AN EFFECTIVE STRUCTURE FOR MAZE TAB LOGIC USING DYNAMIC ARCHITECTURAL DESIGN

P.Malar[1], *X. Agnise Kala Rani*[2]

Abstract

Billing communication is based on credentials (invoices, debits, credits, and payments). It integrates REST into the MVP architecture and uses Ajax to deploy a dynamic, client-side view layer that lets the new tabbing system's users add and search documents, generate invoices based on usage and account balance, and generate reports. It also offers a set of Web services for interfacing with third-party vendors and supports a wide range of languages. The existing tabbing system based on model view controller has some disadvantages such as real time implementation is difficult. This paper deals with the design of Internet tabbing system based on model view presenter pattern to overcome the problems in MVC and it is possible pay invoices electronically.But theModel-View-Presenter (MVP) pattern which is designed at providing a cleaner parting between the View, the Model and the Presenter.

Index Terms : MVP, PHP, E-Tabbing System, Dynamic Maze Architecture

I. Introduction

Tabbing systems are key competitive weapons for telecommunications companies [1]. A tabbing system is a combination of software and hardware that receives call detail and overhaul usage information, assemblage this information for exact accounts or customers, produces invoices, creating reports for management, and recording (posting) payments made to customer accounts. Tabbing systems are composed of boundary (Network, Advertising, Client Care, Business, etc.), Computers are the hardware (computer servers) and OS (Operating System) are used to run the task and methods. Network'scrossing pointis the hardware devices that gather accounting information (usage) from multiple networks, convert it into detailed tabbing records, and pass it on to the tabbing system.

Internet Tabbing or electronic tabbing promises far more than a new and inexpensive way to deliver tabbing information. Business specialists forecast that Internet Tabbing will fundamentally change the way firm interrelate with their clients. Finally, the Internet Tab will be an interactive entry to a host of additional services including customer self-care, automated sales one-to-one marketing. The Internet Tab will become the gateway through which customers and companies have electronic one to one dialogs[2].

Electronic tabbing is one of the fastest growing technologies for corporate law departments. Recent surveys indicate that roughly 15 percent of corporate legal departments require electronic tabs from their law firms, and another 15 percent are considering it. If the person is a law firm with corporate clients, the person have probably

[1]Research Scholar, Karpagam Academy of Higher Education, Coimbatore - 641 021, Tamil Nadu, India. Email : malar.prp@gmail.com

[2]Research Scholar, Professor, Department of Computer Applicatios, Karpagam University, Coimbatore - 641 021, Tamil Nadu, India. Email : agneskala72@gmail.com

seen acceleration in the number of requests from clients who want their tabs submitted electronically. Choosing electronic tabbing and matter management systems are among the most important technology decisions that a law department can make, with significant potential consequences both positive and negative [3].

The concept of electronic tabbing is not new. Since the advent of the Internet, a small number of consumers have been using this electronic medium to pay tabs online after receiving standard paper invoices via regular Postal Service. What is new in the electronic tabbing arena is the concept of electronic tab presentment. With electronic tab presentment, companies that send tabs post consumers' statements to the Internet, enabling consumers to view the statements and make e-payments.

With ever rising spread of Internet, Tab presentment and expense is becoming a new type of overhaul area for sporadic billers like Telephone Companies, Electricity etc. Internet pedestaltab presentment and payment system translates tabbing axis from cost axis to returns axis and for customers (payer) the system is a modified overhaul. Internet based tab presentment and compensation system gives direct modified communication control between Billers and Payers, opens a new proceeds channel by cross-selling commercial. Drastic decrease of costs that are connected with paper based tabbing system. For customers or payers, receiving tabs to payment of tabs at one window through a Personal Computer, figure 4 shows the on-line tabbing system.

In this research new pattern is proposed for tabbing system based on web applications.

RELATED WORKS

J Crookes (1996) [1], the term adopted for the system is multiservice billing system (MSBS). The strategic business issues are shaped the design of MSBS. It describes the scale and complexity of the problem which makes the construction of a multiservice platform such a difficult feat of software engineering. The concept of a common product model, which underpins the system's design, is introduced.

NN Murthy, et al.(2000) [4], In their paper, the authors presented a brief description of the technologies for e-commerce. The authors also present TWINS (Twin Cities Information Network Service) test-bed application being developed as part of this project. TWINS, operational at twin cities of Hyderabad-Secunderabad, facilitates payment of various utility bill payment (like water, electricity, etc.) through a single window system. Payment of water bills through Internet using E-Cheque (Electronic Cheque) will be operational soon. This enables customers to pay their bills from anywhere, anytime. Thus the realizing benefits of e-commerce to the citizens.

Yang Bo, Liu Dongsu and Wang Yumin (2001) [5], in their paper, the authors improved the e-payment system with a smart card proposed by S.Brands, and present an anonymity-revoking e-payment system. On the one view, the customer's privacy cannot be negotiated by the bank or by the payee. On the other hand, anonymity can be removed by a TTP with the help of the bank. In this case, the third party can link a payment to a corresponding withdrawal and prevent money laundering and blackmailing.

EWB Team (2000) [6], This document provided information regarding the use of the Extra Work Billing System (EWB). The document is organized with step-by-step instructions for each task to be accomplished using the EWB system. The EWB System may be accessed through the Internet using either Netscape Navigator or Internet Explorer.

P.S. Barreto, et al. (2005) [7]. In their paper, the authors presented a discussion concerning the performance of four network scenarios for tabbing purposes. Using the results of packet losses in an experimental platform simulating a NGN (Next Generation Network) environment, the authors evaluate on each scenario the impact in the tabbing process with different traffic flows comparing the total revenue calculus for two tabbing schemes: (1) charging per packet and (2) reducing the value corresponding to undelivered packets. Our results show that the environments that use Differentiated Services are both convenient for costumers and service providers.

Shiqun Li , et al. (2008) [8]. In their paper, the authors first identified some vulnerability in the mobile tabbing system. Then, the authors propose a fair and secure tabbing system based on a proper combination of digital signature and hash chain mechanism. The proposed system can achieve authentication, non-repudiation, and fairness, which are desirable security requirements for an undeniable mobile tabbing system.

Albert Levi, Cetin Kaya Koc (2009) [9], In their paper the authors proposed a new Internet e-payment protocol, namely CONSEPP (Convenient and Secure E-Payment Protocol), based on the account authority model of ANSI (American National StandardsInstitute) X9.59 standard. CONSEPP is the specialized version of X9.59 for Internet transactions (X9.59 is multi-purpose). In CONSEPP the authors propose a light weight method to avoid the need for merchant certificates. Moreover, the authors propose a simple method for secure shopping experience between merchant and consumer. Merchant authentication is embedded in the payment cycle. CONSEPP aims to use current financial transaction networks, like Visa Net, Bank Net and ACH (Automated Clearing House) networks, for communications among financial institutions. No certificates (in the classical sense) or certificate authorities exist in CONSEPP.

Giannakos Antoniou, et al. (2009) [10], In their paper, the authors proposed an online payment scheme which uses the traditional e-payment infrastructure but which reveals no payment information to the seller. This is done with only an incremental increase in computational power.

II. Methodology

2.1. REPRESENTATIONALSTATE TRANSFER (REST)

Roy Fielding [11] described constraints and design decisions behind the World Wide Web as an architectural style pattern named Representational State Transfer, or REST for short, that was applied to the design of the Hypertext Transferprotocol (HTTP) [12] and Uniform Resource identifiers(URI) [13].

The REST approach aims to support distributed hyper media systems that scale to Internet size, with general interfaces and intermediary components that reduce interaction latency [11].

Some of the fundamental parts of REST when appliedto HTTP and URIs are:

- Resources are targets of references and are identified by resource identifiers in the form of URIs

- Resources can have different representations (formats), e.g. HTML, XML, JSON (JavaScriptObject Notation), plain text, serialized Java objects. The representations are sent in the body of themessage. In the HTTP protocol representationmetadata sent in the HTTP header contains things like media (MIME) type. HTTP header fields canal so contain cache directives, as exemplified in the'performance' section below.

- A predefined set of methods (corresponding toverbs) that acts on resources. The method is supplied in the client's request. Methods like OPTIONS and GET should not lead to changes at the server side, so results from those methods can usually be cached. The GET method is used to retrieve are presentation of the resource that the URI identifies, which is how most normal web pages are fetched. The PUT method replaces content at the target URI (or creates it if missing), and DELETE deletes resources. The POST method is often used to modify or add information, or to perform other operations determined by the server.

- A response contains headers, a HTTP Status Code and possibly a body. Status codes indicate things like success (including '200 OK', '201 Created', etc.),redirection (including '301 Moved permanently', '303 Seeother', '304 Not Modified'), errors in the client request(including the familiar '404 Not found' and the '412Precondition Failed' that can be returned when conflictsblock conditional updates), and server errors (including'500 Internal Server Error'). Responses indicating redirection or creation of new resources should containa 'Location' header field with the target URI.

2.2. PHP

PHP provides a light weight, clean syntax, making the code easy to read and thus easier to maintain.PHP also has a good documentation system that lets us generate navigable HTML and PDF documents. PHP design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. This reducesthe amount of time spent writing and maintaining documentation, so we're more likely to keep them up to date. PHP uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of PHP is dynamic name resolution (late binding), which binds method and variable names during program execution.

Furthermore, PHP offers readily available,stable Web frameworks. Django is particularlysuitable,[14] because its URL dispatcher serves wellas the naming authority for a system.Through this, the URL (ourresource identifier) doesn't change regardless of system changes is ensured. The dispatcher also makes iteasy to deploy REST Web services, where individual requests can be dispatched to different PHP methods based on the URI and HTTPrequest method.

2.3. AJAX

For the front end, Ajax brings more interactivity to the user experience. Once loaded on the client, Ajax-powered views can respond to user actionsby requesting data from the back-end services asynchronously, resulting in a smoother user experience[15].

In addition, third-party vendors can selectively wrap our Web services. For example, if acompany wants to customize only a part of the system (invoice), they would only have to develop a new set of views using raw data fromour /invoice/ interface. All noncustomized views can be configured to retain the default Ajax version.

2.4. MODEL VIEW CONTROLLER

Model–view–controller (MVC) is a software pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. The central component, the model, consists of application data, business rules, logic, and functions. A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. The third part, the controller, accepts input and converts it to commands for the model or view.

Model: The domain-specific representation of the information on which the application operates.The model is another name for the application logic layer (sometimes also called the domain layer). Application (or domain) logic adds meaning to raw data (e.g., calculating if today is the user's birthday, or the totals, taxes and shipping charges for shopping cart items). Many applications use a persistent storage mechanism (such as a database) to store data.MVC does not specifically mention the resource management layer because it is understood to be underneath or encapsulated by the Model.

View: Renders the model into a form suitable for interaction, typically a user interface element. MVC is often seen in web applications, where the view is the HTML page and the code which gathers dynamic data for the page.

Controller: Processes and responds to events, typically user actions, and may invoke changes on the model and view.

1. The user interacts with the user interface in some way (e.g., user presses a button)

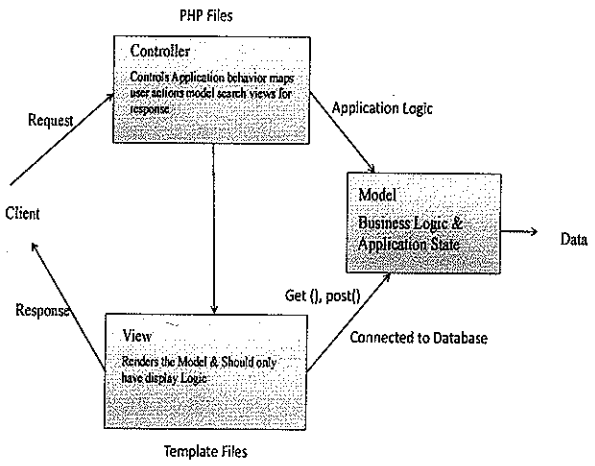
2. A controller handles the input event from the user interface, often via a registered handler or call back.

3. The controller accesses the model, possibly updating it in a way appropriate to the user'saction (e.g., controller updates user's shopping cart). Complex controllers are often structured using the command pattern to encapsulate actions and simplify extension.

4. A view uses the model to generate an appropriate user interface (e.g., view produces a screenlisting

the shopping cart contents). The view gets its own data from the model. The model has no direct knowledge of the view. (However, the observer pattern can be used to allow the model to indirectly notify interested parties, potentially including views, of a change.)

5. The user interface waits for further user interactions, which begins the cycle anew.


PHP Files
Controller
Controls Application behavior maps user actions model search views for response
Request
Application Logic
Client
Model
Business Logic & Application State
Data
Response
Get (), post()
View
Renders the Model & Should only have display Logic
Connected to Database
Template Files

LIMITATIONSOF EXISTING TABBING SYSTEM USINGMVCPATTERN

Even if this pattern comes with a lot of advantages, there are distinct disadvantages too. Firstly, it is too complex to implement and is not suitable for smaller application; rather, implementing this pattern in such applications will have adverse effects in the application's performance and design. In this regard, [18] the MSDN states, "You are building a Web application in Microsoft ASP.NET, and, based on the complexity of your application, you need to separate different aspects of the program to reduce code duplication and to limit the propagation of change." This is where this design pattern fits in.

2.5. MODEL VIEW PRESENTER

Model-View-Presenter (MVP) is astructural pattern for the presentation layer of software appliances. The prototype was originally implemented at Taligent in 1990s [19] and first was developed in C++ and Java.

In MVP, the View and the Model are neatly alienated and the View interpretation is a contract through which the Presenter access the segment of View that is reliant on the rest ofthe system.

- The Model is the constituent which conserves data, state and business logic; it just rendering a group of service boundary to Presenter and hides the inside details.
- The View is the user interface, it receives user's action and contract to Presenter to achieve user's need, and then theView responds user by result information.
- The Presenter sits in between the View and the Model; itreceives input from the View and passes commands down to the Model. It then gets result and updates the View trough the contracted View interface.

"Fig. 1" illustrates the parts of MVP pattern and how they interact with each other.

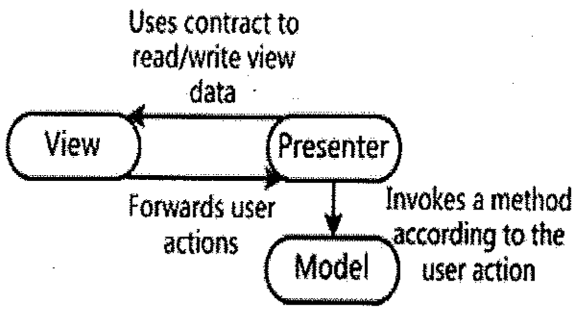

Uses contract to read/write view data
View
Presenter
Forwards user actions
Invokes a method according to the user action
Model

Figure 2. MVP pattern

Contrast to traditional presentation layer, the advantage of presentation layer with MVP pattern is based on tree facts:

- The View doesn't know the Model. Because of this, there is a low coupling between Model and View. It means that if Model or View was changed, another part not needs to modify as long as interfaces are stable. This also stands for the flexibility of architecture and the reusability of business logic in Model.
- The Presenter ignores any UI technology behind theView. According to this, the replacement of UItechnology, such as transfer Windows Forms toWPF or to Web Forms, is not need any change of other parts.
- The View is given an important for testing purposes. Intradition, it is impossible to test View or business logic component before another has completed because of the tight coupling between View and business logic. By the same token, the unit testing for View or business logic component is difficult.

All of those problems are solved by MVP pattern. In MVP, there is no direct dependency between Viewand Model. For that reason, developer could usemock object to inject into View or Model so thatthey can be tested on one's own.

2.6. TABBING SYSTEM USING MODEL VIEW PRESENTER

The View Layer

A front end is developed for demonstrating the capabilities and testing the web services. The view layer uses a split implementation between the server and client. Our view layer isn't only a display but also a framework for adding third-party functionalities. The relatively lightweight JavaScript language is used to deploy the client framework.

On the server side, Django template engine is used, which generates responses usinga template. Within the templates, deployed the Yahoo UI Web framework is used. Yahoo UI is an Ajax framework that provides several Web application components such as DataTable andDialog. When a user requests a particular view,the Django template engine generates the view with Ajax code loaded. Once received by the client,the Ajax code takes over and continues to process the user's requests. The client side is only responsible for user interaction and data display. The RESTful Web services can deliver representations in different formats based on the request headers. When the user clicks on a payment,Ajax application requests the data in JSON, and then prepares a dialog view on the fly. When the user requests payment details by clicking on "View HTML," the Ajax application opens an ew window to /payment/ID, and the returned document is directly viewable as HTML(because a browser request has an HTML accept header).

Using RESTWeb services, authentication occurs using existing technologies, further reducing complexity. HTTP basic authentication over SSL is used. Once authenticated, the actions a client can perform are restricted by login credentials. Of course, this setup has all the drawbacks of HTTP basic authentication.

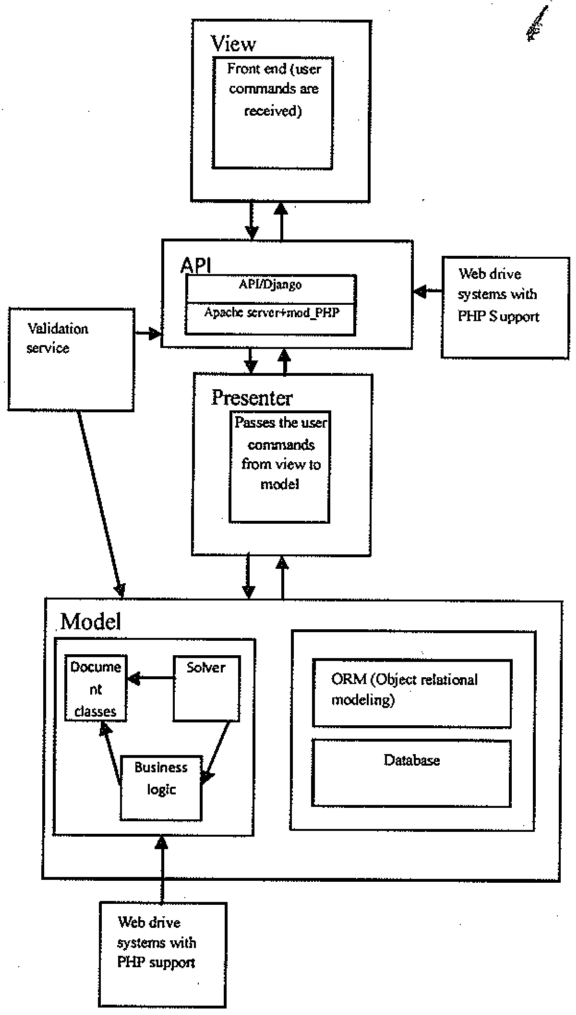

View
Front end (user commands are received)
API
API/Django
Apache server+mod_PHP
Web drive systems with PHP Support
Validation service
Presenter
Passes the user commands from view to model
Model
Document classes
Solver
Business logic
ORM (Object relational modeling)
Database
Web drive systems with PHP support

Figure3: Overview of controlling system using MVP, AJAX

The Presenter layer

Presenter is the core of MVP pattern. Presenter receives user actions and read input data from view, and then it invokes functions in Services such as filing, searching, solving and reporting to complete the business logic and modifies View's state. This entire works are called presentation logic.

The Model Layer

The Model is the component which preserves data, state and business logic; it just exposes a group of service interfaces to Presenter and hides the internal details. For system, there are four business logic: filing, searching, solving, and reporting.

Filing

The first action of filing a record is straight forward. The ORM automatically saves changes on a mapped object, so an insertion simplifies to instancing the correct object:

```
newDocument =
   DocumentCredit({'credit_id':
     0, 'amount': post['amount'],
     'description': post['desc'],...})
```

Saving an object involves many SQL statements and code steps. Interruption while executing these steps can cause database inconsistencies. So, our system uses transactions, which are groups of SQL statements that must be completed atomically:

```
try:
   ins = DocumentPayment(paymentData)
   connection.session.add(ins)
   connection.session.commit()
except Exception, e:
   connection.session.rollback()
   raise e
```

If the process of creating a new document object and saving it succeeds, the changes are committed (written to the database). Otherwise, the transaction is terminated, and the database is immediately rolled back (returned to the statebefore last transaction).

Searching

This is always a challenge, because there are many document attributes on which to filter and sort.

Using ORM, the attributes based search is used for mapping in to an object, and PHP lets us use dictionaries as inputs to easily accommodate a varying number of parameters. The combination of these two results given in a system that can sort and filter along any attribute mapped to any object. The code below demonstrates the search function:

- Function prototype: `def searchDocuments(self, filters=None, limit=None, sortBy=None).`
- Example usage: `searchDocuments({'type': 'credit','user':'test',},(0,10),('amount', 'asc')).`

This query returns a list of credit documents belonging to user "test" sorted by amount in ascending order, but limited to the top 10 records.

Solving

This involves associating ("solving") credits and payments to invoices and debits and is handled by the solver module. Credits and payments have a positive amount (positive documents), while invoice and debits have a negative amount (negativedocuments). The solver first searches for both positive and negative documents using these arch functionality. These are then matched until all amounts are balanced, or until one type of document runs out. With each association, an association object is created with a time stamp to document the processing of a payment or credit.

Reporting

The solver also generates reports. The main reports are the amount left over on a single document and the total amount due. The former can be found by going through a document's association entries and subtracting the total from the amount already matched to other documents (for example, subtracting the total invoice amount from all associated payments). The total amount due is just an extension of this, because the mean amount left on all documents to obtaina user's grand total.

Database

The apparent choice would be a standard SQL database, perhaps with parts of the business logic offloaded into stored procedures. However, this increases system complexity, because developers must be competent in an additional (and to someextent vendor-specific) language (T-SQL versusMySQL).[20]It must also be mindful of the complex interaction between these procedures and the program code to avoid redundant implementation of functionality and to reduce error.

ORM, on the other hand, is designed to bridge the gap between relational database and object oriented programming by combining rows from different tables together to form logical objects.[21]Business logic is a good candidate for object oriented programming, and here full advantage of ORM is considered to reduce our design complexity.

PHP's answer to ORM is SQLAlchemy. Using SQLAlchemy, start the design by defining objects rather than the database structure.Our final design is implemented on MySQL with InnoDBengine (for its support of foreign keys and cascades). The system is based on financial records, such as payments and invoices. Here, debits and invoices are negative documents, while credits and payments are positive documents. As these documents accumulate for a user, they'reperiodically "solved" (negative documents arebalanced against positive documents) to indicatethe user's overall account balance.

Once the objects are designed, transferring them to a relational database is done. The "table per class" strategy is followed,[21,22] using nonbusiness-logic related object-identifier keying.[22]The strategy dictates that any column that might have any business significance. This is advantageous, for example, to include an alphanumerical letter in our invoice ID (perhaps to indicate the office branch that generated the invoice), this wouldn't require many code changes (changeswill be mainly related to rendering) as long as the invoice ID isn't used as a primary key.

Using the "table per class" design, the advantage of ORM reflection is considered—the ability to automatically mirror a database's structure into objects. This allows for rapid modification of objects without additional code input. For example, to implement an additional gift code for credit documents, simply adding a column called "gift_code" into the credit_document table, and ref lection will map the new column as an attribute in our Document Credit class.

III. CONCLUSION

Web-based tab system has handled hundreds of thousands of transactions without error, offering flexibility toour tabbing staff. Deploying technologies such as ORM into the model layer, the powerful object-oriented programming is exploited and combines it with the flexibility, robustness,and scalability of SQL databases. PHP let us quickly develop robust modular applications and simplify code design by using lists and dictionaries as inputs. Finally, using REST and Ajax,the lightweight Web services is developed that third parties can easily interface with, enabling rich,interactive Web front ends.

Reference

[1] Crookes J, *"Multiservice Billing System - a platForm for the future"*, BT Technol JVol 14 No 3 July 1996.

[2] Assimakopoulos Nikitas A., Anastasis N. Riggas & Giorgos K. Kotsimpos, *"A Systemic Approach for an Open Internet Billing System"*,2003,

[3] Thomas Rob, *"Choosing an E-Billing System"*, published in I L T A - December, 2005.

[4] NN Murthy, BM Mehtre, KPR Rao, GSR Ramam, PKB Harigopal, and KS Babu, *"Technologies For E-Commerce: An Overview"*, CMC Center-R&D, CMC Limited Old Mumbai Highway, Gachibowli Hyderabad – 500 019, Andhra Pradesh ,2000.

[5] Bo Yang,Liu Dongsu and Wang Yumin, *"An Anonymity-Revoking E-payment System with a Smart Card"*, springer-verlag, Volume 3, Number 4, 4 December 2001,

[6] EWB Team, *"Electronic Extra Work Billing System: Online Step -By-Step Instructions"*, Revision 2, ISSC, EWB Release 1.1 Instructions, January 12, 2001, http://www,dot,ca,gov/hq/esc/tollbridge/BenMar/006034/MaterialsHandout/EWB,pdf.

[7] Barreto P.S. , G. Amvame-Nze, C.V. Silva, J. S. S. Oliveira, H.P. de Carvalho, H. Abdalla Jr, A.M. Soares, and R. Puttini, *"A Study of Billing Schemes in an Experimental Next Generation Network"*, Springer-Verlag Berlin Heidelberg 2005, http:// www.springerlink.com/content/ nh3n0ebgf7w2h3/

[8] Shiqun Li • GuilinWang • Jianying Zhou • Kefei Chen, *"Fair and Secure Mobile Billing Systems"*, Springer Science+Business Media, LLC, 2008.

[9] Levi Albert and Çetin Kaya Koç *"CONSEPP: Convenient and Secure Electronic Payment Protocol Based on X9.59"*, IEEE Computer Society Press, Los Alamitos, California, March 21, 2009, http://discuss.itacumens.com/ index.php?topic=57564.0.

[10] Antoniou Giannakis, Lynn Batten, Shivaramakrishnan Narayan, and Udaya Parampalli, *"A Privacy Preserving E-payment Scheme"*, Springer-Verlag Berlin Heidelberg 2009, http://www.springerlink.com/content/ h2253738530vm061/.

[11] Fielding R: Architectural styles and the design of network-based software architectures. PhD Thesis. Irvine: University of California; 2000.

[12] Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T: Hypertext Transfer Protocol — HTTP/1.1. Internet RFCs 1999, RFC 2616:1–176.

[13] Berners-Lee T, Fielding R, Masinter L: Uniform Resource Identifier (URI): Generic Syntax. Internet RFCs 2005, RFC 3986:1–61.

[14] G. Lindsrom, *"Programming with Python,"* IT Professional,vol. 7, no. 5, 2005, pp. 10–16.

[15] L. Daly, Next-Generation Web Frameworks in Python,O'Reilly Media, 2007.

[16] K. Smith, *"Simplifying Ajax-Style Web Development,"* Computer, vol. 39, no. 5, 2006, pp. 98–101.

[17] A.Leff and J.T. Rayfield, *"Web Application Development Using the Model/View/Controller Design Pattern,"* Proc. IEEE 5th Int'l Conf. Enterprise Distributed Object Computing (EDOC 01), IEEE CS Press, 2001, pp. 118–127.

[18] D.Parsons, Dynamic Web Application evelopment using XML and Java, Cengage Learning EMEA, 2008.

[19] Mike Potel, *"MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java"*, Taligent Inc. 1996.

[20] R.F. Van Der Lans, Introduction to SQL: Mastering the Relational Database Language, Addison-Wesley Professional, 2006.

[21] C. Russell, "Bridging the Object Relational Divide," ACM Queue, May/June 2008, pp. 16–26.

[22] S.W. Ambler, Mapping Objects to Relational Databases, white paper, AmbySoft, 1997.

Author's Biography

MP.Malar is working as Assistant Professor in the Department of Computer Applications, Kaamadhenu Arts and Science College,Sathyamangalam, Erode, Tamilnadu,India. She has 5 years of experience in teaching. She has published many research articles in the National / International journals. She is currently pursuing doctor of programme in Computer Science at Karpagam Academy of Higher Education, Coimbatore, Tamilnadu, India. Her current area of research interests Web Based Computer Applications.

Dr.X.Agnise Kala Rani is working as Professor in the Department of Computer Applications at Karpagam University, Coimbatore. She received her Ph.D. in 2011 from Mother Teresa University. She holds the Master of Engineering degree from VMKV University and Master of Computer Applications degree from Madras University. She has several publications including scientific journals and top-tier networking conferences to her credit.