate various WML file for different mobile versions from the same XML file. I such a case the user selects the generated HTML file from the XML file and chooses the various Styleshee he wants to be applied to the file and then presses the translate button, now the WML files are named i sequence and generated.

4.3 Detailed Design

Detailed design of a system includes developing prototypes, User interfaces and Backen databases. For this phase, Data Flow diagrams (DFD), Entity Relationship diagrams (ERD) and Syster Flow Charts (SFC) are used.

Data Flow Diagrams depict how data interact with a system. DFDs are extremely useful i modeling many aspects of a business function because they systematically subdivide a task in to its basi parts, helping the Analyst understand the system, which they are trying to model.

A DFD models a system by using external entities from which data flows to a proces which transforms the data and creates output data which goes to other processes on external c files. Data in files may also flow to processes as inputs.

The main merit of data flow diagram is that it can provide an overview of what data a syster would Process, what information of data are done, what files are used and where the results flow. Th graphical representation of the system makes it a good communication tool between the user and a analyst. It's difficult to represent the business process through verbal description alone. Here data flo diagrams help in illustrating the essential component of a process and the way they interact.

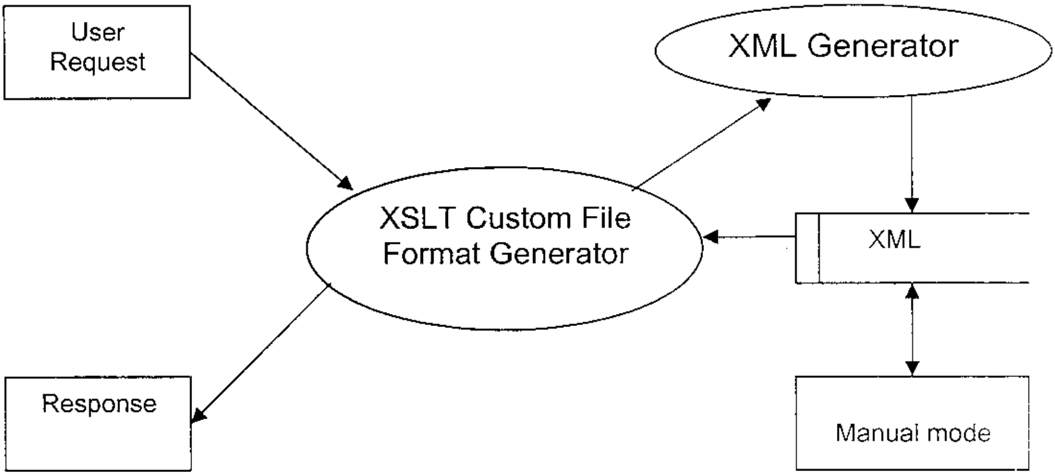
DFD Components

DFDs are constructed using four major components : (a) External entities (b) Data stores (c Processes and (d) Data flows. External entities represent the sources of data that enter the system c the recipients of Data that leave the system. Data store represent stores of data within the system. may be a Databases or individual files. Processes represent activities in which data is manipulated b being stored or retrieved or transformed in some way. Data flows represent the movement of data betwee other components, for example a report produced by a process and sent to an external entity.

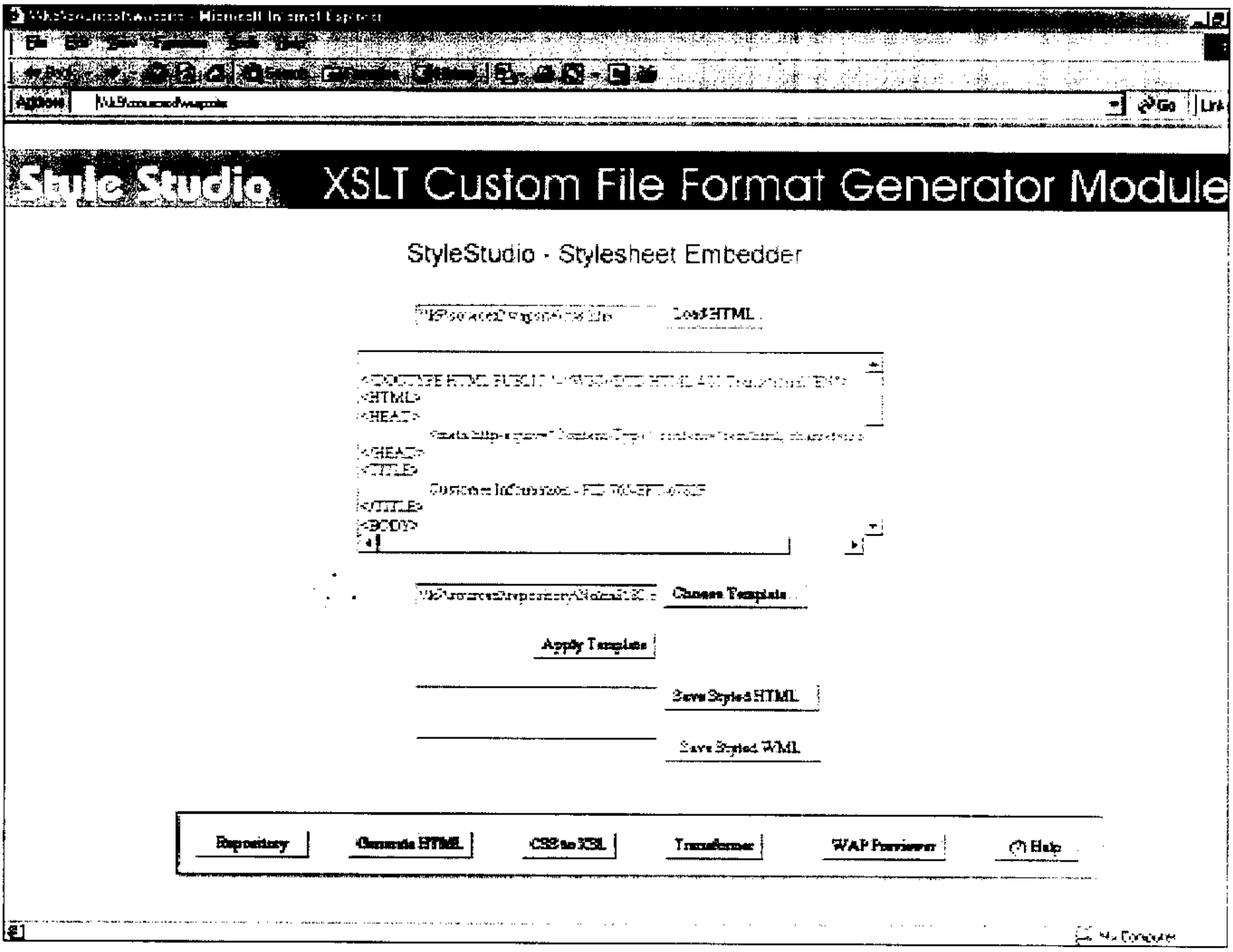
A circle is used to depict a process. Both input and output are dataflow. An arrow represents th data flows. External entities are represented by rectangles. Entities supplying data are known a *Sources* and those that consume data are called as *Sinks*

4.4 Data Flow Diagrams

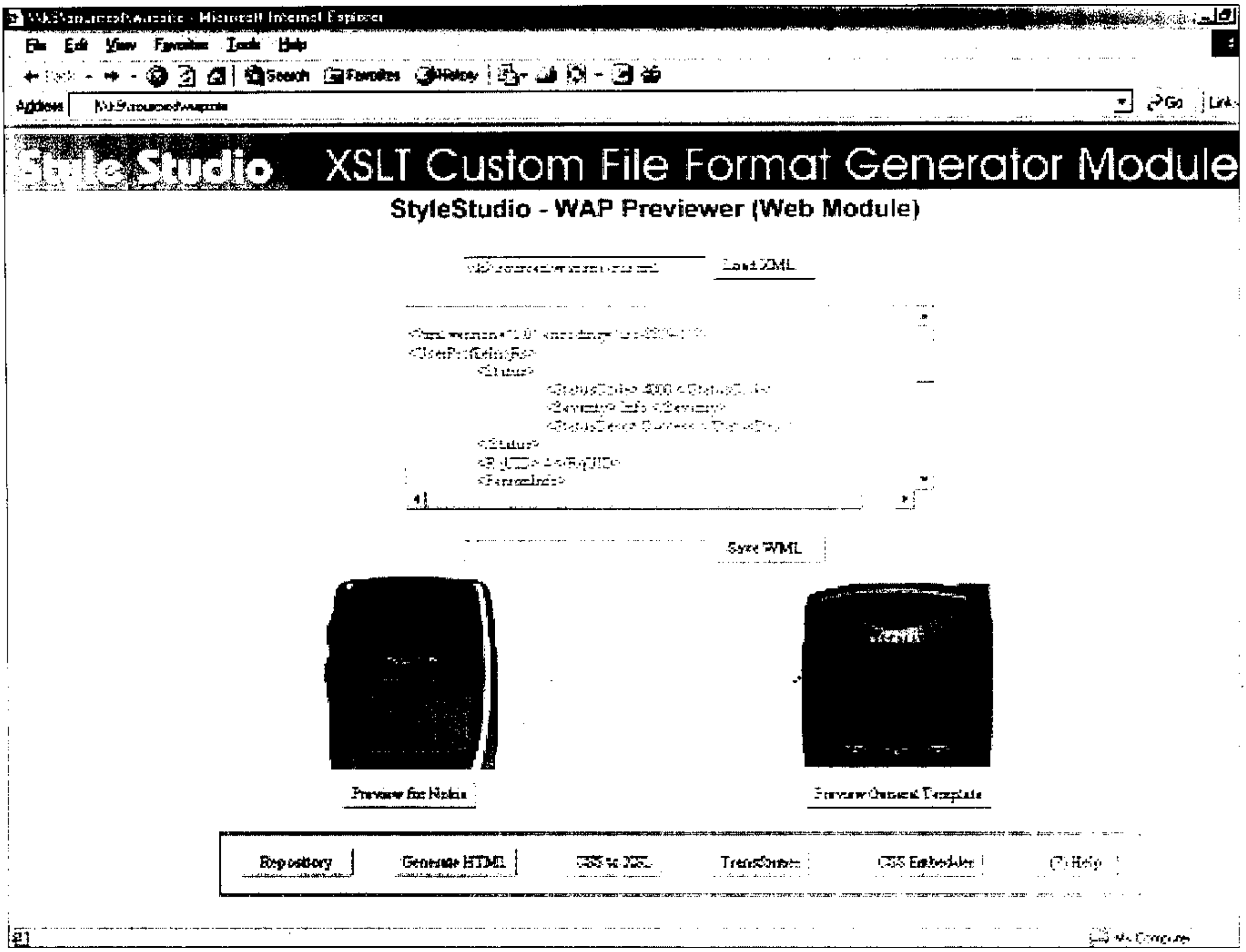
Context Level Flow



9.2 Important Screen Shots



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
        <meta http-equiv="Content-Type" content="text/html; charset=[illegible]
</HEAD>
<TITLE>
        Customer Information - [illegible]
</TITLE>
<BODY>
```



StyleStudio - WAP Previewer (Web Module)

```
<?xml version="1.0" encoding="[illegible]"?>
<UserProfileInfoRs>
        <Status>
                <StatusCode>4000</StatusCode>
                <Severity>Info</Severity>
                <StatusDesc>Success</StatusDesc>
        </Status>
        <RqUID>1</RqUID>
        <PersonInfo>
```

9.3 References

Websites :

1. **www.w3schools.com**
2. **www.wapforum.com**
3. **www.xmlspy.com**
4. **www.sun.java.com**

Books :

1. Java - Complete Reference,

   - Patrick Naughton and Herbert Schildt,Tata McGraw-hill, Third edition, 2000.

2. Oracle 8.0 - User Manual

   - Oracle Corporation Press , 2000

3. Software Engineering

   - Roger Pressman,Tata McGraw Hill edition,1998.

4. Java Script

   - JavaScript Workshop, Laura Lemay and Michael G. Moncur,Prentice hall.