An RIA Based Environment for Collaborative Web Application Authoring.

Deven Shah, Manas Apte, Shrikant Goswam, Ajit Padukone and Sahil Shirodkar

Abstract-- **This work addresses a Rich Internet application for collaborative web development in a virtual environment inside a browser, involving multiple users at the same time with an extremely user friendly development interface and customized deployment options. The RIA environment we propose would facilitate highly intelligent code generation enabling even the most naïve user to develop fully functional dynamic web pages, limited only by his creative ability. This environment would also provide for communication between client modules which would provide implementation of the collaborative environment.**

Index Terms—**Rich Internet Applications, Web 2.0, RIA, Web Authoring, Collaboration.**

I. Introduction

In recent years different methodologies and tools have been proposed to support the Web authoring process. Web applications can be modeled by means of different levels of abstraction and their code can be automatically generated from the models, thus simplifying some of the more expensive stages of the development process (i.e., codification, checking, and maintenance) [1][2][4]. If they are used, they are devoted to improve manual modification [3]. Moreover, methodologies and tools have a high learning curve for designers/developers with no experience in solving problems using conceptual models [3]. For making web application development less of an expert's job, the need is not of more advanced editors or powerful web authoring tools; but that of a better mechanism which would guide the user in the process of developing web applications.

Currently existing applications with the same scope as that of our proposed application mainly fall in two categories viz. offline and online tools, both these types aid a user in developing static web pages and sometimes dynamic web pages but for that the user need to have the basic knowledge of scripting languages. Also these tools come with the installation issues in case of offline tools and online tools offer different set of difficulties like delay in the operations, page loading problems. The experience presented in this paper has the goal to carry out an intuitive and highly abstracted approach of developing a web application where the user is guided through the entire development process with an easy to use interface. In the end the user would be able to design and deploy static as well a dynamic data-driven websites without requiring knowledge of DHTML or scripting languages. Our proposal simplifies the development process by enabling the user to translate his requirements using an interactive GUI based on concepts needed to model web applications.

This paper contains the proper analysis and design of the proposed solution divided in the logically separate parts. Section II contains the Introduction to the RIA and how it is useful in web authoring, this section also emphasizes on the features provided by the proposed application along with the assumptions made and the estimated drawbacks. Section IV illustrates the implementation of the proposed solution regarding its overall architecture and containing modules viz. GUI, Code generation, Database, Collaborative modules. Also the website content management is explained in this section. Finally the paper is concluded with the advantages and drawbacks of the proposed application along with the brief mention of future additions.

II. RIA Based Web Authoring.

The user experience in thin-client Web applications is not comparable to desktop interfaces, responsiveness is lower due to network overhead and unnecessary round-trip server access, and disconnected usage is not supported [5]. Rich Internet Applications (RIAs) have been recently proposed as the response to the above mentioned drawbacks [6]. They provide sophisticated interfaces for representing complex processes and data, while minimizing client-server data transfers and moving the interaction and presentation layers from the server to the client. Most of the processing of data can be done at the Client Side itself using a powerful client scripting language. Instead of loading new pages for every change of view, an RIA can modify the components of the present document using DOM along with a markup language. The original data remains on the server, of course, but the RIA can request data in the background, while the user focuses on the application itself. Different types of media are used for data representation, making the user more engaged in the application. With a view of fully exploiting these capabilities, in the proposed application is designed to provide facilities like

Deven Shah is a faculty in Information Technology Dept. at S.P. College of Engg., Mumbai, India (email: devenshahin@yahoo.com).

Manas Apte is a student of final year I T Engg., S.P. College of Engg., Mumbai, India. (Email: manasapte@gmail.com).

Shrikant Goswami is a student of final year I T Engg., S.P. College of Engg., Mumbai, India. (Email: shrikant.goswami@gmail.com).

Ajit Padukone is a student of final year I T Engg., S.P. College of Engg., Mumbai, India. (Email: aj_it_86@yahoo.com).

Sahil Shirodkar is a student of final year I T Engg., S.P. College of Engg., Mumbai, India. (Email: s20sahil@gmail.com).

A. WYSIWYG Environment.

The thick client engine enables browser to reflect the changes in real time. The user is aware of the changes he is making in the design layout as they are reflected immediately, thus giving him the same output as seen during the designing time. This facilitates "what you see is what you get" environment.

B. Drag and Drop Toolbar.

The proposed solution uses drag and drop toolbar for providing a user rich usability in arranging components, alignment, etc of the design; giving it a desktop application feel.

C. Automatic Code Generation.

The code for static HTML as well as validation script for form based pages is generated at the client side engine. Also the script for database connectivity is created at the server side which is then integrated with the client-side code for data-driven web pages.

D. Automatic Database Creation and Normalization.

As per the forms designed by the user the DDL queries are passed to the backend and the database is created. The Queries for data insertion or data retrieval are created as per the user's needs. With the help of adaptive normalization technique the database is normalized initially using some trial data and then on addition of further data records.

E. Collaborative environment for Page Design.

Collaborative Development means working together and co-operating in the process of development. This facility is provided by the two ways viz. Bi-directional communication and real time reflection of design changes in the collaborating modules. The Collaboration feature enables more than one client to participate in the design and editing of the same webpage. The changes made at one client are reflected immediately at the other collaborative client's view.

F. Website Content Management and Publishing.

Even after publishing the website to the internet users, the designer of the web site will still be able to edit contents of the any web page as well as add the new pages to the existing site and again publish it with the updated version.

Some of the assumptions made while designing the proposed system is the user uses fast internet connection with RIA technology compatible browser. The user has enough free disk space on his workstation so as to host a client engine in the browser which has to be transferred from the server side in order to load the application. Also the User should not hit the browser refresh or reload button which will result in restart of the entire application which might cause loss of unsaved work.

III. Implementation

Fig. 1 illustrates the overall Architecture and Implementation of the System.
The system has the traditional 3-tier architecture for web applications. It consists of a Client Module, a Server module and a database tier. In this case the client module is a "thick client" that is along with the Presentation and Interface functions it also consists a major amount of processing, business logic as well as a temporary data store in which the data that is currently required is brought in from the server. The Client module is based on the Model-View-Controller architecture. That is within this Module which is actually a "View" component of the overall system, there are 3 separate tiers. The data that is currently required for processing is brought in to the client and stored in XML data Islands, which form the temporary data tier of the client module. These data structures also provide for persistent data because whenever changes are to be saved then the required data is sent to the server database.

The Client module also called the "Client Engine" consists of the following.

i. A GUI module which handles the display view and also the User interaction events. It uses Markup Languages for display and DOM structure display modification.

ii. Controller which consists of business logic using client scripting languages. This controller handles the events of user interaction and carries out the required processing. Any data it requires is stored in the data islands and if not present then is brought in from the server.

iii. Data Islands which store the data brought in from the server as well as data that is created dynamically and is required to be stored.

iv. Server Communication Module which handles all server communication. Most of the server requests are done in the background without the user's knowledge. Also these requests may be Asynchronous which allows for Application processing to continue without waiting for server response. The data transfer between the client and server is done in form of XML data. HTTP or HTTP(S) channels are used for the transfers.

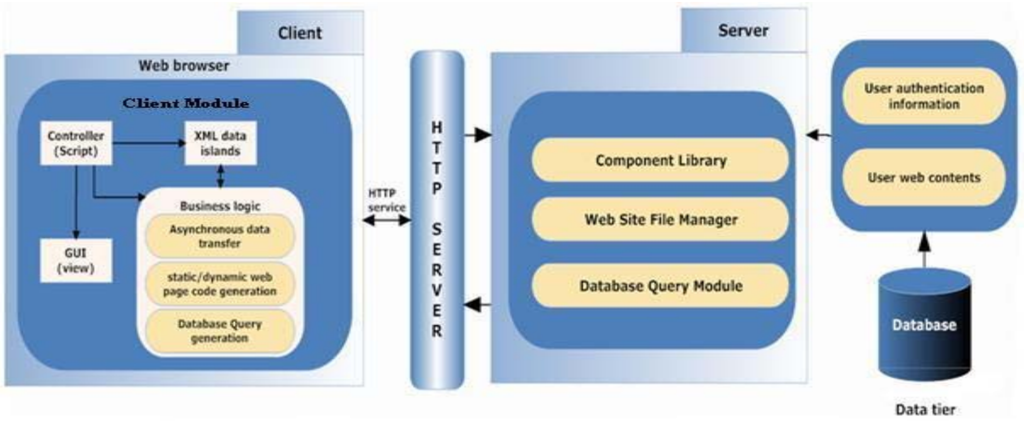

Fig.1. Overall System Architecture

The Data Islands store data that are required like current User's data, user's web site related data, or data related to the design of the currently edited web page. The controller module is responsible for the functions like code generation, generation of Queries for database related functions, as well as handling user interaction like drag-and drop actions, navigation between view states, and requesting data from server of required. A normal sequence of interaction between User and the module would be as shown in Fig.2.

The interactions enclosed in the white rectangle in Fig.2. represent the generalized form of the user interactions with the client module. The User interacts with the GUI which gives the controller information about the actions. Based on this the controller handles the event and may modify the data in the data store, update the view of the interface and may optionally save the current state of data to the server. All communication with the server is done through the Server Sync component which acts as an interface between the Client module and Server.

The implementation of the proposed functionalities also follows the basic interaction shown in Fig.2.

A. GUI Module

The GUI module is responsible for providing the interface and a "Design Canvas" to the user.

The module basically consists of different "view states" which are technically different "page views" that are downloaded from the server all at once or in the background without the user's knowledge. As the user navigates between the various features, the display switches between corresponding view states. Each view state will contain components which are either static, and do not change, or dynamic, which are added, modified or removed from view as per users actions. Static Components include tool bars, menus, layout and style components etc. Dynamic components include components which the user adds to his page design. These components have to be created dynamically and added to the view and modified or deleted.

Thus the user actions have to be monitored and the the events are handled in various ways.

The User may click on a button or any component, he may "drag" a component around in his web page design or he may edit some components properties. These events have to be identified along with the component that the user is acting on and then the view state is modified dynamically.

Fig.3. shows the components of the GUI module. It includes Controller Components like the Event Listeners, Event Handlers and Display Manager. There are various user actions which are handled in different ways. They may or may not require data from the data store. The data store has the data related to the GUI which is used by the event handlers for updating the display view.

Fig.4. shows the various types of user action

When the user drags any control icon from the tool bar to the design canvas, this event is handled by the drag manager, which identifies the type of component to be added to the design as well as the location. The properties of the component along with their default values are retrieved from the data store which contains all properties associated with each type of component and the default value.

The display manager then takes over and using the specifications creates the desired component dynamically and adds it to the view.

The user may also drag a component to some other location. This is again handled by the Drag Manager which identifies the location and informs the Display manager to update the view accordingly.

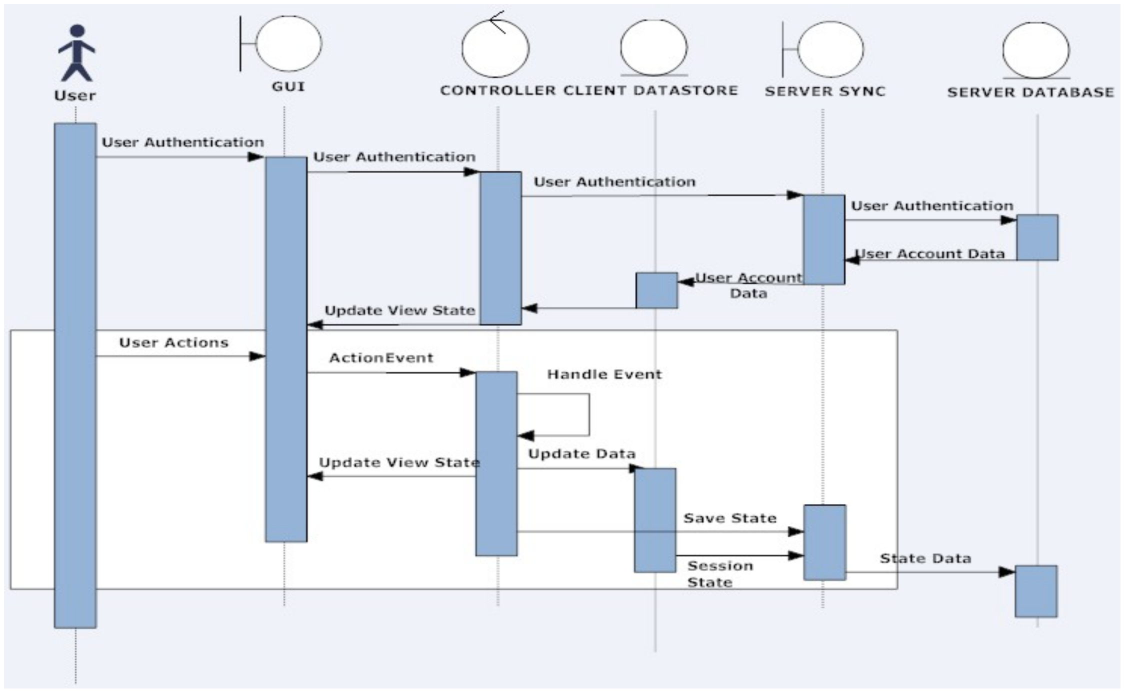

Fig.2. User Interaction Sequence.

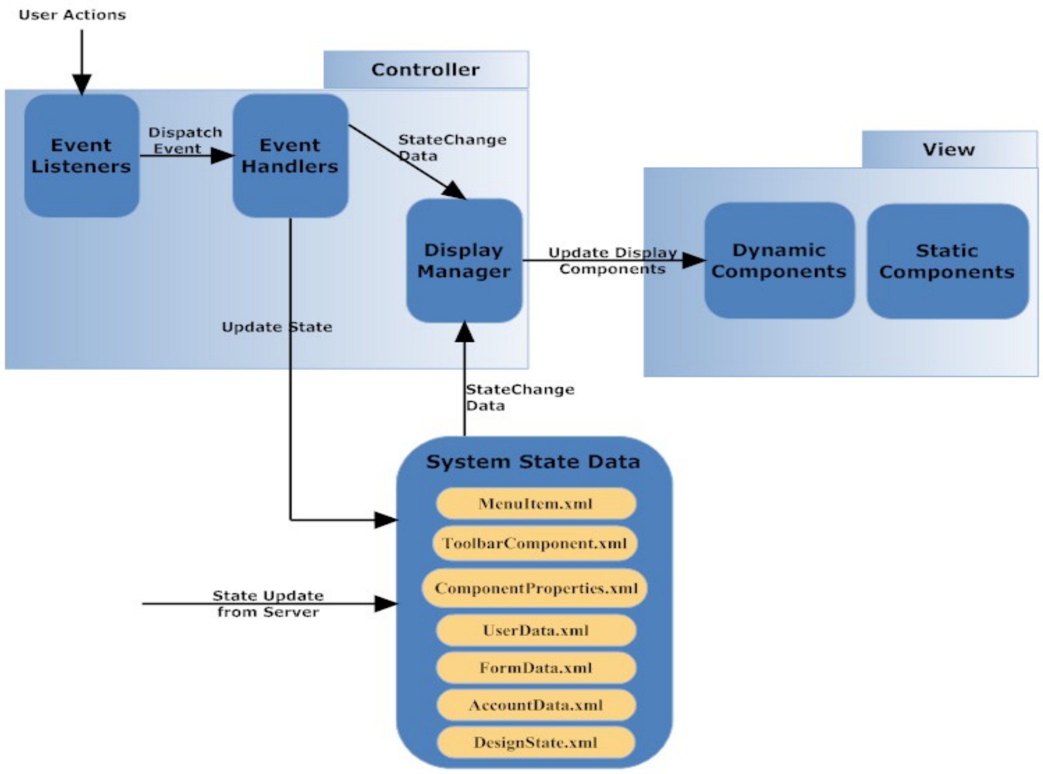

Fig.3. Components of GUI Module.

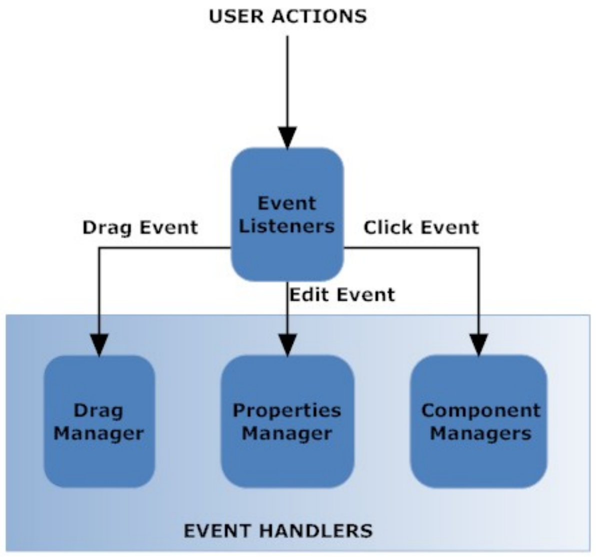

Fig.4. Types of Events

Whenever the user selects any component to edit with a mouse click, the event is identified and then the properties and their current values are retrieved from the data store which stores the state of the design that is all the currently added components and their property values. These values are returned to the display manager which updates the view state and displays them to the user. The user may then edit the properties. Again this is handled by updating the design state as well as changing the properties of the component in the display.

The user may click a button to navigate from one view state to other. This is handled by the Controller which identifies the view state to be activated and informs the Display Manager to load it.

Thus the various actions of the user are handled by the client module using client scripting requiring minimum server interaction for page refresh.

B. *Code Generation Module.*

The static HTML code for the user workspace design is generated using the concept of XML data islands. Data islands are XML elements embedded inside the page source which can be treated as DOM objects and can be suitably accessed and modified. The GUI module provides a toolbar comprising different components, layouts and templates for user design. The designing of the Web page requires only dragging and dropping of these components onto the work area called design canvas. Whenever a particular item from the toolbar is dropped on the work-area. the corresponding XML data island is appended with an HTML tag or a set of nested HTML tags corresponding to that item, nesting constraints are taken care of by the code generation module through the use of event handlers which fetch the current target where that item is dropped (this facilitates addition of a child tag under proper parent tag in the XML data island).
Fig.5. shows components of the Code Generation module.

In Fig.5. The Controller components are shown which monitor and handle the user actions. The Event handlers know which component is to be added or what modification is to be done on the components. They get the information from the GUI module. The generation of HTML code is done by the event handlers. They update the “Design State” data store; which stores the code of the current page design, which in turn leads to change in the view state. The server sync module reads the Design State and saves the state to the server. Sometimes in collaborative environment the server sync receives an update to the Design State from the Server. The Server sync updates the design state according to this information. This facilitates the exchange of “design state” between collaborating client modules.

C. *Database Design Module.*

1) *Database Creation and Connectivity.*

Another important aspect of code generation is, the creation of database entities at the data-base server corresponding to data acquisition forms created by the user. The items dropped into the form container are stored in an array, which is then used to generate and fire corresponding SQL query at the database server. Support for the data retrieval is provided by the query generator module. This provides intelligence for getting the entities in the database on which the query is to be fired and building a query on the fly depending upon the particular constraints on the data to be displayed in the form. The query building functionality is provided by recursively asking for constraints in FROM WHERE and SELECT clause of standard SQL. This query forms the basis for the generation of server side code for the same web page. For data acquisition pages the query is simply fired in the script code embedded in HTML, whereas in case of data retrieval pages, first of all a connection is established with the database by creating a connection object, then a standard recordset object is linked to this connection and finally the query is fired using this recordset.

The Components of the Database Creation Module are as shown in Fig.6 on next page.

The GUI module provides the information about the type of component added as well as their properties which helps the Query builder to generate the corresponding DDL Query. Also the GUI provides the user’s parameters in the Form Design properties which enable the Query builder to build the corresponding DML Query. These Queries are sent to the server and integrated with the code of the web page. The DDL queries are used to create the database schema for the user’s website. The Database Connectivity code is generated at the server and integrated with the code generated at the client side. The code created at client side contains only the server script code for the binding of form fields and tables, but does not contain the code for connecting to the database. This is because the connection string to be generated depends on the type of database system used in the back end.

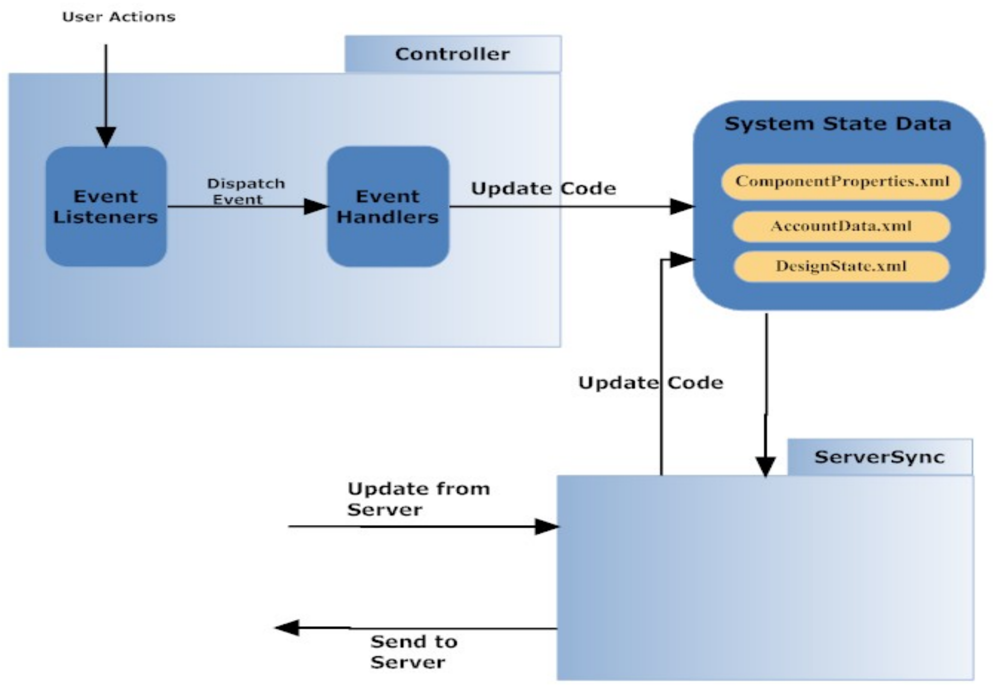

Fig.5. Components of Code Generation Module

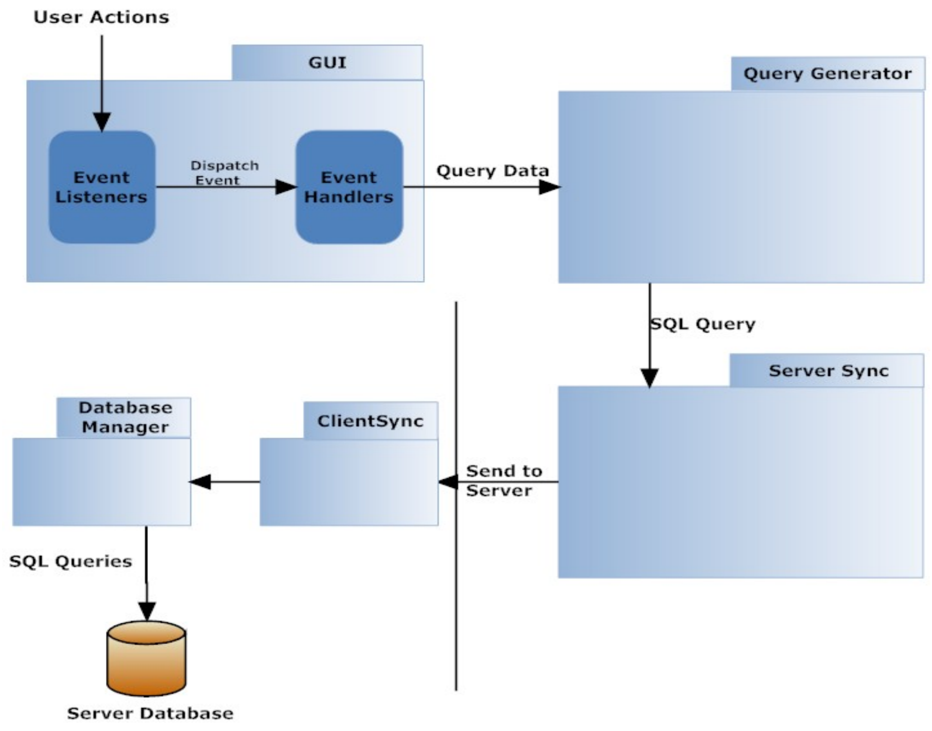

Fig.6. Database Module

2) Database Normalization.

Normalization helps to minimize the duplication of data and safeguard against Data anomalies. Normalization of the database is done by finding out the functional dependencies between various columns of the table. The database created as a result of the user designed forms will be by default in First Normal Form as all the fields will be atomic and meant to store only atomic data. The user will be asked to define a primary key for every table generated or if the user is a naïve user who doesn't have any idea of the primary key a primary key will be generated by the system. Higher normal forms can be achieved by monitoring the data entered in the table for the dependencies. Initially we will have to assume maximum dependencies. A dependency can be taken as nonexistent only when we have a valid record which violates it. We will monitor all dependencies validity after the addition of some number of records. This number will be decided depending on the size of the table to be normalized. For tables with few columns, a few records are sufficient, while for larger tables, more test data is required. The user will be initially asked to enter the test data for the schema just created. Based on the analysis of the test data an initial number of dependencies will be proven false. On further analysis of the data when more records have been added, more dependencies will validated or invalidated. Thus we will be able to adaptively normalize the data according to the records currently entered in the table.

The Server module for database management will maintain information about all tables that are currently created. This information will be used along with our algorithm which will help to achieve Normalization. Based on the results the tables will be spitted or combined and the information will be updated. This algorithm to normalize a schema is still under work.

D. Collaborative Design Modules.

Collaborative Development means working together and co-operating in the process of development. Collaboration can be between various stakeholders. The following scenarios may exist where a collaborative environment will prove extremely useful.

1) Expert and Beginner Designer Collaboration.

A beginner with no knowledge of visual components and web designing process may want to develop a web site for personal or business use. In this development process a mentor would help the user to convert the design concepts into the actual page layout by giving suggestions in the collaborative environment. The mentor would guide the naïve user in the process of creating his/her page with all the desired functionalities, by interacting with the user in the realms of a virtual environment where the mentor can suggest the usage of appropriate components at appropriate places to design his page in suitable . It also enables an experienced developer to write customizable codes for a naïve user who can thereby have all the desired functionalities.

2) Client and Designer Collaboration.

A client who wants a web application developed may collaborate with the actual designer in this environment. Whatever changes designer brings about are reflected immediately at the Clients view. This will help him to monitor the design process and give instant feedback and suggestions for the design. This would enable the designer to meet all design requirements of the client with minimal physical communication. Most of the ideas can be ex-changed through the environment easily.

3) Co-designer Collaboration.

In case of a webpage development where two co-designers are working on the designing of the same web page simultaneously may need to communicate in order to introduce parallelism in the designing process thus reducing the development time. Collaborative environment would help this scenario by allowing simultaneous changes in the mutually exclusive parts of the same web page at the same time exchange of ideas is also encouraged.

4) UI Expert and Backend Designer Collaboration.

Consider a case of a specialized designing team, there has to be collaboration between the UI designer and the backend designer to optimize the design. In such a case the interaction between the backend designer and the lay out designer is facilitated in which the UI design could be monitored by the back end developer so that the interactions with the database are efficient.

During the designing process when the two users collaborate and if one user edits or changes any design component on his work space the change is reflected immediately on the interface of the other client module. This is done by sending the changes that are done by one user to the other client module through a common server, this updating can be done at uniform intervals of time or can be done whenever a change is made in the design by one user. This collaboration can be enabled by a module which maintains a "state" of the web page design that is being edited currently. The "state" can include a list of elements that have been currently added like for e.g. An image as well as the values of all its properties such as positioning, layout, etc. This is maintained using XML data islands at the client side.

This XML data can be processed using client side scripting. Whenever one user makes a change in the GUI, the "state" is updated according to the change. And vice versa whenever there is a change in the "state", the change is reflected in the GUI or the "view" of the web page being edited. Thus to implement a collaborative environment where the changes made at one client module are reflected almost immediately at the other module all that is needed is some way to communicate the change of "state" at one client to the other.

This is done through the server. The change is first sent to the server which in turn does the job of forwarding the change and updating the "state" at the other client module. The same kind of technique can be used to implement a Bi-Directional "chat" functionality within the application which would facilitate instant verbal communication. Fig.7. shows a series of interactions and messages passed between the client and server modules during collaboration.

E. Website Content Management and Publishing

It is used to manage and control a large, dynamic collection of web material (documents and their associated components). It facilitates document control, auditing, editing user web pages and organizes information into a set of folders. Standard visual templates can also be automatically applied to new and existing content. When a user logs in, all data related to his website created earlier like database, tables, webpage information etc. is first downloaded and stored at client side. When a user selects any particular web page, the code for that page is downloaded and stored into DesignState which causes it to be reflected on the GUI for editing which helps the user to edit his web page and the code for that page gets modified accordingly.

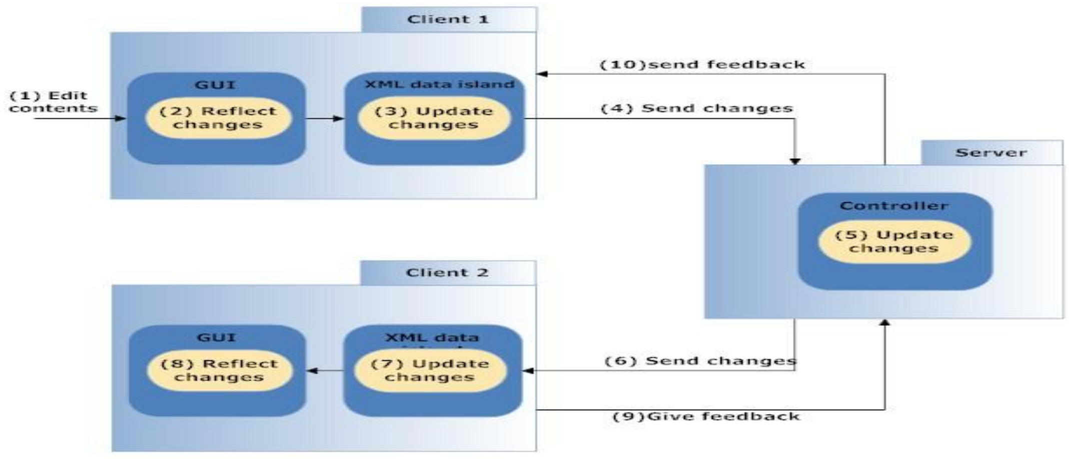

Fig.7. Collaboration Module

The user can also add new/remove pages and is given a facility to manage entire navigation structure of his website by editing the hyperlinks.

When the user finished creating his website and wants to deploy it online, he just needs to click one button "Publish". When he does so the server side will automatically create the files of his web pages as well as folders of the images, JavaScript codes etc in his pages. This folder will be hosted on our website by creating a sub domain under our URL. Thus the user will have successfully created, designed and deployed a website.

IV. Conclusion

In this paper we have discussed how our tool facilitates creation of static and database-driven web applications with an extremely simple and intuitive user interface. Even in the presence of traditional web authoring tools, it has been found that a naïve user unfamiliar with web markup and scripting languages and database is not able to develop web applications of his own. In our proposed solution he is aided in the development process by a very easy to use interface along with features like automatic code generation and database design. The collaborative environment aids a system user in the process of web page development where the
development takes place between the two users as mentioned in the scenarios. We have in this paper described how a Rich Internet Application providing following features can be designed and deployed

- Rich WYSIWYG Environment.
- Drag and Drop facility.
- Automatic Code Generation.
- Automatic Database Creation and Normalization.
- Collaborative environment for Page Design.
- Website Content Management and Publishing.
- Ability to work offline.
- Immediate reflection of changes made.
- Website Preview and Deployment.

While some problems do exist with our application like most RIA applications which take more time for loading into the browser, also fully offline working ability is not supported as new data has to be transferred periodically depending upon the users tools demand.

The limitations of our application suggest a natural direction for future work which includes enhanced performance, adaptive normalization of database and simultaneous editing during collaboration. Moreover, we plan to incorporate a feature in which any webpage on the Internet can be imported into the environment, edited as per the users wishes and saved. This would facilitate a universal collaborative environment over the Web where personalization, collaboration and on-the-fly browsing/editing are possible.

We believe that our proposal can be considered an innovative step in the direction of this goal, providing a rich environment where a user, can create, modify and customize any type of web page. As required for a complete content management, the application is completely independent form any data format, editing tool and user skills.

V. References

[1] A. Knapp, N. Koch, F. Moser and G. Zhang, "ArgoUWE: A Case Tool for Web Applications", Int. Workshop on Engineering Methods to Support Information Systems Evolution, Geneva (Switzerland), Sept. 2003.
[2] C. Barry and; M. Lang, "A survey of multimedia and Web development techniques and methodology usage", IEEE Multimedia June 2001, vol. 8, issue 2, - pp. 52-60.
[3] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai and M. Matera, Designing Data-Intensive Web Applications, Elsevier, Amsterdam (Netherlands), 2002.
[4] J. Gómez, C. Cachero, and O. Pastor, "Extending a Conceptual Modelling Approach to Web Application Design", Int. Conf. on Advanced Information Systems, Stockholm (Sweden), June 2000, LNCS 1789, pp. 79-93.
[5] M. Driver, R. Valdes, and G. Phifer. Rich Internet Applications Are the Next Evolution of the Web. Technical report, Gartner, May 2005.
[6] J. Duhl. White paper: Rich Internet Applications. Technical report, IDC, November 2003

VI. Biographies

Deven Shah is professor in IT Dept; S.P.College of engineering, Mumbai. He is currently pursuing PhD from NIT, Surat.
(E-mail: devenshahin@yahoo.com)

Manas Apte is student of Fourth Year Information Technology Engg. at S.P.College of engineering, Mumbai.
(E-mail: manasapte@gmail.com)

Shrikant Goswami is student of Fourth Year Information Technology Engg. at S.P.College of engineering, Mumbai.
(E-mail: shrikant.goswami@gmail.com)

Sahil Shirodkar is a student of Fourth Year Information Technology Engineering at S.P College of Engineering, Mumbai. (Email: s20sahil@gmail.com)

Ajit Padukone is student of Fourth Year Information Technology Engg. at S.P.College of engineering, Mumbai.
(E-mail: aj.it.engg@gmail.com)