An AJAX Tool for Online Modeling of Model 2 Web Applications

Relatore: Prof. Marco Brambilla

Tesi di laurea di: Alessandro Origgi
matr. 682197
Sommario

Le moderne applicazioni Web sono caratterizzate da un elevato livello di complessità e trattano grandi quantità di dati. Quando l'applicazione cresce in complessità, lo sviluppo di codice manuale non è adatto, perché può causare problemi in termini di efficienza, di riusabilità, di affidabilità e di manutenibilità.

In questo progetto proponiamo una soluzione progettuale light-weight per lo sviluppo di applicazioni MVC. Il progetto offre un semplice modello concettuale, che consente agli sviluppatori di modellare l'applicazione Web, e un editor tool visuale, chiamato MVC-Webflow, per semplificare il design dei modelli delle applicazioni MVC da sviluppare. Lo strumento fornisce anche una parziale generazione automatica del codice, che viene effettuata dal tool on the fly. L'aspetto innovativo del progetto è la costruzione, non solo di un semplice tool visuale, ma di uno strumento fruibile on-line da qualsiasi browser Web.

Il lavoro si è sviluppato a partire da una analisi dei requisiti per passare poi al design e alla scelta delle tecnologie da utilizzare per lo sviluppo. L'editor MVC-WebFlow è stato sviluppato usando le tecnologie Web Java e Ajax (per la realizzazione di applicazioni dinamiche e interattive).

Il nostro progetto ci ha consentito di provare un nuovo approccio per supportare lo sviluppatore durante la progettazione di applicazioni Web. I vantaggi di questo approccio sono l'utilizzo del ben noto paradigma MVC, il solido fondamento su modelli di ingegneria per il Web, e la riduzione della durata del ciclo di sviluppo.
Abstract

Modern Web applications are characterized by a high level of complexity and deal with huge amounts of data. When the application grows in complexity, manual code development is not suitable, because it lacks in efficiency, reuse, reliability, maintainability, and group work facilities.

In this project we propose a light-weight design methodology that leads to the development of MVC applications. The project offers a simple conceptual model, which allows developers to model the application, and an on-line visual editing tool, called MVC-Webflow, for the specification of these simple conceptual models for MVC applications. The tool provides also a partial automatic code generation, that can be performed on the flight directly on the deployed application. The innovative aspect of the project is the construction of, not just a simple visual tool, but a tool available online from any Web browser.

The work has developed starting from a requirements analysis, the application design and the selection of the technologies to be used for the development. The MVC-WebFlow tool has been developed using Java and Ajax Web technologies (to realize dynamic and interactive applications).

Our project has allowed us to try a new approach to support the developer during the design of Web applications. The advantages of the approach are the closeness to the well known MVC paradigm, the foundation on solid web engineering models, and the reduction of the duration of development cycle.
INDEX

1. Introduction and problem presentation, 6
2. Rich Internet Applications & Ajax, 7
   2.1. Rich Internet Applications, 7
   2.2. Ajax Web applications, 8
      2.2.1. What is Ajax, 8
      2.2.2. Characteristics of Ajax Applications, 9
      2.2.3. Ajax technologies, 10
      2.2.4. Ajax versus the Traditional approach, 14
      2.2.5. Alternatives to Ajax, 16
3. Frameworks and graphic libraries for Ajax, 18
   3.1. Frameworks, Toolkits and Libraries introduction, 18
   3.2. Frameworks, Toolkits and Libraries for the Online Editor, 20
      3.2.1. OpenJacob Draw2D library, 20
      3.2.2. Yahoo! User Interface library (YUI), 26
4. MVC-WebFlow model, 32
   4.1. Introduction, 32
   4.2. Aims and benefits, 32
   4.3. Jakarta Struts overview, 34
   4.4. Definition of the conceptual model, 38
   4.5. Object Oriented view of the model, 41
5. MVC-WebFlow Ajax tool, 45
   5.1. Objectives and context, 45
   5.2. Business requirements and functionalities, 47
   5.3. Users groups and site-views, 48
   5.4. User requirements, 48
      5.4.1. Functional requirements and use case view, 48
5.4.2. Non-functional requirements, 54
5.5. Interface requirements, 55
5.6. Software requirements and system architecture, 56
5.7. Data schema design, 59
5.8. Logical view and components view, 60
  5.8.1. Logical view, 60
  5.8.2. Components view, 63
5.9. Tool Interface and sample application, 64
6. Conclusions and future development, 69
7. Appendix A: Ajax open source and free solutions, 71
  7.1. JavaScript Multipurpose frameworks, 71
  7.2. JavaScript Remoting frameworks, 79
  7.3. JavaScript Graphics and Effects frameworks, 81
  7.4. JavaScript Flash frameworks, 88
  7.5. Javascript Logging frameworks, 92
  7.6. Javascript XML frameworks, 93
8. Bibliography, 96
1. Introduction and problem presentation

The World Wide Web has evolved a lot since its initial creation. While the first sites were mainly based on textual (and hypertextual) information, the new trends move towards complex enterprise-wide applications and multimedia contents. In today’s World Wide Web, you can create a great variety of interactive multimedia presentations and powerful applications, such as Webmail, e-commerce, Web-forum, blog, MMORPG, and others. Modern Web applications are characterized by a high level of complexity and deal with huge amounts of data. When the application grows in complexity, manual code development is not suitable, because it lacks in efficiency, reuse, reliability, maintainability, and group work facilities. On the other hand, several Web engineering approaches are too far away from the average developer and designer way of working to be widely adopted. In this project we propose a light-weight design methodology that leads to the development of MVC applications. We present an on-line visual editing tool called MVC-Webflow for the specification of simple conceptual models for MVC applications and we provide partial automatic code generation, that can be performed on the flight directly on the designed application. The advantages of the approach are the closeness to the well known MVC paradigm, the foundation on solid web engineering models, and the reduction of the duration of development cycle. This tool will be something new because it will allow the creation of the application model directly from the Web browser. To do this we have decided to use the most innovative technologies for the Web, the Ajax technologies. Before entering in depths of the model description, benefits and features, in the following sections, we will present an overview of the potentials of Ajax, what can be done and the comparison with the traditional approach of developing Web application. In particular we have made a deep research among Ajax frameworks and graphic libraries to identify the solutions useful for our project and specifically for the dynamic creation of diagrams inside the browser.

Document organization

Chapter 2: introduction about Rich Internet Applications (RIA) and Ajax technologies

Chapter 3: overview of Ajax development frameworks and presentation of the two main solutions used for the project, OpenJacob Draw2D and Yahoo! User Interface (YUI)

Chapter 4: focus on the MVC-WebFlow conceptual model and its benefits

Chapter 5: project specification of the MVC-WebFlow Ajax editor and brief overview of the tool interface with example of modelling process

Chapter 6: conclusions and future works on the model and the tool

Chapter 7: appendix A with the depth research among Ajax frameworks
2. Rich Internet Applications & Ajax

2.1. Rich Internet Applications

Today, inside the Web, you can find any type of Web application. Although these applications are powerful, they aren't without their drawbacks. Most are incredibly clumsy when compared to their native application counterparts (Hotmail versus Outlook Express, for instance), and many more have massive usability problems. However, problems or not, the Web and especially Web applications is one of the fastest growing and most important fields of software development. Internet applications bring huge benefits to the table when compared to a normal application. They are highly accessible, require no installation, can be upgraded at any time, and offer access to large amounts of data without complex networks. These advantages allow for a shorter time to market, as well as lower development and support costs, when compared to a native application. Even though Internet applications usually have poorer usability due to their simpler, less interactive interfaces and slow update times, they are replacing native applications everywhere you look. A Rich Internet Application (RIA) is an Internet application that attempts to bridge the usability gap between native applications and normal Internet ones. It contains more code on the browser, which offers higher levels of interactivity and an experience similar to native applications. With RIAs, it’s possible to use many technologies, such as Flash, Java, and ActiveX, but the most important one is JavaScript. Because JavaScript is provided directly by the browser instead of being an add-on like the other technologies, you can get the most benefit from the least amount of work. One of the driving technologies behind RIA in the JavaScript language is a technology called AJAX. AJAX offers the ability to communicate with your Web server outside of the normal load flow. It will always be the user who will feel the effect of the technology you choose, and the first priority of any Web or desktop application developer should be the user experience. Users are not interested in what technology is being used or whether the application is a traditional desktop application or a Web application. Users demand a feature-rich and interactive interface. Traditionally, desktop applications have been able to provide users with the richness required to fulfill their demands, but an increasing number of desktop applications are migrating to the Web. Therefore, Web application developers have to provide richer Web interfaces.
2.2. Ajax Web Applications

2.2.1. What is Ajax

Ajax has been minted as a term describing a Web development technique for creating richer and user-friendlier Web applications. In this chapter, we will give you an overview of Ajax. One of the reasons Ajax has gained a great popularity is the XMLHttpRequest object and the way this object makes it possible for developers to asynchronously communicate with underlying servers and any business services used by Web applications. Popular sites such as Google GMail use Ajax techniques to provide users with rich interfaces that have increased the awareness of Ajax. Although the name Ajax is new, the technologies listed as the foundation of this technique (JavaScript, XMLHttpRequest, and the DOM) have been around for some time. In fact, the latest addition to this suite of technologies, the XMLHttpRequest object, was introduced by Microsoft in 1999 with the release of Internet Explorer 5.0 and was implemented as an ActiveX component. The XMLHttpRequest object, although widely used, is not a standard; it could at best be called a “de facto” standard, since most modern browsers, including Firefox, Internet Explorer, Opera, and Safari, support it. With the XMLHttpRequest object, developers can now send requests to the Web server to retrieve specific data and use JavaScript to process the response. This ability to send data between the client and the Web server reduces the bandwidth to a minimum and saves time on the server since most of the processing to update the user interfaces takes place on the client using JavaScript. The term AJAX refers to a loose grouping of technologies that are used to create dynamic, interactive web content. The term AJAX is an acronym for “Asynchronous JavaScript and XML”. It’s simply describing a technique that uses JavaScript to refresh a page’s contents from a web server without having to reload the entire page. This is different from the traditional method of updating web pages, which requires the browser to refresh the entire page in order to display any changes to the content. The increasing availability of the XMLHttpRequest class in browsers, the coining of the catchy term AJAX, and the advent of a number of high-profile examples such as Google Maps have allowed these kinds of highly interactive web applications to begin to gain traction in the development world. As the term AJAX has become more widespread, its definition has expanded to refer more generally to browser-based applications that behave much more dynamically than old-school web apps. This new crop of AJAX web applications make more extensive use of interaction techniques like edit-in-place text, drag and drop, and CSS animations or transitions to effect changes within the user interface. AJAX can be a great solution for many web development projects, it can empower web apps to step up and take over a lot of the ground that previously was occupied almost exclusively by desktop applications. AJAX can improve the interactivity and speed of your app, ultimately making that application easier, more fun, and more intuitive to use. Often, AJAX applications are described as being “like a desktop application in the browser.” AJAX web apps are significantly more responsive than traditional, old-fashioned web applications, and they can provide levels of interactivity similar to those of desktop applications.
2.2.2. Characteristics of Ajax Applications

Earlier on, Ajax was defined as a technology that "builds on standard web technologies to deliver a rich, responsive, user experience". The characteristics here are intended as a general guide, but not hard-and-fast rules, for what constitutes an Ajax application.

Applications, Not Just Web Sites

These days, you'll hear a lot more about "web applications" or "webapps" than about "web sites." Driving many modern web projects is the perspective of the browser as a platform and the Web as an operating system. Ajax can really be applied to anything running inside a browser and it helps keeps users engaged and productive.

Smooth, Continuous Interaction

Traditional web sites make you submit a form, wait a few seconds, watch the page redraw, and then start the whole cycle again. That's because the tiniest server interaction, and even the tiniest display change, requires a call to the server, and then a complete page refresh. Ajax changes the model in a few ways. First, JavaScript running inside the browser can manipulate the display directly you don't have to send a whole new page from the server in order to hide an element or rearrange the page. Second, server interaction can be handled via JavaScript, so you can upload user commands and download new information without any page refresh. Third, user actions such as mouse clicking and typing can be handled by JavaScript, so the interaction is a lot richer than just filling in a form and hitting Submit. All of these enhancements make Ajax interaction feel faster and more continuous.

Live
With browser-server Ajax interaction it's possible to continuously poll the server for new information. Thus, an Ajax App can be programmed to always show the latest news, details on who else is online, or to send messages to the user. The content is "live".

**Supportive**

Ajax Apps can monitor user actions and proactively support whatever task the user's working on. Actions as small as a single keystroke can lead to a server call, where the server's knowledge and processing power can be harnessed to produce useful content a subsecond later. For example, a form might change according to the user's input, or an error message might appear as soon as a value is typed.

**Visual Effects**

Ajax Apps look similar to conventional web apps, but do tend to include a little more animation. Not the kind of flashy animation that's just there for the sake of it, but animation that gets across a message about what's happening and what the user can do next. For example, a deleted icon might slowly shrink and disappear.

**New Widgets**

Ajax widgets go beyond the standard HTML controls text fields, selectors, buttons, and so on. Widgets such as sliders and progress indicators, built on standard HTML elements, are becoming popular. For example in place of a textarea element, you might see a rich text editor with "What-You-See-Is-What-You-Get" qualities similar to most word processors.

**New Styles of Interaction**

It's not only widgets that are getting an upgrade, but styles of interaction. Here, too, Ajax developers have been borrowing from concepts in traditional desktop environments. Drag-and-drop, for example, has been a staple feature of windowing environments for two decades, but somehow didn't make it onto the Web for a long, long time. Now we're seeing it more and more, and it all makes so much sense, as the earlier portal example illustrated. Other styles of interaction are also becoming more popular. Keyboard shortcuts are being used to streamline activity. And some developers are being a bit more adventurous with mouse buttons, experimenting with double-clicking as well as the right and middle buttons.

**Standards-Based**

Ajax applications require nothing more than a standard web browser from the past few years, such as Internet Explorer 6 or Firefox 1.0. Ajax makes the most of standard browser features and avoids browser-specific features and plugins where possible. This means more than standard technologies; it's also about respecting standard user-interface conventions of the Web.

2.2.3. Ajax technologies

The technologies that are used to build AJAX web applications encompass a number of different programming domains, so AJAX development is neither as straightforward as regular applications
development, nor as easy as old-school web development. On the other hand, the fact that AJAX development embraces so many different technologies makes it a lot more interesting. Here’s a brief listing of the technologies that work together to make an AJAX web application:

- **HTML/XHTML**
- **HTTP, CGI, Form Submission**
- **XML**
- **the W3C DOM**
- **CSS**
- **XMLHttpRequest**
- **JavaScript**
- **Server-Side Scripting**
- **Event model (“DHTML”)**
- **Web Remoting**

**HTML/XHTML**
HTML provides the structure of a web page. An Ajax App uses an HTML document to show the initial page, and the document is continuously manipulated to change the display and set up new events. Where possible, its XML-compliant variant, XHTML, should be used in order to make manipulation more robust.

**HTTP, CGI, Form Submission**
As with conventional web applications, Ajax communicates via HTTP. The difference is that instead of returning full pages, the server returns concise results that are then processed in the browser script. Form submission often with CGI-style URLs is also used, but again is initiated programmatically, meaning that no page refresh need take place.

**Data Exchange and Markup: XML**
XML is where AJAX gets its letter “X”. XML often serves as the main data format used in the asynchronous HTTP requests that communicate between the browser and the server in an AJAX application. This role plays to XML’s strengths as a neutral and fairly simple data exchange format, and also means that it’s relatively easy to reuse or reformat content if the need arises. There are, of course, numerous other ways to format your data for easy exchange between the browser and the server but XML is one of the most common.

**XML as Markup**
The web pages in AJAX applications consist of XHTML markup, which is actually just a flavor of XML. XHTML, as the successor to HTML, is very similar to it. It’s easily picked up by any
developer who’s familiar with old-school HTML, yet it boasts all the benefits of valid XML. There are numerous advantages to using XHTML:

- It offers lots of standard tools and script libraries for viewing, editing, and validating XML.
- It’s forward-compatible with newer, XML-compatible browsers.
- It works with either the HTML Document Object Model (DOM) or the XML DOM.
- It’s more easily repurposed for viewing in non-browser agents.

You can serve XHTML to the browser with a Content-Type of text/html, as all the mainstream browsers render correctly all XHTML documents served as text/html. Although browsers will treat your code as plain old HTML, other programs can still interpret it as XML.

**W3C Document Object Model**

The Document Object Model (DOM) is an object-oriented representation of XML and HTML documents, and provides an API for changing the content, structure, and style of those documents. Originally, specific browsers like Netscape Navigator and Internet Explorer provided differing, proprietary ways to manipulate HTML documents using JavaScript. The DOM arose from efforts by the World Wide Web Consortium (W3C) to provide a platform and browser neutral way to achieve the same tasks. The DOM represents the structure of an XML or HTML document as an object hierarchy, which is ideal for parsing by standard XML tools.

**DOM Manipulation Methods**

JavaScript provides a large API for dealing with these DOM structures, in terms of both parsing and manipulating the document. This is one of the primary ways to accomplish the smaller, piece-by-piece changes to a web page that we see in an AJAX application.

**DOM Events**

The other important function of the DOM is that it provides a standard means for JavaScript to attach events to elements on a web page. This makes possible much richer user interfaces, because it allows you to give users opportunities to interact with the page beyond simple links and form elements. A great example of this is drag-and-drop functionality, which lets users drag pieces of the page around on the screen, and drop them into place to trigger specific pieces of functionality.

**Presentation: CSS**

CSS (Cascading Style Sheets) provides a unified method for controlling the appearance of user interface elements in your web application. You can use CSS to change almost any aspect of the way the page looks, from font sizes, colors, and spacing, to the positioning of elements. In an AJAX application, a very good use of CSS is to provide user-interface feedback (with CSS-driven animations and transitions), or to indicate portions of the page with which the user can interact (with changes to color or appearance triggered, for example, by mouseovers).

**Communication: XMLHttpRequest**

XMLHttpRequest, a JavaScript class with a very easy-to-use interface, sends and receives HTTP requests and responses to and from web servers. The XMLHttpRequest class is what makes true
AJAX application development possible. The HTTP requests made with XMLHttpRequest work just as if the browser were making normal requests to load a page or submit a form, but without the user ever having to leave the currently loaded web page. Microsoft first implemented XMLHttpRequest in Internet Explorer 5 for Windows as an ActiveX object. The Mozilla project provided a JavaScript-native version with a compatible API in the Mozilla browser, starting in version 1.0. (It’s also available in Firefox, of course.) Apple has added XMLHttpRequest to Safari since version 1.2. The response from the server, either an XML document or a string of text, can be passed to JavaScript to use however the developer sees fit, often to update some piece of the web application’s user interface.

Putting it All Together: JavaScript

JavaScript is the glue that holds your AJAX application together. It performs multiple roles in AJAX development:

- controlling HTTP requests that are made using XMLHttpRequest
- parsing the result that comes back from the server, using either DOM manipulation methods, XSLT, or custom methods, depending on the data exchange format used
- presenting the resulting data in the user interface, either by using DOM manipulation methods to insert content into the web page, by updating an element’s innerHTML property, or by changing elements’ CSS properties

Because of its long history of use in lightweight web programming (and at the hands of inexperienced programmers), JavaScript has not been seen by many traditional application
developers as a "serious programming language," despite the fact that, in reality, it’s a fully featured, dynamic language capable of supporting object-oriented programming methodologies. The misperception of JavaScript as a "toy language" is now changing rapidly as AJAX development techniques expand the power and functionality of browser based applications. As a result of the advent of AJAX, JavaScript now seems to be undergoing something of a renaissance, and the explosive growth in the number of JavaScript toolkits and libraries available for AJAX development is proof of the fact.

Server-Side Scripting
The server is still required to perform tasks like data persistence and input validation. However, in some Ajax architectures, it no longer performs any duty of display or application logic, leaving those things for the browser script to handle.

Event Model ("DHTML")
The event model allows JavaScript to respond to events such as mouse clicks. This is the key to make interactive a conversation between the user and the web site, within the browser.

Web Remoting
Web Remoting is the ability of JavaScript to talk directly with the server, no page refresh is required. XMLHttpRequest dominates most of the headlines, but there are actually several viable remoting technologies.

2.2.4. Ajax versus the Traditional approach

With Ajax, instead of having to send everything to the server in a single, huge mass, then wait for the server to send back a new page for rendering, web developers can communicate with the server in smaller chunks, and selectively update specific areas of the page based on the server’s responses to those requests. This is where the word asynchronous in the AJAX acronym originated. It's probably easiest to understand the idea of an asynchronous system by considering its opposite, a synchronous system. In a synchronous system, everything occurs in order. If a car race was a synchronous system, it would be a very dull affair. The car that started first on the grid would be the first across the finish line, followed by the car that started second, and so on. There would be no overtaking, and if a car broke down, the traffic behind would be forced to stop and wait while the mechanics made their repairs.

Traditional web apps use a synchronous system: you must wait for the server to send you the first page of a system before you can request the second page, as shown in Figure 1.1.
An asynchronous car race would be a lot more exciting. The car in pole position could be overtaken on the first corner, and the car that starts from the back of the grid could weave its way through the field and cross the finish line in third place. The HTTP requests from the browser in an AJAX application work in exactly this way. It’s this ability to make lots of small requests to the server on a needs-basis that makes AJAX development so cool. Figure 2.4 shows an AJAX application making asynchronous requests to a web server. The end result is an application that feels much more responsive, as users spend significantly less time waiting for requests to process, and don’t have to wait for an entire new web page to come across the wire, and be rendered by their browsers, before they can view the results.
2.2.5. Alternatives to Ajax

Ajax meets a need in the marketplace for richer, more responsive web-based clients that don’t need any local installation. It isn’t the only player in that space, though, and in some cases, it isn’t even the most appropriate choice. In the following section, we’ll briefly describe the main alternatives.

**Macromedia Flash-based solutions**

Macromedia’s Flash is a system for playing interactive movies using a compressed vector graphics format. Flash movies can be streamed, that is, played as they are downloaded, allowing users to see the first bits of the movie before the last bits have arrived. Flash movies are interactive and are programmed with Action-Script. Some support for input form widgets is also provided, and Flash can be used for anything from interactive games to complex business UIs. Flash has very good vector graphics support, something entirely absent from the basic Ajax technology stack. Flash has been around for ages and is accessed by a plug-in. As a general rule, relying on a web browser plug-in is a bad idea, but Flash is the web browser plugin, with the majority of browsers bundling it as a part of the installation. It is available across Windows, Mac
OS X, and Linux, although the installation base on Linux is probably smaller than for the other two platforms. For the purposes of creating rich clients with Flash, two very interesting technologies are Macromedia's Flex and the open source Laszlo suite, both of which provide simplified server-side frameworks for generating Flash-based business UIs. Both frameworks use Java/Java 2 Enterprise Edition (J2EE) on the server side. For lower-level control over creating Flash movies dynamically, several toolkits, such as PHP’s libswf module, provide core functionality.

**Java Web Start and related technologies**

Java Web Start is a specification for bundling Java-based web applications on a web server in such a way that a desktop process can find, download, and run them. These applications can be added as hyperlinks, allowing seamless access from a Web Start–savvy web browser. Web Start is bundled with the more recent Java runtimes, and the installation process will automatically enable Web Start on Internet Explorer and Mozilla-based browsers. Once downloaded, Web Start applications are stored in a managed “sandbox” in the filesystem and automatically updated if a new version is made available. This allows them to be run while disconnected from the network and reduces network traffic on reload, making the deployment of heavy applications weighing several megabytes a possibility. Applications are digitally signed, and the user may choose to grant them full access to the filesystem, network ports, and other resources. Traditionally, Web Start UIs are written in the Java Swing widget toolkit, about which strong opinions are held on both sides. The Standard Widget Toolkit (SWT) widgets used to power IBM’s Eclipse platform can also be deployed via Web Start, although this requires a bit more work. Microsoft’s .NET platform offers a similar feature called No Touch Deployment, promising a similar mix of easy deployment, rich UIs, and security. The main downside to both technologies is the need to have a runtime preinstalled. Of course, any rich client needs a runtime, but Flash and Ajax (which uses the web browser itself as a runtime) use runtimes that are commonly deployed. Java and .NET runtimes are both very limited in their distribution at present and can’t be relied on for a public web service.
3. Frameworks and graphic libraries for Ajax

Up until now, we’ve looked at Ajax either at an abstract architectural level, or from down in the tunnels underneath the structure. In this chapter, we will take a look at several frameworks and instruments at our disposal which can be a valid support during the Ajax development.

3.1. Frameworks, Toolkits and Libraries introduction

An explosion of libraries and frameworks has been one of the fortunate consequences of the Ajax phenomenon. They come in all shapes and sizes and in any licensing model you’d care to name. The listing here makes a good companion to the Ajax Patterns. As Ajax has taken off, we have been inundated with projects claiming to have Ajax support. In the following figure we can see a diagram which represents all the layers of Ajax frameworks.

- **Ajaxian Web Frameworks**
  - Rails, Tapestry, WebWork, ASP.NET, ...

- **UI Toolkit**
  - Dojo, SmartClient, Backbase, ...

- **Remoting Toolkit**
  - DWR, JSON-RPC, dojo.io.bind()

**Figure 3.1 – Layers of Ajax Frameworks**

**Remoting Toolkit**

The lowest level of Ajax helpers is a remoting toolkit. If you were to create your own toolkit, this would probably be where you started out: wrapping XMLHttpRequest with your own API to make life easier. A really good remoting toolkit should be able to do much more than simply hide our ugly try/catch XHR instantiation code. What should happen if your Ajaxian page is loaded by a browser that does not support XMLHttpRequest? It ought to find a way, if possible, to provide all (or at least some) of its functionality by other means. For example, some remoting toolkits will use a hidden iframe to provide fake XHR support to the page. The Dojo Toolkit, JSON-RPC, and Prototype are all pure JavaScript frameworks that are agnostic about the world of the server side (although Prototype was built with Ruby on Rails in mind).

Others, such as DWR (Direct Web Remoting), couple a JavaScript client library with a server side listener piece written for the Java platform. JSON-RPC itself has various bindings for many back-end languages. DWR, JSON-RPC, Ajax.NET, and SAJAX are all examples of ORB-based Ajax.
frameworks. They allow you to map JavaScript methods to back end services, treating the client-side JavaScript as though it could directly access your server-side objects.

**iframes**

Prior to the broad adoption of the XMLHttpRequest object, many web applications were using a hidden iframe to accomplish in-page round trips back to the server. An iframe is just like a normal HTML frame (a container that can be targeted at a URL and render the results) except that it is embedded in another page. These applications simply created an iframe of 0px by 0px, then caused it to refresh against a given URL in order to pull more data back from the server. While the technique is valid, and worked for many, there were two main problems. The first is, if you wanted multiple asynchronous requests, you had to have multiple iframes. More important is the question of coding intentionally, while XMLHttpRequest, is an object specifically designed for initiating, monitoring, and harvesting the results of in-page postbacks, iframe always feels a little like cheating.

**UI Toolkit**

Above, or potentially alongside, remoting toolkits we find JavaScript UI libraries. These give us the ability to use rich UI components and effects out of the box, but different in many ways.

**Richer UI Components**

Toolkits such as Dojo give us rich widgets like trees, tabbed panes or menus. These are self-contained, instantiable UI components that can be used to compose a rich, though still very “web-ish” application. The result is still unmistakably an HTML UI.

**Web Application Toolkit**

Toolkits such as SmartClient aim to give you widgets that build a UI that looks and feels the same as a native application on Windows or Mac OSX. These are useful if you are building an application that happens to be on the web, versus a website that uses a couple of UI effects and components. SmartClient, for example, features widgets that make the page look and feel exactly like a Windows NT application.

**Markup Based**

Backbase allows you to add rich components through a markup programming API. Such a system could potentially enable a new generation of visual development tools. Part of the problem with such tools is the conflict between markup and code. Traditional JavaScript based pages have caused problems for such tools because it is difficult to provide visual representations of code resources. An all-markup framework, on the other hand, would provide the right abstractions for these kinds of development environments. See, for example, the markup-based components in ASP.NET, Tapestry and Java Server Faces.

**Simple JavaScript Driven Effects**
Several frameworks use pure JavaScript and HTML to create extremely complex UI effects. These kinds of frameworks provide high-level abstractions on top of some meaty JavaScript, making the effects simple to implement in your application. The results are often completely cross-browser compatible, and fail gracefully to static HTML in legacy browsers.

**Ajaxian Web Frameworks**

At the top of the tower are the web frameworks that are aware of Ajax. This is a growing group, and covers all of the platforms. All the major players are represented: Java, .NET, Ruby, PHP, Python, Perl, etc. Once again, the various frameworks offer different models for how you can work with them in an Ajaxian world.

**Code Generation**

The Ruby on Rails community jumped on Ajax like nobody else. They offer high-level Ruby helper functions which generate Prototype-based JavaScript code. WebWork2 is doing the same thing on the Java platform, utilizing the Dojo Toolkit as the base JavaScript framework. Many other frameworks are following suit, from Spring to CherryPy to PHP.

**Component-based**

ASP.NET had Ajaxian components before there was Ajax. Other frameworks such as JavaServer Faces and Tapestry on the Java platform join ASP.NET by letting you use components that may happen to use Ajaxian techniques. In this world, you drag your DataTableComponent onto your designer view and start tweaking the property sheet for that component.

### 3.2. Frameworks, Toolkits and Libraries for the Online Editor

After having analyzed in depth all the solutions currently available (see Appendix A) and having focused our research on tools suitable to our goal we have decided to use two packages that allow us to unleash the full potential of JavaScript and DOM but at the same time be compatible, versatile and cross-browser. Our choice has fallen on the OpenJacob Draw2D Javascript library to create diagrams inside the browser and on the Yahoo! User Interface library (YUI) which is a set of utilities and controls, written in JavaScript, for building richly interactive web applications. In the following paragraphs we will describe this two powerful libraries.

#### 3.2.1. OpenJacob Draw2D library

OpenJacob Draw2D is a free Javascript library to create drawings and diagrams. The library is released with an Open-source license under LGPL. The User interface allows interactive drawing by using a standard browser. No additional software has to be used and no third party plug-ins. Draw2D is a client software written in Javascript which can be used to create graphic web applications without being supported by server-side components. In the following figure we can see a simple example of what we can create with Draw2D.
The user can run the client application also from the local filesystem without a webserver. Our project of a Web editor tool will support an offline version runnable in local from the browser and an online version with some server-side script necessary mainly to load and save XML personal projects. So it is possible to realize a server support and manage the interaction with the web client using all the technologies available today, for example Ajax technologies.

Because using Javascript, OpenJacob Draw2D supports the vast majority of browsers that are in general use. After the testing with all the most important Web browser we can conclude that Draw2D can be used taking full advantage of its powerful capabilities using Mozilla Firefox 1.5 and later version and Internet Explorer 6 and later versions. Opera has revealed some problem
using advanced but crucial functionalities of Draw2D library like the link connection between objects in the work area.

![Diagram](image)

**Figure 3.4 – Architecture of an example application**

Before choosing this library for our project we have analyzed some example of realized application, the API documentation and the main classes to see the structure and we have also tested it practically. The main functionalities available in the library are the following:

- Generation of shapes and figures inside the work area
- Use of connection ports (input / output) to connect objects with links
- Personalization of links
- Drag and Drop functionality
- Undo / Redo functionality
- Delete objects functionality
- Pan function to move the diagram in the work area
- Command management functionalities to add, connect, delete, move, ...
- Dynamic editing of objects (figures and connections) name and properties, for example using dropdown menu to edit the properties or to connect other objects
- Possibility to associate images to a figure (to represent different type of objects, for example to represent a database, a service or other)
- Use of compartment figures, figures which can contain other figures or objects (for example to create forms)
- Possibility to use dynamic windows (property windows and dialog windows) to manage the environment, the work area and the diagram
- Use of a tool palette window to add figures to the work area
- Tooltip functionality for the objects
- Use of dropdown menus
- Possibility to use XML format to represent the diagram
- Possibility to implement a zoom instrument (in development phase)
- Possibility to create HTML forms in dialog windows
- Identity for JavaScript objects with an automatic unique Id method generation. The identity for an object does not change during its lifecycle
- Management system for events

The class hierarchy of the library is complex and not simple to read, in the following diagrams we will show a reduced Object Oriented view which represents all kinds of figures, lines and commands on them which can be used with Draw2D.

Figure 3.5 – Draw2D figures class hierarchy
The relevant objects for our objectives are essentially the ImageFigure and Connection objects, the Port object, the Command object and all its sons, the PropertyWindow and the ToolPalette objects.

**ImageFigure**

An ImageFigure is a Node object that allows to create, inside the workflow area, a figure which is represented by a personal .PNG graphic file. All figures of the tool will be realized using this specific Node object.

**Connection**

A Connection is a Line object that allows to create a link between two figures. This object is very crucial for the navigation flow of the diagram.

**Port**

Each ImageFigure of our diagram has a connection Port, this one is needed to generate a link between two figures. The user can connect objects by dragging and dropping an object Port to another object Port. Each port works as InputPort and OutputPort at the same time, that is an object is able to accept input links but also to send off output links.

**Command**

This is the class for execute all kinds of commands available in the Draw2D framework. The main commands are: CommandAdd to add an object to the workflow area, CommandConnect and CommandReconnect to connect objects with links, CommandDelete to remove objects from the workflow area, CommandMove and CommandMoveLine to move objects. This class is also the base class for the undo / redo support of the framework.

**PropertyWindow**
The **PropertyWindow** is maybe the most important class for our project. This window allows to monitor the selection of all the elements. When the user change the selection of the element in the workflow area, the **PropertyWindow** keeps this event and shows the element properties. This window will be crucial to manage the state and the properties of the selected objects.

**ToolPalette**

The **ToolPalette** is a palette which can be personalized by inserting in it all the objects that we want to make available for the user. The user will have the possibility to click an object on the palette and to insert in the workflow area. The **ToolPalette** will therefore be the work instrument of the tool.

To better understand how Draw2D works we show an example of implementation in which we create a tool palette that will be used from the user to add objects to the diagram in the work area. In the main page of the project we have to import all the necessary Javascript files of the library to exploit all needed objects and their functions.

```html
<html>
  ...
  // Import all .js library files needed
  <SCRIPT src="../wz_jsgraphics.js"></SCRIPT>
  ...
  // Create the work area HTML element inside the page
  <div id="paintarea" style="width:3000px;height:3000px" >
    ...
    <script>
      // Generate the Draw2D work area and associate the HTML element
      var workflow  = new Workflow("paintarea");

      // Add a simple PropertyDialog to the Canvas
      // This will display the properties of the current select object
      var dialog = new VectorPropertyWindow();
      workflow.showDialog(dialog,400,10);

      // Add the Tool Window to the screen
      var w = new VectorPalette();
      workflow.setToolWindow(w);

      // move the tool palette out of the hint message at the top left corner
      w.setPosition(20,140);

      window.onscroll=function(){workflow.onScroll();};
    </script>
  </div>
</html>
```
3.2.2. Yahoo! User Interface library (YUI)

The Yahoo! User Interface (YUI) Library is a set of utilities and controls, written in JavaScript, for building richly interactive web applications using techniques such as DOM scripting, DHTML and AJAX. The YUI Library also includes several core CSS resources. Some pieces of YUI, like the Event Utility, operate under the hood and are simply there to make in-browser programming easier. Others, like the Calendar Control, comprise visual widgets that can be placed on your page with a default look-and-feel. YUI version 2.3.0 includes a new, cohesive, visual look and feel, or skin.

All components in the YUI Library have been released as open source under a liberal BSD license that permits both commercial and non-profit use.

YUI supports the vast majority of browsers that are in general use. All YUI components work well in all that browsers designated as "A-Grade" browsers. A-grade support is the highest support level. By taking full advantage of the powerful capabilities of modern web standards, the A-grade experience provides advanced functionality and visual fidelity. A-grade browsers are identified, capable, modern and common.

This chart lists browsers that receive A-Grade support as defined by Graded Browser Support († means "the most recent").
The Yahoo! User Interface (YUI) Library includes a wide set of components. The following list illustrates the composition of the library (in bold the relevant components for our project):

1. **YUI Core:**
   1.1. The YAHOO Global Object (base requirement for all YUI components)
   1.2. **DOM Collection (convenience methods for DOM interactions)**
   1.3. Event Utility (event normalization and custom events)

2. **YUI Library Utilities:**
   2.1. Animation Utility
   2.2. Browser History Manager
   2.3. Connection Manager (for XHR/Ajax)
   2.4. Cookie Utility
   2.5. DataSource Utility
   2.6. **Drag and Drop Utility**
   2.7. Element Utility
   2.8. Get Utility (dynamic script/css loading)
   2.9. ImageLoader Utility
   2.10. JSON Utility
   2.11. Resize Utility
   2.12. Selector Utility
   2.13. The YUI Loader Utility

3. **YUI Library Controls/Widgets:**
   3.1. AutoComplete
   3.2. Button
   3.3. Calendar
   3.4. Charts [experimental]
   3.5. Color Picker
   3.6. **Container (including Module, Overlay, Panel, Tooltip, Dialog, SimpleDialog)**
   3.7. DataTable
   3.8. ImageCropper
   3.9. Layout Manager
   3.10. Menu
   3.11. Rich Text Editor
   3.12. Slider
   3.13. **TabView**
3.14. TreeView
3.15. Uploader [experimental]

4. **YUI Library CSS Tools:**
   
   4.1. CSS Reset (neutralizes browser CSS styles)
   4.2. **CSS Base (applies consistent style foundation for common elements)**
   4.3. CSS Fonts (foundation for typography and font-sizing)
   4.4. CSS Grids (more than 1,000 CSS-driven wireframes in a 4KB file)

**DOM Collection**

The Dom Collection comprises a family of convenience methods that simplify common DOM-scripting tasks, including element positioning and CSS style management, while normalizing for cross-browser inconsistencies.

**Drag and Drop Utility**

The Drag & Drop Utility allows you to create a draggable interface efficiently, buffering you from browser-level abnormalities and enabling you to focus on the interesting logic surrounding your particular implementation. This component enables you to create a variety of standard draggable objects with just a few lines of code and then, using its extensive API, add your own specific implementation logic.

**Container**

The Container family of components is designed to enable developers to create different kinds of content-containing modules on the web. Module and Overlay are the most basic containers, and they can be used directly or extended to build custom containers. Also part of the Container family are four UI controls that extend Module and Overlay: Tooltip, Panel, Dialog, and SimpleDialog. The containers which are useful for the creation of the graphic interface of our editor tool are Panel, Dialog and SimpleDialog. I will shortly describe these components and show some examples of the their great interactive and graphic qualities.

The Panel control is an extension of Overlay that is meant to behave similarly to an OS window. Unlike true browser popup windows, panels are floating DHTML elements embedded directly within the page context. The Panel control extends the functionality of Overlay, adding support for modality, drag and drop, and close/dismiss buttons. Panel includes a pre-defined stylesheet to support default look and feel characteristics. The showed example implements two simple Panels, one from markup and one purely from script, and shows how to configure options like draggability.

```javascript
// Instantiate a Panel from script
YAHOO.example.container.panel2 = new YAHOO.widget.Panel("panel2", { width:"320px", visible:false, draggable:false, close:false } );
YAHOO.example.container.panel2.setHeader("Panel #2 from Script");
```
YAHOO.example.container.panel2.setBody("This is a dynamically generated Panel.");
YAHOO.example.container.panel2.setFooter("End of Panel #2");
YAHOO.example.container.panel2.render("container");

Figure 3.8 – Simple Panel example

Another example shows how to use CSS with panels. In fact panels can be skinned using CSS to customize the look and feel of each component.

Figure 3.9 – Skinning a Panel with CSS

The Dialog component is an extension of Panel that is meant to emulate the behavior of a dialog window using a floating, draggable HTML element. Dialog provides an interface for easily gathering information from the user without leaving the underlying page context. The
information is collected via a standard HTML form; Dialog supports the submission of form data either through an XMLHttpRequest, through a normal form submission, or through a fully script-based response (where the script reads and responds to the form values and the form is never actually submitted via HTTP to a server). This example shows a Dialog panel to collect data from the user and send it to the server using XMLHttpRequest (Ajax) via the YUI Connection Manager.

The SimpleDialog component is an extension of Dialog that reproduces the behavior of a simple dialog box (but without using an actual browser popup window); its primary use is to elicit binary decisions from the user (yes/no, okay/cancel, etc.). SimpleDialog makes it easy to implement this kind of interaction.

Tabview

The TabView component is designed to enable developers to create navigable tabbed views of content.

```html
<div id="demo" class="yui-navset">
  <ul class="yui-nav">
    <li><a href="#tab1"><em>Tab One Label</em></a></li>
    <li class="selected"><a href="#tab2"><em>Tab Two Label</em></a></li>
    <li><a href="#tab3"><em>Tab Three Label</em></a></li>
  </ul>
</div>
```
TabView Control: Build from Markup

This demonstrates how to build a TabView from markup.

CSS Base

Base is an optional CSS file that compliments YUI’s core CSS foundation. Base applies a consistent style foundation for common HTML elements across A-grade browsers.
4. MVC-WebFlow model

4.1. Introduction

Web Engineering has proven a valid approach to the design and implementation of Web applications. A lot of approaches and notations exist to increase efficiency, reuse, reliability, and maintainability. Unfortunately, these approaches are not as widely adopted as one could expect. One of the reasons is that they are often too far away from the average developer and designer way of working. This becomes more and more evident in application development where time to market and continuous update are crucial. Indeed, in these scenarios CMS systems and pre-designed solutions are often preferred. Their main advantage is that most of the updates can be applied directly at runtime, simply by updating the data or the metadata that describe the application content and structure. Unfortunately, this cuts off any design abstractions in the development process, with several disadvantages in terms of overall design quality and cleanness of the application structure. Some intermediate solutions have been proposed, that try to provide some design help while developing the application (e.g., see App2You, where data model and business logic are built at runtime while the designer adds contents and pages to the application [http://app2you.com]), but they still lack a modeling phase of the application.

With this project we want to offer a simple conceptual model which allows developers to manage the application structure exploiting the MVC architecture. To ease the task, we offer an online tool called MVC-Webflow for drawing the data and application structure and turning the specification to a running application on the fly. Using the tool the developer is able to define the flow of the application, the view composed by the JSP pages and the controller which process the user requests and acts as intermediary between the business logic and the view. The tool will provide the possibility to edit each component to define the view of the application and the controller. The tool is implemented on the AJAX platform, for granting maximum usability and interface quality.

4.2. Aims and benefits

The main purpose of the MVC-Webflow tool is to lower the barrier to the adoption of Web Engineering techniques in the development practices. This aim is addressed by the following basic choices:

- We allow to model applications with a MVC approach adapted for the Web (namely, the Model 2 design framework), which is widely adopted by developers and has some valid implementation counterparts (e.g., Jakarta Struts);
- We offer an online design tool that provides both solid model-driven design features typical of Web Engineering and quick application evolution thanks to a Web based interface that does not require a rigid development process;
- We provide basic on the fly code generation on Model 2 reference architectures (at the moment, we provide code generation for Struts), with special attention to the specification of the Controller.

The advantages of MVC architectures are well known. They allow the developer to design the Web application defining three separate levels: Model, View and Controller. In complex applications, separation of concerns is fundamental; MVC allows it by separating data and business object (the Model), user interface aspects (the View), and the rules that control the execution logics of the application (the Controller). MVC decouples business logics and data access from the presentation thanks to the Controller intermediate component. For the Web, a special version of MVC called Model 2 (also called MVC-2) has been studied.

![Figure 4.1 – Architecture of a typical MVC application](image)

In our proposal, we offer automatic code generation on the Jakarta Struts platform, an implementation of the Model 2 paradigm. From an implementation point of view, the configuration of the Jakarta Struts controller is specified in the file `struts-config.xml`, that
specifies the logics for invoking the *action classes*, the model objects playing as interfaces toward the data sources and the business logic.

We provide a visual model for easily describing the controller behaviour and the used action classes. The visual model can be specified through an online tool implemented on AJAX technologies, that allows to design the diagrams, to store them on a server, to retrieve them and to generate the basic pieces of the Struts components. We automatically generate the contents of the struts-config.xml file and the stubs of the action classes from the diagram created using the model. The advantages of the approach are:

- simple and quick design of the application model;
- good overall vision of the application, with a notation directly referring to the MVC models;
- independent design of the controller flow and of the called action;
- easy reuse of parts of the application defined in the diagram.

4.3. Jakarta Struts overview

To implement the MVC architecture for the generated application we will use the MVC framework Jakarta Struts. An application framework is a skeleton of an application that can be customized by the application developer.

Struts is an application framework that unifies the interaction of the various components of a J2EE Web application — namely Servlets, JSP pages, JavaBeans, and business logic — into one consistent whole. Struts provides this unification by implementing the Model-View-Controller (MVC) design pattern. Struts provides an implementation of the MVC design pattern for Web applications. Jakarta Struts is incredibly useful in helping to create excellent Web applications. When you use Jakarta Struts, your applications should work more effectively and have fewer bugs. Just as important, Struts should save you hours and hours of programming and debugging. Jakarta Struts ask the developer to create and configure the controller defining the configuration file struts-config.xml and implementing the Action Class performing the function interface between view and business logic. It is possible to obtain the contents of the file struts-config.xml in automatic mode by the diagram created using model. Action Classes can instead be derived from Action units.

4.3.1. Struts architecture

The architecture of Struts provides a wonderful mechanism that, when followed, ensures that the MVC pattern remains intact. Although Struts provides a concrete implementation of the Controller part of the pattern, as well as providing the connections between the Controller and Model layers and between the Controller and View layers, it doesn’t insist on any particular View paradigm or require that you construct the Model in a particular way.
**The Struts Controller**

Although Struts does not provide or require any particular Model or View components of the MVC pattern, it does implement the Controller as well as the mechanisms that bind the three layers and allow them to communicate with each other. The primary controller class is a Java Servlet called the ActionServlet. This class handles all user requests for Struts-managed URLs. Using information in the configuration files, the ActionServlet class then gets the appropriate RequestProcessor class that collects the data that is part of the request and puts it into an ActionForm, a Bean that contains the data sent from or to the user's form. The final step of the Controller is to delegate control to the specific handler of this request type. This handler is always a subclass of the Action class. Figure 4 shows how Struts uses the MVC pattern. The Action subclass is the workhorse of the Controller. It looks at the data in the user's request (now residing in an ActionForm) and determines what action needs to be taken. It may call on the business logic of the Model to perform the action, or it may forward the request to some other View. The business logic may include interacting with a database or objects across the network or may simply involve extracting some data from an existing JavaBean. After the necessary action has been performed, the Action subclass then chooses the correct View to send back to the user. The View is determined by the current state of the Model's data (the model state) and the specifications you defined in the Struts configuration file. Figure 4.2 shows the principal classes of the Struts Controller.

![Diagram of the Struts MVC pattern](image.png)

**The Struts View**

As mentioned, Struts does not provide, nor is it dependent on, a specific presentation technology. Many Struts applications use JSP (JavaServer Pages) along with the Struts tag library (Struts and Struts-EL), JSTL (JSP Standard Tag Library), and JSF (Java Server Faces). The JSP specification provides for the creation of HTML-like tags that extend the functionality of JSP. These custom tags are bundled by their creators into custom tag libraries and are
accompanied by a descriptor file called a Tag Library Descriptor (tld). The Struts and Struts-EL tag libraries are examples of this extended functionality. Our examples throughout the book use JSP along with Struts-EL, JSTL, and other tag libraries. For new projects, the recommendation from the Struts Web site is to use not the standard Struts tag libraries, but instead the Struts-EL tag library along with JSTL. The Struts-EL tags library is really a reimplemention of the standard Struts tag library to make it compatible with JSTL’s method of evaluating values. However, when a JSTL tag implemented the same functionality, the Struts tag was not reimplemented in the Struts-EL library.

**The Struts Model**

Nothing in Struts dictates how to construct the Model. However, the best practice is to encapsulate the business data and operations on that data into JavaBeans, as we described previously when discussing Data Transfer Objects. The data and operations may reside in the same class or in different classes, depending on your application. The operations represent the business logic that your application is defining. Operations may be the rules that should operate on a particular business entity. For example, if you’re writing a purchasing system, part of the business data might be an entity called a Purchase Order. You may encapsulate this data into a class called PurchaseOrder as a way of representing the Purchase Order entity. Furthermore, you may choose to place your business rules directly into this class, or you may choose to put the rules into a different class. The connection between the Controller and Model rests in the code that you write in the Action subclasses. The Action subclasses contain the analysis of the user’s request that determines the interaction (if any) with the Model. The Action subclass initiates any action required to handle a user’s request, thereby creating the connection with the Model. When formulating a response, the Controller may pass some or all of the Model data to the View through the use of the ActionForm Bean. Although this Bean is a data container, it should not be considered part of the Model but rather just a transport mechanism between the Model and the View. Just as often, the View may directly reference the Model’s data by referencing one or more of the Beans that belong to the Model. The standard MVC pattern describes an interaction between the Model and the View so that when the Model’s data changes, it can immediately push those changes out to the View so the user sees them. However, this is more difficult to achieve in the Web application architecture. Consequently, the View is commonly updated by the user requesting it.

**The Struts configuration file**

The Struts configuration file performs an important role in structuring your Struts application. Although it is not really part of the Model, View, or Controller, it does affect the functioning of the three layers. The configuration file allows you to define exactly which of your Action subclasses should be used under what circumstances and which ActionForm should be given to that Action subclass. So you specify part of the Controller interaction in the configuration file. In addition, when the Controller decides which View to return to the user, it chooses the particular View according to specifications in the configuration file. Thus the configuration file actually defines
many of the connections between the MVC components. The beauty of the configuration file is that you can change the connections without having to modify your code. The configuration file does much more than defining connections.

Then the main task of the MVC-WebFlow editor tool will be the generation of the Struts configuration file, the action classes and the view JSP pages of the Web application modelled in the diagram. In the following code we can see an example of configuration file.

```xml
<form-validation>
  <formset>
    <form name="creditCardForm">
      <field property="validFrom" depends="required,mask">
        <var>
          <var-name>mask</var-name>
          <var-value>\d{4}/\d{2}</var-value>
        </var>
        <arg0 key="validFrom" />
      </field>
    </form>
  </formset>
</form-validation>

<struts-config>
  <form-beans>
    <form-bean name="creditCardForm" type="org.apache.struts.validator.DynaValidatorForm">
      <form-property name="number" type="java.lang.String"/>
    </form-bean>
  </form-beans>

  <global-forwards>
    <forward name="home" path="/index.jsp"/>
  </global-forwards>

  <action-mappings>
    <action path="/checkCard" type="it.polimi.tiaw.CheckCreditCardAction" name="creditCardForm" scope="request" input="home">
      <forward name="success" path="/ok.jsp"/>
    </action>
  </action-mappings>

  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
  </plug-in>
</struts-config>
```
4.4. Definition of the conceptual model

The MVC-WebFlow metamodel is defined by a set of primitives and by the rules for specifying the connections among those primitives. The diagram resulting from the design phase represents the conceptual schema of the MVC application, that mainly consists of JSP pages (view components), actions (model components), and connections (controller definitions). The primitives of the model are represented in Table 1. The developer has to create the diagram which specifies the flow of the application but has also to define the content of each single component, from JSP pages to Action units which are the most important elements of the model. Above we will see in detail what are this three components of the model and their roles.
Table 1. Primitives of the MVC-Webflow metamodel

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP page</td>
<td>Abstract representation of a JSP page</td>
</tr>
<tr>
<td>Action unit</td>
<td>Abstract representation of an Action component</td>
</tr>
<tr>
<td>Form unit</td>
<td>Abstract representation of an Action form</td>
</tr>
<tr>
<td>JSP fragment</td>
<td>Representation of a fragment of JSP code reusable</td>
</tr>
<tr>
<td>Connection Link</td>
<td>Abstract representation of a connection Link between JSP pages and Action units or other JSP pages</td>
</tr>
<tr>
<td>Success Link</td>
<td>Abstract representation of a Success Link out from an Action unit</td>
</tr>
<tr>
<td>Failure Link</td>
<td>Abstract representation of a Failure Link out from an Action unit</td>
</tr>
<tr>
<td>Containment Link</td>
<td>Abstract representation of a containment Link used to define Form unit for an Action unit or JSP fragment unit for a JSP page</td>
</tr>
</tbody>
</table>
**Action unit**

*Action units* are the most important elements of the model and effectively realize the Controller. *Action units* specify the way in which the application reacts to the user input: the correct *Action unit* is called based on the controller decision, and then it processes the user requests, by retrieving data and/or calling the business logic, and provides the information for building the response to the user. The *Action unit* behaviour must be defined completely by the developer. An *Action unit* can be linked to a *JSP page* that shows its results or can be connected to another *Action unit* for performing further processing (possibly based on the results of the first unit).

**JSP page**

*JSP pages* represent the View of the model. Each *JSP page* corresponds to a page in the final Web application and allows to present contents to the user. The *JSP page* must be defined by the developer using any sort of editing tool. *JSP pages* interact with *Action units* through links. The *JSP page* receives the content to be shown in the page. The *JSP page* can call *Action units* and send it some parameters.

**Form unit**

*Form units* must be connected to an *Action unit* and represent the user input from one or more form fields. In this way, *Action units* can receive the user input coming from the pages of the view.

**JSP fragment**

*JSP fragments* are portions of reusable code that can be embedded in *JSP pages*. The main advantage of this primitive is to provide modularization and reusability to the designer. Notice that *JSP fragments* cannot have input links.

**Connection Link**

*Connection Links* allow the developer to define the control flow of the application. A control flow is a directed arc represented with an arrow from the source to the destination. It can connect: (i) a *JSP page* to an *Action unit* or to another *JSP page*; (ii) an *Action unit* to a *JSP page* or to another *Action unit*. A link can carry content objects and parameters. Two variants of the *Connection Links* are used for specifying the outcome of a control unit: *Success Links* are shown with a green arrow; *Failure Links* are shown as red arrows. After its execution, the *Action unit* will follow the success path or the failure path depending on a Boolean obtained results. After the execution of the business logic the *Action unit* can call the success path or the failure path evaluating the obtained results. A path can be a *JSP page* but also another *Action unit*.

**Containment Link**

*Containment Links* are used to include Form units into *Action units* and *JSP fragments* into *JSP pages*. This type of link establishes a relation of containment between the two units. For example
a JSP page can contain more than one JSP fragment unit and this one can be used by more Action units.

4.5. Object Oriented view of the model

In this section we briefly outline the MVC-WebFlow Metamodell. The class diagram in Figure 4.4 represents the object oriented description of the metamodel. The main model element is Project, which represents the whole Web application. The project is composed by the Data model and the Web model. For lack of space, we concentrate only on the Web model, since the Data model can be specified using a standard representation (E-R, class diagram, ontology model, and so on). A Web model is composed of Web elements, which can be links, JSP pages and fragments, action units and form units. The properties that can be specified for each element are summarized in the following tables.

![Figure 4.4 – Object Oriented view of the model]
### 4.5.1. Action unit

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of the Form Bean defined in the Formset (it’s a precompiled attribute obtained from the link with the Form unit)</td>
</tr>
<tr>
<td>Path</td>
<td>String</td>
<td>Path used in the view (JSP page) to call a specific Action Mapping</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Path of the Action Class which contains details about business logic invocation and response of the controller</td>
</tr>
<tr>
<td>Scope</td>
<td>String</td>
<td>Persistence of the Form Bean (Request or Session)</td>
</tr>
<tr>
<td>Validate</td>
<td>Boolean</td>
<td>To use or not the validation offered by the framework</td>
</tr>
<tr>
<td>Landmark</td>
<td>Boolean</td>
<td>If the Action unit can be recalled from any point of the application (for example logout Action unit)</td>
</tr>
<tr>
<td>Execute method</td>
<td>String</td>
<td>To define the logic for the interaction between the Controller and Business Services (the Model of MVC)</td>
</tr>
</tbody>
</table>

### 4.5.2. Form unit

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormBean Name</td>
<td>String</td>
<td>Name of the Form Bean defined in the form-beans, it’s the attribute key of the request/session object where the Form Bean is stored</td>
</tr>
<tr>
<td>FormBean Type</td>
<td>String</td>
<td>The qualified name of the Form Bean Class (Action Form) that represents the input</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Boolean</td>
<td>To define the automatic generation of the Form Bean Class by the framework</td>
</tr>
<tr>
<td>Form Properties</td>
<td>Array of Form Property</td>
<td>Array containing all the property elements of the Form Bean</td>
</tr>
<tr>
<td>ValidationXML</td>
<td>String</td>
<td>Rules of Validation</td>
</tr>
</tbody>
</table>

Form property type

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Property Name</td>
<td>String</td>
<td>Name of a Form Bean property</td>
</tr>
<tr>
<td>Form Property Type</td>
<td>String</td>
<td>Type of a Form Bean property (for example Java.Lang.String)</td>
</tr>
</tbody>
</table>

4.5.3. JSP page unit

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP page name</td>
<td>String</td>
<td>Name of the JSP page</td>
</tr>
<tr>
<td>Validate</td>
<td>Boolean</td>
<td>To generate the Struts tag <a href="">html:javascript</a> to implement the client-side validation (controlling the Form attribute Dynamic)</td>
</tr>
<tr>
<td>Landmark</td>
<td>Boolean</td>
<td>If the JSP page can be recalled from any point of the application</td>
</tr>
</tbody>
</table>

4.5.4. JSP fragment unit

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP fragment name</td>
<td>String</td>
<td>Name of the JSP fragment</td>
</tr>
</tbody>
</table>

4.5.5. Link

Links represents the connections among the elements of the Web model. They realize the navigation flow of the Web application, the invocation of Action units and the definition of Form unit for an Action and also JSP fragment to reuse in JSP pages.

4.5.5.1. Containment link

Fragment link

It connects a JSP fragment with JSP pages which use it.

Form link

It connects the Form unit with an Action unit and it allows to define the attribute Name of the Action unit which points out the definition of a Form Bean for an Action Mapping.
4.5.5.2. Navigation link and Action Link

They allow to define the navigation flow of the Web applications and govern the Action Mappings of the controller. Navigation links establish which Action to call after a request (JSP page) while Action links establish which view (JSP page) to call after the termination of the Action. In plain words these links define the associations among JSP pages and Action Mappings. In particular Success Links and Failure Links allow to define the success path and the failure path for the action unit.
5. MVC-WebFlow Ajax tool

5.1. Objectives and context

MVC-WebFlow project deals with the development of an online editing tool to ease the specification of MVC-WebFlow models. The main purpose of the MVC-Webflow tool is to lower the barrier to the adoption of Web Engineering techniques in the development practices. The tool will allow to model applications with a MVC approach adapted for the Web (namely, the Model 2 design framework), which is widely adopted by developers and has some valid implementation counterparts (e.g., Jakarta Struts). This design tool will provide both solid model-driven design features typical of Web Engineering and quick application evolution thanks to a Web based interface that does not require a rigid development process. The final aim of the tool is a basic on the fly code generation on Model 2 reference architectures.

The advantages of MVC architectures are well known. They allow the developer to design the Web application defining three separate levels: Model, View and Controller. In complex applications, separation of concerns is fundamental; MVC allows it by separating data and business object (the Model), user interface aspects (the View), and the rules that control the execution logics of the application (the Controller). MVC decouples business logics and data access from the presentation thanks to the Controller intermediate component.

In the following figure we can see all the various stages of development of Web applications. MVC-WebFlow tool acts on the Data design step, the Hypertext design step and the Implementation step.

![Development process and tool intervention](image)

Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for the Web application, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirements analysis is critical to the success of a development project and will produce a set of requirements (requirements specification) to be respected.
After the Business Requirements individuation there is the **Conceptual Design** phase. Starting from the application workflow the developer is able to design the Web application. It’s here where the MVC-WebFlow Model and Tool are of vital importance. We can identify two steps in the process of modelling the application, the **Data design** and the **Hypertext design**. Our tool will give to the developer the possibility to design the Web flow of the application (Hypertext design) but also the Data structure of the application. For the moment the tool development will be concentrated on the Web flow design. In this phase the developer has to design the **MVC-WebFlow diagram**, respecting the model rules, that allows to define the flow of navigation for every possible request that the system must be able to process. The creation of this diagram is closely linked to the previous point of designing the workflow. The **Implementation** step of the development process is based on the code generation capabilities of the tool. The MVC-WebFlow tool will allow the developer to generate the basilar structure of the Web application with the view components (JSP pages) and the controller components (struts-config.xml file, the web.xml file, Action classes, ...). The business objects will be realized manually by the developer.

In the following figure we can see the context diagram for the editor tool.

![Figure 5.2 – Context diagram of the tool](image)
5.2. Business requirements and functionalities

In our proposal, we want to offer an online graphic tool for the automatic code generation on the Jakarta Struts platform, an implementation of the Model 2 paradigm. From an implementation point of view, the configuration of the Jakarta Struts controller is specified in the file `struts-config.xml`, that specifies the logics for invoking the action classes, the model objects playing as interfaces toward the data sources and the business logic.

Using the tool the developer is able to define the flow of the application, the view composed by the JSP pages and the controller which process the user requests and acts as intermediary between the business logic and the view. The tool will be composed of a client-side graphic interface, accessible by a browser, and a server-side environment (JSP script, Java Servlet and Java classes) devoted to the application code generation. The graphic interface of the editor is the interactive part which allow the user to create the diagram inside the browser which represents the application structure and flow.

With the MVC-WebFlow tool the developer is able not only to design web application flow diagrams but also to store them on a server, to retrieve them and to generate the basic pieces of the Struts components, to give users to manage their personal profile and their personal application project. We automatically generate the contents of the struts-config.xml file and the stubs of the action classes from the diagram created using the model. The main advantages which the tool will grant to the developer are:

- A simple and quick design of the application model;
- A good overall vision of the application, with a notation directly referring to the MVC models;
- An independent design of the controller flow and of the called action;
- An easy reuse of parts of the application defined in the diagram.

The tool will be realized for an online usage and based on RIA technologies. A Rich Internet Application (RIA) is an Internet application that attempts to bridge the usability gap between desktop applications and traditional Web sites. At this purpose, it exploits the graphical and execution capabilities of the clients (namely, the browsers), by adding scripts into the pages, thus providing higher quality of interactivity and an user experience. The most known platform for RIAs is the AJAX framework, that we adopted for our implementation. In particular, we based the GUI of the tool on the opensource Openjacob Draw2D library and other AJAX technologies. In the 5.6 Software requirements and system architecture section we will analyze in depth the architecture and all the technologies used.
5.3. Users groups and site-views

Figure 5.3 shows the groups of users of the MVC-WebFlow tool, organized in a hierarchy. The tool is accessible only to registered users and obviously to the administrator user.

**Registered user:** that is the designer, the developer of Web applications. The tool is a support instrument which helps the user during the developing phase of an application. A user has a personal profile with info like first name, last name, email, username and password. The site-view of the registered user is the graphic interface of the tool from which users can also access their own account to manage personal data and projects. This category of use can access only to its saved projects and public project (each user is able to make public a personal project).

**Administrator:** that manage the users account and make control on project data. The site-view of the administrator is the same of the registered user to allow him to test the MVC-WebFlow tool working conditions and to manage and control all the registered users accounts. The administrator have not restriction on the accessible data.

![Figure 5.3 – Users groups](image)

5.4. User requirements

In this section we will describe the user requirements overlooking the administrator requirements which are the same of a registered user regarding the effective use of the graphic functionalities of the tool. In addition the administrator can manage users accounts but it's a less relevant aspect not interesting for our work.

5.4.1. Functional requirements and use case view

The graphic editor must allow the developer to model the structure and the flow of the application and to define in detail the behaviour of the controller but also the view of the application, that is the set of pages that form the application. The definition of the controller and
the view needs the possibility for the user to edit each component of the diagram, in particular Action units and Form units.

During the first step of the problem and domain analysis we have identified a set of functional requirements that the editor will have to respect:

- Create a personal account
- Login
- Modify the personal account info
- Modify personal saved projects
- Delete personal saved projects
- Save the diagram model in XML
- Open a diagram model in XML
- Create a new empty project
- Zoom in / Zoom out
- Change the model view (data or hypertext)
- Edit the diagram model of the application
- Add a unit to the work area
- Delete a unit from the work area
- Edit a unit in the work area
- See properties of a unit in the work area
- Add a property field to a Form unit
- Remove a property field from a Form unit
- Edit a property field
- Add a Link (to connect units)
- Delete a Link
- Build and download the application files

In the following picture we can see the general use case diagram of the system which represents all the main functionalities of the editor.
The less relevant aspect of the application is the management of the account personal info. The tool gives the user the possibility to create and manage a personal account saved on the server. With the login, the user is identified by the system, which makes available his saved XML projects and also public projects.
The state diagram in Figure shows all the possible states in which the user can be during the interaction with the tool. With this diagram we can better understand the working mechanism of the editor. To make it easier to see and simpler I have overlooked some aspects of the interaction. The arcs represent the possible actions of the user and the squares are the states in which the system can be. The end point is missing because the user has the possibility, in every moment, to close the application and logout. The user can also save at any time the XML project on the server.

With the model palette of the tool the user can add elements to the model diagram, elements are Action units, Form units, JSP page units and JSP fragment units. Links are created dragging the output port of the source unit on the input port of the target unit. The connection mechanism work through ports connections. Each unit has an Input/Output port, at the center of the figure. When the user select an object the tool will have to underline which other objects can be connected to it. A link can be connected from an object port to another object port. For example an Action unit can be connected with another Action unit but also with a JSP page unit. A Form unit can be connected only with an Action unit. An Action unit must have a Success link in
output; the first time you connect a source Action unit with another target unit the link will be a Success link, which represents the success path needed for the controller configuration file. The second link will be a Failure link, which represents the failure path of the Action unit. If there is no Failure link the target unit, connected to the Success link, will be considered both the success path and the failure path.

![Diagram](image1)

**Figure 5.7 – Use case “Add an element”**

The user can remove a unit or a link selecting it and cancelling or using the undo / redo function of the tool.

![Diagram](image2)

**Figure 5.8 – Use case “Delete an element”**

For each unit the developer can set a list of properties which will go to define the controller configuration file of the application. These properties are managed using the Property panel of the tool. As regards the Form unit we have to specify that the user can define the own properties
of the unit but also add and modify the property fields of the FormBean generated with the controller configuration file, for example in a login form we can found username and password fields.

Selecting an element the user can see all its properties. Properties are of two types. Figure properties are the properties of the object in the diagram such as position, identifier, name. Model properties are the specific properties of the unit, these properties will be used to define the controller configuration file of the application and are visualized and make editable using the Property panel of the tool.
A small example of user interaction with the tool is showed in the following Sequence diagram. In this example the user make the login, opens a saved project, edits some elements of the diagram and saves the project.

Figure 5.11 – Sequence diagram of a typical user interaction

5.4.2. Non-functional requirements

The editor will be realized as a Web application usable from any Web browser. The javascript libraries used provide all the elements to manage the diagram in the browser. The editor will must respect some non-functional requirements.

- To allow the user to make only allowed operations
- To be simple to use and usable
- To be simple and light for the browser
- To be cross-browser (in particular for Internet Explorer and Firefox)
- To have a high level of maintainability and integrability with other solutions
5.5. Interface requirements

The graphic style of the interface is based on the GUI Graphical User Interface of editors tool. The goal of MVC-WebFlow interface is to enhance the efficiency and ease of use for the user, to improve the interactivity and to give the tool the maximum usability. Techniques of user-centered design will be used to ensure that the visual language introduced in the design is well tailored to the tasks it must perform.

The graphic interface of the tool will be composed by:

- A **Logo** section at the left top of the web page
- A **Project panel** at the center top of the page: that is a menu bar to access all the functionalities of the tool, open a project, save the current project, create a new empty project, zoom in and zoom out, build the application modeled files and manage the personal profile.
- A **Work area** on the rest of the page: that is the workflow area in which the developer will go to model the application flow creating the MVC-WebFlow diagram model.
- A **Model palette** on the left side of the page: that is the objects palette with which the user can add units in the work area to model the diagram.
- A **Property window** which shows the workflow property of the selected object in the work area, like the position or the dimension.
- A draggable **Property panel** on the right side of the page: that is the most important part of the interface which allow the developer to define all the state properties of the unit selected in the work area, for example the name and a path of a JSP page or the text fields of form.

In the following figure we can see a simple sketch of the main interface of MVC-WebFlow tool.

![Figure 5.12 – Sketch of the graphic interface](image-url)
The application must have a Login section and with the same style some pop-ups to create an account, manage the account and download the generated files of the modeled application.

5.6. Software requirements and system architecture

The realization of this tool asks for a software solution which support a set of functionalities needed for a good implementation. We have identified a set of software requirements listed above.

- Dynamic creation of the diagram inside the browser (editor)
- Zoom function
- Drag & drop function
- Pan function
- Undo & Redo function
- XML support
- Dynamic editing of objects with windows or dropdown menu
- Possibility to set objects properties
- Platform independent and possibility of integration with other solutions
- Possibility to manage more than one model, for example database and hypertext
- Possibility to interact with the backend with java objects
- Cross-browser compatibility
- Open-source license

To fulfill this software requirements we have opted for the software choices and the system architecture illustrated in the following paragraph.

The architecture of the application has been based on RIA technologies. A Rich Internet Application (RIA) is an Internet application that attempts to bridge the usability gap between native applications and normal Internet ones. It contains more code on the browser, which offers higher levels of interactivity and an experience similar to native applications. With RIAs, it's possible to use many technologies, such as Java and ActiveX, but the most important one is JavaScript. Because JavaScript is provided directly by the browser instead of being an add-on like the other technologies, you can get the most benefit from the least amount of work. One of the driving technologies behind RIA in the JavaScript language is a technology called AJAX. AJAX offers the ability to communicate with your Web server outside of the normal load flow.
We can see in depth all the technologies that are used to build the Web editor tool:

- HTML
- Javascript and DHTML
- W3C DOM
- OpenJacob Draw2D javascript library
- Yahoo! User Interface library (UI)
- AJAX XMLHttpRequest
- Tomcat 5.5.17 sever
- Server-side scripting (Java server pages, Java Servlet) and Java classes
- XML objects
- MySQL database

*Client-side environment*

At client side, HTML provides the frame of the editor Web page. The graphic interface is entirely implemented at client side with Javascript. The event model (DHMTL) allows Javascript to respond to events such as mouse click; this is the key to make interactive the interface. The modelling canvas is implemented by the Openjacob Draw2D library, that in turns exploits JavaScript and DOM. To support the interaction with the user and to build the editor panels and windows we have used the Yahoo! User Interface library, a set of utilities and controls written in JavaScript for building richly interactive Web applications using techniques such as DOM scripting, DHTML and AJAX. User interactions at the interface level trigger events (1) that in turn activate the interaction with the backend components (2) at the server side.

*Server-side environment*

To interact with the server-side of the application and call server objects the editor uses the AJAX technologies and in particular the XMLHttpRequest object (3). This object allows to send and receive HTTP requests and responses to and from web servers. The XMLHttpRequest class is what makes true AJAX application development possible. The HTTP requests made with XMLHttpRequest work just as if the browser were making normal requests to load a page or submit a form, but without the user ever having to leave the currently loaded web page. AJAX technologies build an asynchronous system against the traditional approach which is based on synchronous calls. Traditional web apps use a synchronous system: you must wait for the server to send you the first page of a system before you can request the second page. With XMLHttpRequest the client has the ability to make lots of small requests to the server and the page is dynamically updated by the server response. The end result is an application that feels much more responsive, as users spend significantly less time waiting for requests to process, and
don’t have to wait for an entire new web page to come across the wire, and be rendered by their browsers, before they can view the results. All the requests are collected by Java server pages on the server-side and each of them make the proper AJAX response (5). The editor allows the user to save its own project data and to load it at the login. At the moment of the saving of the data the diagram created in the work area is converted in a XML string and saved in a MySQL database on the application server. When the user wants to load the project the editor makes a parsing of the XML string and generates all the figure objects and connections of the diagram. For the code generation part the client will invoke a Java Servlet which has the aim of creating the needed objects (Server Java classes) to generate all the files of the Web application modelled with the tool. The server used to contain the application is the Bundled Tomcat 5.5.17 server. The following figure shows the overall architecture of the system.

![Diagram](image)

Figure 5.13 – Architecture of the Web editor system
5.7. Data schema design

The data schema is very simple and aims only at saving the user account personal information, the membership group name and projects saved data. In particular we can identify two different structures in the ER schema:

- **Core sub-schema**, Core Objects which are the most important items that the application must handle; a particular attention must be put on two attribute:
  - **XML_string**: is the XML descriptor string of the MVC-WebFlow diagram designed with the graphic tool, this string contains all units and their properties and links.
  - **public_project**: indicates that the project is public, that is the project can be open in read mode by other users in addition to the owner.

- **Personalization sub-schema**, Personalization Objects Used to embed relevant user data in the model; the **role-name** attribute represents the user group which can be Administrator or User and it also aims at realizing the Login validation during the access at the tool.

From this ER diagram we can derive the Logic schema of the database which will be created to store user data and project data.

user_role (user_name, role_name)

user (user_name, user_password, first_name, last_name, email_address)

project (name, XML_string, public_project, user_name)
5.8. Logical view and components view

In this section we are going to illustrate the Logical view and the Components view of the MCV-WebFlow tool. To reach this goal we will use the WAE Web Application Extension for UML. This extension enables us to represent the web pages and other elements of the model, from a significant architectural point of view, with the normal classes of the model. In addition to WAE class and association stereotypes we have introduced some new stereotypes to better describe the system model.

5.8.1. Logical view

Using the stereotypes defined below we are able to model the basilar architecture of the MVC-WebFlow application. The diagram in Figure represents the application flow and give a complete view of the application structure.

Classes stereotypes:

- <<client page>>: an html page showed by the Web browser which can contain client scripts;
- <<html form>>: a set of input fields which are a section of a client page;
- <<server page>>: it contains server-side scripts and interacts with the business logic and the database;
- <<servlet>>: a Java servlet invoked by a client page;
- <<script library>>: it defines a set of Javascript variables and functions (or a set of .js files);
- <<server resource>>: a generic server resource (like the Tomcat validation environment);
- <<java class>>: a Java object used at server-side to realize business logic.

Association stereotypes:

- <<link>>: it’s a connection between a client page and server resources or other pages;
- <<Ajax request>>: it represents an Ajax request of a client page to a server page or a servlet;
- <<Ajax response>>: it represents an Ajax response from a server page or a servlet to the client page that make the Ajax request or to another server resource;
- <<submit>>: a directional relation between an html form module and a server page;
- <<build>>: a directional relation between a server page and the client page that represents the html output;
- <<script>>: a Javascript inclusion in the client page;
- <<use>>: it indicates the usage of server-side resources like Java classes.
Figure 5.15 – Logical view of the application
All the logic dedicated to the graphical creation of the diagram is on the client-side of the tool. The MVC-WebFlow Javascript library is composed by all the .js files, functions and variables which exploit the Draw2D and the Yahoo! YUI libraries to realize the graphic interface of the tool. The libraries are enough complex and going into the details of all their functionalities would be heavy and useless to describe the general structure.

The server-side components are quite simple and they have the hard task to take the XML string descriptor of the project saved by the client application, to parse it and to generate the corresponding Web application structure and code, based on the Jakarta Struts framework. For the moment the generation is only partial and this components make the basilar structure of the application. The application structure will be composed of two directory, one with the web application files (configuration files, JSP pages, ...) and the other with all the necessary Jakarta Struts files (to put in the parent folder of the application directory). In the near figure we can see the tree structure of the application created that will be available for the download, in zip format, after the building.

We can analyze in details the content of each of this folders to see the real composition of the Web application that the tool will generate starting from the project-descriptor.xml file, which is the configuration file of the MVC-WebFlow diagram saved by the client application.

- **project name** folder is the main folder and contains:
  - **JSP pages** (the view of the application) and project-descriptor.xml file
  - **WEB-INF** folder is the private folder with the configuration files for a struts application, in particular it contains:
    - The struts-config.xml file, the web.xml file, and the validation.xml file if the application requests the struts framework validation
    - **classes** folder: with the compiled Java class files
    - **src** folder: with the source Java class files to be compiled, it contains the Action classes corresponding to the Action unit of the diagram
  - **struts** folder is the folder dedicated to host all the files to make Jakarta Struts working:
    - **dtd** folder: with the Document Type Definition files
    - **lib** folder: with the .jar files of the struts library
    - **taglib** folder: with all the Tag Library Descriptor files usable in the view of the application like the struts-html.xml or the struts-html-el.xml
    - **validator** folder: it contains the validators-rules.xml file, with the validation struts rules, and a trace of validation.xml file to use for the generation of the personal validation file of the application

The packages defined in the server source resources are the ApplicationFiles package, the Util package and in addition the Jdom package, a library package needed for the XML parsing.
**ApplicationFiles package**

This package contains the Java servlet invoked by the client application to generate the Web application code and all the Java classes needed to fulfill this task.

**Util package**

The Util package contains some utilities for the zipping of the application generated directories and for their deletion when the user makes the download of the files.

**Jdom package**

Jdom is a complete Java based library for accessing, manipulating and outputting XML data from Java code.

5.8.2. Components view

The components view of the application model describes the modules which form the executable system. The stereotypes used below allows to keep all the relevant properties of the components.

**Components stereotypes:**

- `<<Web page>>`: static page accessible from the Web browser
- `<<JSP page>>`: dynamic page accessible from the Web browser
- `<<Script library>>`: implements one or more Script library classes of the logical view

![Components view of the application](image-url)

*Figure 5.16 – Components view of the application*
5.9. Tool interface and sample application

In this paragraph we will show the Editor tool interface. To access the tool the user has to be registered in the system and have a personal account. The homepage contains a simple form panel to enter the application.

![Figure 5.17 – Homepage with the access form panel](image_url)

In figure 5.18 you can see the complete GUI of the tool. The interface is formed by two main areas: the project work area and the project management area. In the project work area we can find at the left the **Model palette**, used to add elements to the work area and to use the undo / redo functionality, while at the right we can see the **Property panel**, which is the panel used by the developer to set all the properties of the diagram element selected. When the user change the element selection on the area the **Property panel** will show its own properties. In the center of the project work area there is the diagram zone, the area in which the developer create and manage the flow diagram of the application to generate. In the **Model palette** the user cannot find link elements because a link must be generated making a connection between two elements dragging the output port of an element to the correct input port of another element. The project management area allows to manage the project. With two tabs the user can choose between the **Data model**, which contains the data structure of the application, and the **Web model** (the Hypertext model), which regards the definition of the conceptual schema of the application and
its navigation flow. At the top the tool shows the **Project panel**; with this panel the user can open a saved project 📁, save the current project on the server 🌐, create a new empty project 🗿, use the zoom in 🚶️‍♂️ and zoom out 🚶️‍♂️ functionalities (not yet implemented), build the application files 🗺️ and manage his personal profile ⬨.

![Figure 5.18 – Homepage with the access form panel](image)

Accessing personal profile management ⬨, the user can modify all personal account info, such as first name, last name and password, but also delete all saved projects or setting them as public projects. Figure 4.19 shows the profile management page.
To show the approach at work, figure 5.20 presents a very simple piece of Web application specified with MVC-WebFlow and edited within the MVC-Webflow online editor. The application is a credit card validator which receives in input from the user a credit card number and makes its validation generating a confirmation or a refusal. The example diagram shows the JSP page `creditCardCheck`, that leads to the `checkCard` Action that includes the form `creditCardForm`. The successful outcome of the `checkCard` Action leads to the JSP page `confirmed`, while a failure leads to the JSP page `notConfirmed`. 
The developer has to define each element of the diagram setting its own specific properties showed in the Property panel. These properties will be saved in the XML description string for the diagram and are necessary for the generation of the controller configuration file of Jakarta Struts. The business logic of the application is not managed by the tool and must be completely realized by the developer. In the **Execute method panel** the developer has to instantiate all the needed business objects. Later, in this paragraph, we will illustrate a sample Struts-config.xml configuration file for the credit card validation application. In the following figures you can see some cases of element properties editing, such as the **creditCardForm** properties and fields creation or the **checkCard** Action with the **Execute method panel**.

![Property Panel](image1.png)

**Figure 5.21 – creditCardForm** properties and form fields creation
In these rows you can see the Struts-config.xml configuration file for the credit card application. The underlined code refers to those elements generated starting from the flow diagram representation and the objects properties edited. In red we have everything concerning the *creditCardForm*, while in blue we have the action-mappings elements which define the real web flow of the application (derived from the *checkCard* Action).

```xml
<struts-config>

  <form-beans>
    <form-bean name="creditCardForm" type="org.apache.struts.validator.DynaValidatorForm">
      <form-property name="number" type="java.lang.String"/>
      <form-property name="owner" type="java.lang.String"/>
      <form-property name="validFrom" type="java.lang.String"/>
      <form-property name="validUntil" type="java.lang.String"/>
    </form-bean>
  </form-beans>

  <action-mappings>
    <action path="/checkCard" type="package.CheckCreditCardAction" name="creditCardForm" scope="request">
      <forward name="success" path="/confirmed.jsp"/>
      <forward name="failure" path="/notConfirmed.jsp"/>
    </action>
  </action-mappings>

  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <setProperty property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
  </plug-in>

</struts-config>
```
6. Conclusions and future development

In this project we have proposed a light-weight design methodology that leads to the development of MVC applications. With this project we have presented an on-line visual editing tool called MVC-Webflow for the specification of simple conceptual models for MVC applications and we provide partial automatic code generation, that can be performed on the flight directly on the deployed application.

The tool is based on a visual model for easily describing the controller behaviour and the used action classes. To achieve this goal we have developed a metamodel defined by a set of primitives and by the rules for specifying the connections among those primitives. The diagram resulting from the design phase represents the conceptual schema of the MVC application. With this model and using the tool the developer is able to define the data structure and the flow of the application, the view composed by the JSP pages and the controller which process the user requests and acts as intermediary between the business logic and the view.

Our project has allowed us to try a new approach to support the developer during the design of Web applications. The advantages of the approach are the closeness to the well known MVC paradigm, the foundation on solid web engineering models, and the reduction of the duration of development cycle.

The tool is still under development and the future works will deal with this two main aspects:

- The implementation of the data view of the graphic tool which will allow to model the data structure of the application;
- For the moment the generation is partial, the tool generates the basilar structure of the application and allow the user to download it. The application structure is composed of two directory, one with the web application files (configuration files, JSP pages, ...) and the other with all the necessary Jakarta Struts files (to put in the parent folder of the application directory). The generation logic will have to be completed and refined, the goal will be to compile the generated file and run the application on the fly

Demo for The Eighth International Conference on Web Engineering

The Eighth International Conference on Web Engineering (ICWE2008) will be held in July 2008 in Yorktown Heights, New York, USA. ICWE2008 aims at promoting research and scientific excellence on Web Engineering and at bringing together practitioners, scientists, and researchers interested in technologies, methodologies, tools, and techniques used to develop and maintain Web-based applications, leading to better applications and systems, and thus to enabling and improving the dissemination and use of content and services through the Web.
During the MVC-WebFlow tool development Professor Marco Brambilla and I we have worked at a demo article to be submitted to the conference. The demo paper, entitled “MVC-Webflow: an AJAX Tool for Online Modeling of MVC-2 Web Applications”, has already been delivered and accepted by the review commission with good rating. The paper will be corrected referring to reviewer comments, and trying to incorporate them in the final version.
7. Appendix A: Ajax open source and free solutions

7.1. JavaScript Multipurpose frameworks

7.1.1. DOJO

Dojo is an Open Source DHTML toolkit written in JavaScript. It builds on several contributed code bases (nWidgets, Burstlib, f(m)), which is why we refer to it sometimes as a "unified" toolkit. Dojo aims to solve some long-standing historical problems with DHTML which prevented mass adoption of dynamic web application development. Dojo allows you to easily build dynamic capabilities into web pages and any other environment that supports JavaScript sanely. You can use the components that Dojo provides to make your web sites more usable, responsive, and functional.

Dojo offers comprehensive widget and browser-server messaging support:
- Extensive deployment support such as dependency-based packaging and compression of required libraries into a single download
- Framework for creation of custom JavaScript widgets
- Library of pre-built widgets
- Solid drag-and-drop, effects, and generic animation support
- Browser-server messaging support such as XMLHttpRequest and other mechanisms
- Event management
- Support for bookmarkability and manipulating URLs in the browser
- Open source license (Academic Free License 2.1); led by Alex Russell of JotSpot

http://dojotoolkit.org/

7.1.2. Engine for Web Applications

Engine for Web Applications is an application framework for client-side development and Web applications. It provides an environment in which to develop and run JavaScript components and applications. Each Engine can be pre-built to user specifications, and components can be built into the engine itself, deployed as separate script packages, or deployed as application components. The Engine code is designed within a hierarchical structure, and can be used with most existing JavaScript toolkits and libraries. Engine is based on a structured API, a core set of background services, and an optional abstraction layer. Engine for Web Applications encapsulates several services and utilities: managing application interdependencies and bootstrapping (tasking), messaging, transactions, configuration, dynamic runtime components, and monitoring. The engine service, from which the product gets its namesake, connects the other services and utilities to create an abstraction layer for associating script with content and exposing declarative scripting with XML.

Engine for Web Applications is a framework for building modular components for web applications and separating content from functionality:
- Configurable deployment
- Registry for storing, retrieving, and destroying objects that conform to the common interface
- Message service for intraobject publications and subscriptions
- Transaction service for asynchronous multiobject communication
✓ Task service for bootstrapping client-side modules of a web application and for creating dependency-based task lists; can load an XML-based task.
✓ Ability to easily create scoped and redistributable components in XML with automatic event hook-up and containment
✓ Store and retrieve form field values regardless of whether the form field still exists
✓ Monitor service for easily creating beacon and Ajax monitoring solutions
✓ API Documentation
✓ Demonstrations with source
✓ Open source license (custom license) by Stephen W. Cote

http://www.imnmotion.com/projects/engine/

7.1.3. Freja

Freja is a specialized Javascript Framework for creating template-based, single-screen web applications. Freja is a javascript MVC (Model-View-Controller) framework. Freja stands for Framework for REstful Javascript Applications. It offers the following benefits:

✓ Browser-centric and server-agnostic.
✓ Geared toward single-screen, zero-latency web applications.
✓ Excellent scalability (much less server round-trips).
✓ Based on recognized standards and accepted best practices:
  o XML, XSL Templates and XPATH.
  o Separation of data, logic and presentation (MVC pattern)
✓ True Server-side / Client-side code separation.
✓ Very light footprint. Only a handful of methods to learn (the framework is 25kb uncompressed).
✓ Based on Open Standards (XML/XSLT).
✓ Works with any modern web browser: IE, Firefox, Safari, Opera.
✓ Plays well with other javascript toolkits and libraries (prototype, scriptaculous, dojo, etc.). Freja is not "yet another Ajax library", it is a Ajax-based high-level framework. It is not its purpose to provide drag&drop functionality or visual effects, but it will let you use your favorite javascript library to do so.
✓ Tutorials and Documentation available
✓ Open-source and licensed under CC-GPL.

http://formassembly.com/blog/the-freja-framework/

7.1.4. Gravey

Gravey is a framework for developing Rich Internet Applications using standard design patterns like MVC, Command, AJAX, and REST. It is distinct in being a completely stand-alone JavaScript library, and hence, the Gravey AJAX framework is compatible with any server-side platform (e.g. Java, .NET, PHP, Coldfusion). It’s is a very small set of JavaScript code and programming conventions that together emulate classical classes of the sort familiar to C++ and, in particular, Java programmers. This platform is under constant development as it is the basis for several enterprise Banking applications.

✓ Grv object approach to declare classes with Javascript
✓ Traditional Javascript allowed
✓ Object property lookup and storage more efficient
Avoid traditional approaches problems of creating multiple copies of class prototypes
Cross browser compatibility
Eclipse and WSAD friendly, compatible with JSDoc
Js documentation
Examples and technical articles available
Source code of Gravey and example applications
Open-source license
http://www.polyglotinc.com/Gravey/

7.1.5. Javeline PlatForm

Javeline PlatForm is a software development kit for building next-generation applications. With Javeline PlatForm developers can deliver applications to modern browsers on all platforms, in shorter development cycles and with more functionality. Javeline PlatForm outperforms other toolkits in two key aspects, development time and user functionality. PlatForm introduces you to writing Rich Internet Applications (RIA) using Ajax. The advantages of Ajax and Javeline's declarative development enable you to focus on the end result you want your programs to achieve, rather than the "how" in building them. The platform is flexible, user-friendly, and has support such as programming examples, tutorials, and documentation. Platform uses JavaScript and HTML/CSS, enabling you to easily adjust to using these tools with existing web architectures and desktop applications.

About Javeline technologies and features:

- TelePort: is the part of Javeline PlatForm that actually makes it 'Ajax'. TelePort is the Asynchronous communication library that uses JavaScript And Xml (Ajax) to communicate to a server without refreshing the page.
- SmartBindings: is a development method created by Javeline to leverage existing technologies and methodologies into the world of Web 2.0 and beyond. It couples tried-and-true Xpath-powered databinding with the exciting new programming languages and techniques that today's Internet offers. Data binding is the act of interactively connecting a piece of data to a user-interface. SmartBindings can facilitate data binding both between data and the user-interface and also within the user interface itself.
- FlowML: Javeline FlowML is a multimedia system tightly integrated in the Javeline Product line. Manipulating video streaming, video encoding, visual effect processing and computer vision, it can be used as an integral part of Javeline DeskRun and PlatForm driven applications.
- W3C standard compliant.
- Opensource: the core of PlatForm is released under the GPL license
- Write Less Code
- Create User-Friendly Applications
- Quickly Adapt when the spec changes
- Good documentation

http://www.javeline.com/

7.1.6. MochiKit

MochiKit is a highly documented and well tested, suite of JavaScript libraries that will help you get your work done, fast. Authors borrowed good ideas from Python, Objective-C, etc. and adapted them to JavaScript.

Its features:
MochiKit.Async - manage asynchronous tasks
MochiKit.Base - functional programming and useful comparisons
MochiKit.DOM - painless DOM manipulation API
MochiKit.Color - color abstraction with CSS3 support
MochiKit.DateTime – time management
MochiKit.Format - string formatting
MochiKit.Iter - itertools for JavaScript
MochiKit.Logging – logging management
MochiKit.LoggingPane - interactive MochiKit.Logging pane
MochiKit.Visual - visual effects
Reliable: hundreds of tests, objects to reuse
Documented: 100% documentation covered
Evolutionary: it can adapt to anything you throw at it
Plays well with others: it is very highly interoperable (Jsan, DOJO, …)
Compatibility with all of the modern and popular browsers
MIT license or Academic Free License, v2.1.

http://mochikit.com/

7.1.7. Mootools

MooTools is a compact, modular, Object-Oriented JavaScript framework designed for the intermediate to advanced JavaScript developer. It allows you to write powerful, flexible, and cross-browser code with its elegant, well documented, and coherent API.

Features:

✓ It is lightweight, powerful and makes use of many of the new javascript coding functions that we've become accustomed to (such as $()).
✓ The FX library is incredibly solid, doesn't require div-itis, and uses the same physics easing equations as most flash users.
✓ Browser Compatibility -mootools is compatible and fully tested with Safari, internet explorer 6 and 7, Firefox (and its mozilla friends), Opera and Camino.
✓ Class is slightly based on Base.js http://dean.edwards.name/weblog/2006/03/base/ © 2006 Dean Edwards, License http://creativecommons.org/licenses/LGPL/2.1/
✓ Some functions are inspired by those found in prototype.js http://prototype.conio.net/ © 2005 Sam Stephenson sam [at] conio [dot] net, MIT-style license
✓ Js documentation by Aaron Newton and Valerio Proietti
✓ Great support: blog, demos, code, technical articles
✓ License - mooTools is released under the Open Source MIT license, which permits you to use it and modify it in every circumstance.

http://www.mootools.net/

7.1.8. Open Rico

Open Rico is a multi-purpose framework with support for Ajax infrastructure and user interaction. Rico provides a very simple interface for registering Ajax request handlers as well as HTML elements or JavaScript objects as Ajax response objects. Multiple elements and/or objects may
be updated as the result of one Ajax request. With Rico you can realize animation effects, styling, drag and drop.

- An XMLHttpRequest response can be routed to one or more callback operation, DOM object, or Javascript object.
- Easy drag-and-drop.
- Ajax animation such as scaling and transitions (and presumably the increasingly common idioms such as progress indicators and fading technique)
- "Behaviors" - Essentially a widget library.
- External tutorial by Yonah Russ of Mirimar
- Builds on prototype library.
- Rico is provided free and open-source (Apache 2.0 License) for either your personal or commercial use. From Sabre Airline Solutions. By Bill Scott, Darren James, and others.

http://openrico.org/

7.1.9. OpenLink AJAX Toolkit (OAT)

OAT, OpenLink AJAX Tools, is a Javascript framework for building rich browser-based user interfaces using XML/A to connect to databases. It includes a rich collection of UI Widgets/Controls, Event Management System, and a truly platform independent Data Access Layer called AJAX Database Connectivity (ADBC).

- Widgets include Bar Chart, Pie Chart, Line Chart & Sparklines, Grid, Pivot table, Tree, Dock, Panelbar, Ticker, FishEye, WebDAV Browser, Timeline, RDF Graph, Variable upload, Input restrictions, Dimmer, QuickEdit, Combo list, Combo box, Combo button, Slider, Date picker, Color picker, Enhanced Anchor, and an RSS Reader.
- Libraries include JSON, Cryptography, Statistics, AJAX, Ghost Drag, OS-style Windows, Rounded Corners
- AJAX Database Connectivity, or ADBC, enables and richens Mapping Mashups, Pivot Tableless, Scrollable Cursors, and Data Modelling, through trivally easy binding of widgets to SQL (via XMLA), RDF (SPARQL), XML (HTTP), and Web Services (both SOAP & REST).
- Usage is as simple as including two .js files in the <head> portion of an HTML document, and specifying a list of features you want to use. The framework will build the dependency tree and dynamically include all needed (and no other!) files, minimizing client payload size.
- OAT has been tested in Gecko browsers (Firefox, Mozilla/SeaMonkey), IE6, Opera and Apple WebKit.
- OpenAjax Compliant (as of March, 2007)
- Fully functional online demo
- Full documentation also online
- Open source, released under GPLv2, found on SourceForge

http://sourceforge.net/projects/oat

7.1.10. Plex Toolkit (Open Source RIA Framework and GUI Toolkit)

Plex Toolkit is an open source feature-complete DHTML GUI toolkit and AJAX framework based on a pure Javascript/DOM GUI toolit and AJAX framework. Uses the almost identical markup language to PXML (Flex) embedded in ordinary HTML documents for describing the UI. Binding is done with Javascript. Plex is a cross browser toolkit for DHTML programming, and runs on Opera-8.5, Konqueror, Firefox-1.5 and IE6. Plex is a framework for building AJAX graphical user interface (GUI) toolkits.
Features:

- Full set of widgets such as datagrid, tree, accordion, pulldown menus, DHTML window manager, viewstack and more (over 60 libraries)
- Markup driven (makes it easy to build the interface)
- Interface components can be easily themed with CSS
- Client side XSLT for IE and Mozilla
- Well documented with examples.
- Multiple remoting transport options - XMLHttpRequest, IFrame (RSLite cookie based coming soon)
- Back button support (complete history management for all components)
- Support for YAML serialization
- Open-source under GPL, LGPL, or Artistic License (your choice) by Richard Hundt.
- No animation framework

http://www.plextk.org/trac/wiki/

7.1.11. Prototype

Prototype is a JavaScript Framework that aims to ease development of dynamic web applications. It’s a easy-to-use toolkit for class-driven development and uses the nicest Ajax library around. Prototype makes it easy to use object-oriented concepts like classes and inheritance within Javascript. It also supports basic Ajax functionality such as web remoting. Prototype framework enables you to deal with Ajax calls in a very easy way, safe and cross-browser. Besides simple requests, this module also deals in a smart way with JavaScript code returned from a server and provides helper classes for polling.

- A project run in conjunction with Ruby on Rails, but can be (and certainly is) used independent of Ruby or RoR.
- Easy management of Ajax calls
- DOM extensions is the core of Prototype
- JSON data-interchange format
- Animation and effects can be added on using the Scriptaculous library
- API docs, tips and tutorials, blog and other support
- Open-source by Sam Stephenson.

http://www.prototypejs.org/

7.1.12. qooxdoo

Qooxdoo is an innovative Open Source (EPL/LGPL) multipurpose AJAX frameworks. It includes support for professional JavaScript development, a state-of-the-art GUI toolkit and high-level client-server communication.

Features:

- pure JavaScript framework
- entirely class-based
- fully based on namespaces
- easy integration with other libraries
- comprehensive API reference
complete JavaScript parser which optimizes, compresses and makes easy the
development of user-friendly applications

- easy implement of user interfaces with a full-blown set of widgets
- no HTML and CSS must be known and used
- client-side and implementation of RPC servers
- an abstract layer supports queus, timeouts and implementation via XMLHttpRequest, Iframes and Scripts
- Portable abstraction of the DOM and event/focus management
- Debugging support
- Timer class for easy scheduling
- Getter/Setter support
- Widget framework and library of prebuilt widgets
- Layout managers
- Image caching and portable PNG transparency
- Wide documentation and community support
- Open source license (LGPL) by various contributors

http://qooxdoo.org/

7.1.13. ThyAPI

ThyAPI is a library of DHTML reusable components. AJAX is one of its features. Its main purpose is to allow the creation of full desktop-like applications in browser. ThyAPI is a library based on DynAPI, and Open-source (LGPL).

- Made entirely in DHTML (Javascript + CSS)
- Totally object-oriented, extensible
- XMLRPC protocol, for data exchange with the server. Plans to implement JSON-RPC.
- Interface definition entirely in CSS
- Working in Mozilla Firefox (version 1.0+) and Internet Explorer (version 6+)
- A DataSource component, to simplify linking of javascript widgets with server methods
- Cross-browser, cross-platform support (it inherits this characteristic from DynAPI)
- Aimed to provide RAD development of browser-based applications, and reuse of custom-made widgets
- Open Source (LGPL)

http://sourceforge.net/projects/thyapi/

7.1.14. Yahoo! User Interface Library (YUI)

The Yahoo! User Interface (YUI) Library is a set of utilities and controls, written in JavaScript, for building richly interactive web applications using techniques such as DOM scripting, DHTML and AJAX. The YUI Library also includes several core CSS resources. All components in the YUI Library have been released as open source under a BSD license and are free for all uses.

Features:

- Global Object for all the components
✓ Methods collection for DOM interactions
✓ Events normalization and tools
✓ YUI library utilities: animation, browser history, connection manager, datasource, drag and drop, image loader, YUI loader and test
✓ YUI library controls/widgets: autocomplete, button control, calendar, color picker, container (panels, windows and other), datatable, logger, menu, rich text editor, slider, tabview, treeview
✓ YUI library for CSS management
✓ API documentation
✓ Support and material, technical articles and tutorials
✓ Liberal BSD license

http://developer.yahoo.com/yui/

7.1.15. Zimbra AjaxTK (Commercial-OpenSource RIA Toolkit)

Zimbra is a recently released client/server open source email system. Buried deep within this product is an excellent Ajax Tool Kit component library (AjaxTK) written in Javascript. A very large and comprehensive widget library is now available to the open source community.

Features:

✓ Full support of drag and drop in all widgets. Widgets include data list, wizard, button, text node, rich text editor, tree, menus, etc.
✓ Build system uses Ant and hosting is based on JSP and Tomcat.
✓ Very strong client-side MVC architecture based; architect is ex-Javasoft lead developer.
✓ Communications support for client-side SOAP, and XMLHttpRequest? as well as iframes.
✓ Support for JSON serialized objects, and Javascript-based XForms.
✓ Strong multi-browser capabilities: IE 5.5+, Firefox 1.0+, latest Safari
✓ Hi quality widgets have commercial quality since this is a commercial open source product.
✓ Widget library is available as a separate build target set from the main product.
✓ Debugging facility is built in to library, displays communications request and response.
✓ License terms making it suitable for inclusion in other commercial products free of charge.
✓ Does not currently support: Keyboard commands in menus, in-place datasheet editing.
✓ Does not support graceful degradation to iframes if other transports unavailable.
✓ Documentation is lacking, but PDF white paper describing widget set and drag and drop is available.
✓ Open-Source License (Zimbra Ajax Public License ZAPL, derived from Mozilla Public License MPL) from Zimbra.
7.2. JavaScript Remoting frameworks

7.2.1. Juice Lib

Juice Lib is a simple javascript library providing an useful set of functions and object that will enhance web user interface development.

Features:

✓ Text manipulation and string buffers.
✓ Map, Reduce and Filter for array data processing.
✓ Associative arrays utilities.
✓ Simple log system.
✓ onload and keyboard event handlers.
✓ HTTPXMLRequest processing (simple request and scheduler).
✓ Apache 2.0 License.

http://code.google.com/p/jucelib/

7.2.2. Ajax Client Engine

Ajax Client Engine is a powerful remoting wrapper.

✓ Object-oriented API.
✓ Cross-browser support
✓ Flexible:
  o Request option
  o Request parameter validation
  o Callback argument
  o Callback option
  o Callback timeout
  o Tracing service
  o Caching service
  o Polling service
  o Common callbacks
  o Exception handling
✓ Open-Source (MIT License) by Li Shen.

http://www.lishen.name/

7.2.3. Ajax Gear

AjaxGear Simple and cross-platform AJAX toolkit.

✓ Enables a Web browser to make asynchronous call to the Web server without the need to refresh the whole page.
✓ Use the Ajax class to communicate with the Web server.
✓ A complete website is available for download to see AjaxGear in action.
✓ Use the AjaxGear.PageEngine.js file to learn how a website can easily use the toolkit.
✓ Current version includes a Tab component.
✓ Three components are currently being developed as part of the AjaxGear toolkit namely progress bar, auto complete and form validator.
✓ Released under MIT open-source license by Allan Spartacus Mangune of ArchCommerce.

http://www.ajaxgear.com/

7.2.4. AJForm

AJForm is a JavaScript toolkit which simply submits data from any given form in an HTML page, then sends the data to any specified JavaScript function. AJForm will, with minimal developer interaction, automatically “AJAXify” your HTML forms so you don’t have to write even a line of AJAX specific code. If the browser doesn’t support it, the data will be sent through the form as normal. It is designed to be for entry level AJAX coders. It’s single purpose is to send data from any HTML form via XMLHTTP.

The AJFORM Framework provides you with:

✓ 3 Step setup
✓ Automatic support for any HTML form
✓ Ability to implement with little coding knowledge
✓ Open-Source (BSD License)

http://projects.aphexcreations.net/ajform/

7.2.5. Javeline Teleport

TelePort is Javeline TelePort is an Open Source server communication layer supporting all popular web services such as XML-RPC, SOAP, JSON, REST and Comet. Javeline TelePort is an Open Source server communication layer supporting all popular web services. Developers can implement asynchronous communication as part of their AJAX projects. It has robust error, time-out and retry handling. TelePort features an advanced polling interface for controlling reconnecting and the polling frequency, and has advanced debugging features. Currently Javeline TelePort has built-in support for XML-RPC, SOAP, JSON, REST, Comet and Javeline FrameWork-specific protocols JPHP. TelePort is released under the Open Source GPL license so support for other protocols can easily be added.

Features:

✓ Modular protocol support
✓ Robust polling engine
✓ Elegant error handling
✓ Elegant time-out handling
✓ Automatic retry
✓ Comet server component included
✓ JPHP server component included
✓ Realtime communication trace for easy debugging
✓ Interchangeability between protocols
✓ Highly Scalable
 ✓ Build for reliability
 ✓ Stable communication foundation
 ✓ Tested in real-world applications

http://www.javeline.org/

### 7.2.6. Lokris

Lokris is a lightweight XMLHttpRequest wrapper. Remarkable: user can set timeout limit and timeout handler.

 ✓ Small footprint (minimized ~ 3K)
 ✓ Synchronous/Asynchronous cross-browser Ajax calls (of course)
 ✓ Convenience features for easy XmlHttpRequest handling
 ✓ Handling of plain text and XML data
 ✓ Setting timeouts for Ajax calls / Setting timeout handlers
 ✓ Setting error handlers for Ajax Calls
 ✓ Easy to understand, no blown up code, no overhead
 ✓ BSD-style licence

http://www.ajaxbuch.de/lokris/

### 7.2.7. Sack

SACK is a simple set of code to allow you to put AJAX into your webpages with none of the fuss or overhead of other packages. Just AJAX, just simple. It is a thin wrapper around XMLHttpRequest.

 ✓ Caller can specifies callback function or callback DOM object. With a callback DOM, the response text is pushed directly into the DOM.
 ✓ Full documentation and example code
 ✓ Open source (modified MIT license) by Gregory Wild-Smith.

http://www.twilightuniverse.com/projects/sack/

### 7.3. JavaScript Graphics and Effects frameworks

#### 7.3.1. Bitjuice

Bitjuice is a bitmap/raster API for easy graphics in the browser.

 ✓ Uses an HTML table (not canvas/SVG).
 ✓ Works in old and new versions of all major browsers.
 ✓ A single Javascript file, several KB.
 ✓ Open-source (MIT license) from Michael Mahemoff.

http://ajaxian.com/archives/bitjuice-javascript-bitmap-api

#### 7.3.2. Open-jACOB Draw2D

 ✓ OpenjACOB Draw2D OOP javascript vector library to create Vision like Ajax web applications.
 ✓ Open-source (LGPL 2.1 or later) by andreas Herz.
7.3.3. InfiView™ - Interactive & Infinite Ajax Graphical Maps and Diagrams

InfiView is a software platform for developing enterprise interactive, dynamic graphical maps and diagrams. InfiView is using Ajax technology, and its advanced dynamic memory use enables use of ANY data size (from very small all the way to infinite). InfiView applications can be programmatically generated from your databases or your web services and can generate both embedded controls and complete applications.

Features:

✓ A class based, Object Oriented API
✓ A complete windowing system
✓ Native XML, SOAP and XML-RPC support
✓ Supports enterprise scale projects
✓ Cross-browser, cross-platform support
✓ Server independent architecture
✓ Interoperability with new and existing resources
✓ Uniform development methodology
✓ Your InfiView applications are agnostic to:
  o Server - any server (Java, .NET, PHP, etc. can be used with Bindows)
  o Backend platform - Any backend hardware and backend software
  o Operating system - Any operating system can be used on the client (as long as it includes a supported browser)
  o Language - All languages supported. Bindows supports Unicode
  o Browser - most browsers are supported (i.e., Internet Explorer 5.5 and higher, Mozilla 1.4 and higher, Netscape 7.1 and higher, Firefox, K-Meleon, and Camino)
✓ InfiView applications are defined by XML data and supports standard SOAP Web Services.
✓ Integrated Layering System: it can include multiple layers such as map, network topology, live dashboard, ...

Main Benefits

✓ Advanced Memory Management - supports data of ANY size (from small to INFINITE)
✓ Open Platform - you define and select your tools, interface, style and logic
✓ Add a smart layer to existing data, static or dynamic (for example: maps, flowcharts, schematic diagrams, LDAP and SNMP)
✓ Superior GUI
✓ Rich, scalable and dynamic client-side charting
✓ Dramatically reduces server hits
✓ Dramatically reduces network bandwidth
✓ Unified design methodology
✓ Complete separation between server and client development. Any server development tool might be used.
✓ Knowledge of HTML or CSS is not required
✓ Support for multilingual interface
✓ Zero-footprint (for the client-side: no installation, no upgrades, no patches, ever)
✓ Powered by Bindows
  o Strength of a framework, wealth of components
  o Interoperability of the framework components
  o No memory leaks
  o No code duplication
  o Framework of components versus collection of widgets
✓ Vendor independent
  o GUI and client side software are independent of server technology choice
  o Investment in client software is reusable with other server products
  o Develop once, use on many web platforms
✓ Built for integration
  o XML based GUI and client application definition
  o Comprehensive Framework and Object model
  o Enables out of the box, zero footprint deployment
✓ Built for Security
  o Operates in Browser sandbox
  o Zero footprint
  o SSL tested
  o No cookies required
✓ Reduced TCO
  o Thin client
  o No installation, no upgrades, no patches (for the client-side)
  o Dramatically reduces server hits
  o Dramatically reduces network bandwidth
✓ Open Platform

  http://www.infiview.com/

7.3.4. Interface

Interface Interface is a collection of plugins for jQuery javascript library. With this plugins you can build Web 2.0 applications with the same same simplicity as using jQuery and provide a rich user interface.

  http://interface.eyecon.ro/

7.3.5. Moo.fx

Moo.fx builds on a lite version of Prototype library to provide an extremely compact effects package.

✓ Visual Effects
✓ cookie memory for effects
✓ 3-6kb total file size
✓ Open-source (MIT license) from Valerio Proietti

  http://moofx.mad4milk.net/
7.3.6. Script.aculo.us

Script.aculo.us, building upon the wonderful Prototype JavaScript library, script.aculo.us provides you with some great additional ingredients to mix in.

- Visual effects
- Drag and drop
- Controls
- Live demos and sample code
- Licensed under a MIT-style license

http://script.aculo.us/

7.3.7. MxGraph

MxGraph is a Javascript library to draw and share complex diagrams through a web browser, across every platform. Using open standards, mxGraph does not depend on any third-party plugins and proprietary software. MxGraph works straight out of the box, no client configuration, no plugin installation, no platform dependencies. MxGraph simplifies the development of critical component for rich web applications.

Advantages

- Centralised updating managed at the server
- No client plug-ins or installation required
- Platform independent
- Deployment using Java, .NET, PHP, static HTML and more
- Easy configuration of the application using XML
- User interface is HTML and the data is exchanged in XML
- Java, .NET or PHP backend for concurrent diagram editing

Disadvantages

- Open license only for non commercial use

Advanced functionalities available (main features are explained above):

- Specific class for creating and configuring an editor (popup menu; default toolbar; graph; undo manager, outline, tasks, help, properties).
- XML editor configuration file: it’s an XML file used to setup the the editor object hierarchy. This is normally done at startup time to configure the editor, graph, model, toolbar, popupmenus etc using the I/O subsystem.
- I/O functionalities: to map all non-object fields and all object field of the graph using XML.
- Possibility to recall saved diagram in XML format.
- Different layouts to represent the graph which can be selected for example with a task window during the creation of the graph (CircleLayout, CompactTreeLayout, CompositeLayout, Fast OrganicLayout, GraphLayout, ParallelEdgeLayout, PartitionLayout, StackLayout).
- Dynamic editing of objects (cells and connections) name and properties, for example using dropdown menu to edit the properties, to add sons or to connect other objects.
- Random rearrangement of the diagram.
✓ Possibility to use images in the diagram and to associate one to an object (for example to represent a database, a service or other).
✓ To define more layers for the graph in order to view only a specific part of the graph, to view some objects and hide the rest (for example, in our case, the action units and not the jsp units).
✓ To associate an object with an action like open a URL.
✓ To mark an object with a check sign.
✓ Tooltip functionality for the objects.
✓ Undo / Redo functionality (Undo Manager): when changing the graph model, an <mxUndoableChange> object is created at the start of the transaction (when model.beginUpdate is called). All atomic changes are then added to this object.
✓ Cut / Copy / Paste functionality.
✓ Drag and Drop functionality.
✓ Possibility to implement a zoom instrument (already developed).
✓ Personalized use of connection ports (in order to use more kinds of connections between objects).
✓ Preview window (outline) to show a global vision of the entire graph (useful for very big graphs).
✓ Save (generation of XML), print and show (preview) functions.
✓ Google gears support to manage local database (for example for an editor of database schemas).
✓ Possibility to use stylesheets.
✓ Group function to group the objects.
✓ Select All, Vertices or Edges function.
✓ Pan function to move the diagram in the work area.
✓ Swimlanes to visually group the objects in the work area (like the definition of a business process with different actors).
✓ MxGraph Log: to implement a simple console to see info, DOM, trace, properties.
✓ Possibility to use dynamic windows (task windows) to manage the environment, the work area and the diagram.
✓ Different shapes: Actor, Arrow, Connector, Cylinder, DoubleEllipse, Ellipse, ImageShape, Label, Line, Polyline, Rectangle, Shape, RhombusShape, Swimlane, Text, Triangle.
✓ Datatransfer function: singleton that implements drag and drop between different graphs.
✓ Cross-browser event-handling support.
✓ Possibility to create HTML forms.
✓ Identity for JavaScript objects. This is implemented using a simple incrementing counter which is stored in each object under <ID_NAME>. The identity for an object does not change during its lifecycle.
✓ Possibility to create VML and SVG paths.
✓ To Implement internationalization by providing any number of resource files.
✓ Session for sharing an mxGraphModel with other parties via a backend that acts as a multicaster for all changes.
✓ XML HTTP request wrapper.

Architecture:
The mxGraph package contains a client software, written in JavaScript, and a series of backends for various languages. The client software is a graph component with an optional application wrapper that is integrated into an existing web interface. The client currently runs on Firefox 1.5 and Internet Explorer 5.5 and later and requires a webserver to deliver the required files to the client. It is also possible to run the client from the local filesystem without a webserver. The backends may be used as is, or they may be embedded into an existing server application in one of the supported languages.

http://www.mxgraph.com/

7.3.8. Batik Java SVG toolkit

Batik is a Java-based toolkit for applications or applets that want to use images in the Scalable Vector Graphics (SVG) format for various purposes, such as display, generation or manipulation. With Batik, you can manipulate SVG documents anywhere Java is available. You can also use the various Batik modules to generate, manipulate and transcode SVG images in your applications or applets. Batik makes it easy for Java based applications or applets to deal with SVG content. Using Batik’s SVG generator module, a Java application or applet can very easily export its graphics into the SVG format. Using Batik’s SVG viewing component, an application or applet can very easily integrate SVG viewing and interaction capabilities. Another possibility is to use Batik’s modules to convert SVG to various formats, such as raster images (JPEG, PNG or TIFF) or other vector formats (WMF or PDF, the latter two due to the transcoder provided by Apache FOP).

The Batik toolkit includes the following:

Modules
An SVG DOM implementation
A set of SVG microsyntax parsers
A scripting module
A generator that creates an SVG document from Java2D calls
A Swing SVG component
A transcoder module

Tools and applications

- Squiggle, an SVG browser
- An SVG rasterizer
- A TTF to SVG converter
- A pretty printer for SVG source files

http://xmlgraphics.apache.org/batik/

7.3.9. Big Faceless Java Graph Library

The Big Faceless Java Graph Library is a Java class library for creating graphs and charts. Version 2 of the Graph Library builds on the same full 3D platform and adds a substantial number of new features requested over the last 5 years. Enhancements include JSP Tag Library interface, to allow simple creation of graphs with interactive JavaScript direct from the JSP; extensible XML interface, output to SVG as well as PNG, Flash and PDF; combination of lines, bars or any other type of data on the Axes Graph plotted against up to four axes; logarithmic and "Gap" axes, for plotting large ranges of data; the same extensible Axes, 3D engine and simple design as before; full 3D engine. All graphs are rendered in shaded 3D and may be rotated to any angle. The new Big Faceless Java Graph Library will allow users to create complex graphs with very little code.

Features:
- XML and JSP Tag Library interface, for extremely easy integration with web projects
- SOAP and XML Web Services interface for graph creation
- Very comprehensive API and Tag documentation with dozens of examples
- Embed JavaScript in your tags to easily create powerful rollovers
- Full 3D engine with adjustable light source, produces the most realistic graphs on the market
- Output to PNG, SVG, Macromedia Flash, PDF or java.awt.Image for use in Applets
- Graph Dates, Currencies, Percentages, Numbers and Discrete data out of the box, easily extended for total control over the axes
- Mix Bars, Lines, Candles, Bubbles, all on the same graph
- Add text, lines or custom markers anywhere on the graph, including axes
- Use up to four axes on the one graph, including logarithmic axes
- Plot mathematical functions, data samples and spline curves fitted to data, all on the same graph
- It's fast - plot several 3D graphs a second to PDF or SVG
- Tag Library interface: it simplifies the creation of embedded graphs in the web page choosing among simple PNG, vector image, SVG and Flash configuring the XML
- Very reliable and efficient in terms of load times, scalability, and usability
- Complete trial version but not open license

http://big.faceless.org/products/graph/
7.3.10. JSGraphics + Drag&Drop library

This JavaScript VectorGraphics library provides graphics capabilities for JavaScript: functions to draw circles, ellipses (ovals), oblique lines, polylines and polygons (for instance triangles, rectangles) dynamically into a webpage. Draw Shapes Vector Graphics Library achieves optimized performance and cleanly arranged pixel stair-step patterns (pixel-optimization).

Features:

✓ Simplicity of usage with a set of method to draw shapes
✓ Performance: Draw Shapes JavaScript VectorGraphics Library tries to minimize the amount of such overhead and optimize performance using also fast algorithms (based on Bresenham algorithms)
✓ Alternative to SVG or <canvas> tag
✓ Cross-browser functionality
✓ Complete documentation and support
✓ Open-source license
✓ Limited functionalities, however it’s possible to combine its usage with the JavaScript DHTML API Drag & Drop

http://www.walterzorn.com/jsgraphics/jsgraphics_e.htm

7.4. JavaScript Flash frameworks

7.4.1. AMFPHP

AMFPHP is an open-source implementation of the Flash Remoting framework. Amfphp is an RPC toolkit for PHP. It allows seamless communication between Php and:

✓ Flash and Flex with Remoting
✓ JavaScript and Ajax with JSON
✓ XML clients with XML-RPC

Amfphp lets you focus on features instead of implementation details. Amfphp has a built-in service browser which lets you test your services before you start writing the front end, and allows you to generate code for various clients.

✓ Nothing required - PHP4/PHP5 compatible, no extensions needed
✓ Low footprint, lightweight, fast
✓ Convention over configuration (service and class mapping)
✓ Can be embedded into a framework (see CakeAmfphp, Seagull)
✓ Services are "non-specific" PHP classes that are portable to anything without code change
✓ Productivity tools included (service browser, code gen, profiling)
✓ XML clients with XML-RPC, JSON
✓ Not a framework by itself (use your own)
 ✓ It’s fast, reliable, more stable and 100% free and open-source.
 ✓ Use of NetConnection debugger, which shows you exactly what's being sent between the client and server.
 ✓ Remoting uses AMF, a very lightweight binary format that cuts the bulk out of packets, meaning data exchange is a lot faster than with XML
 ✓ Open-source license (GPL) created by Wolfgang Hamann and maintained by various developers.

http://amfphp.sourceforge.net/

7.4.2. Stream

Stream is a data communications platform for the Web 2.0. Stream takes the concepts of ajax and adds true push support through the utilization of a custom socket server, web service platform, and client interface libraries. Stream enables developers to create data driven and extremely low latency user interfaces. Stream works by scripting a Macromedia Flash XmlSocket object through a technology called liveconnect. But thats really just the surface. Stream is a platform, including tools to integrate this push technology with your development workflow. It also provides a SOAP-enabled server that can handle complex scaling scenarios where traditional http push techniques cannot. The Stream server will notify the client when a message is received.

Features:

 ✓ Plug-in API (C# and JavaScript client libraries)
 ✓ No More Polling. (True server pushed events, not emulated events through polling). See HTTP_Streaming.
 ✓ Ideal for chat, monitoring and client interaction systems that would requiring polling.
 ✓ Sessions (Per-Session and Per-Page state tracking.)
 ✓ Event Monitoring (Interact with mouse events, including maximum fire rate throttling)
 ✓ SOAP Support [ Integrated with WSE 3! ]
 ✓ .NET 2.0 Architecture
 ✓ New Fully Event Driven Client Side API!
 ✓ WS-Addressing Support
 ✓ Session Support
 ✓ Dual-License with GPL!
 ✓ Full Source Code
 ✓ Currently Alpha, Technology Preview / Proof of concept Stage.
 ✓ Dual License Open-source (Commercial License and GPL) by StormTide Digital Studios. Full Source Available.

http://stream.stormtide.ca/

7.4.3. Aflax

Aflax stands for Asynchronous Flash and XML. Defined simply, Aflax is a development methodology which combines Ajax and Flash to create more dynamic web based applications. the Aflax technology is available as a library that enables developers to use JavaScript to fully utilize all of the features of Adobe's Flash runtime -- including graphics, networking, video and camera support. By using Aflax, all development for the browser can be accomplished using JavaScript, both Flash and AJAX development. The biggest difference with calling functions directly in Flash is that you don't need to use ActionScript or the Flash IDE and you don't need to compile

89
anything. Aflax hides all the communications to the Flash object so all of your logic can remain in one place.

Features:

✓ Improving of online user experience
✓ Use of Javascript to utilize Flash features
✓ Use Flash features directly in your development tool
✓ Not need to use Actionscript
✓ API docs and community support
✓ Open MPL (Mozilla Public Licence)

http://www.aflax.org/

7.4.4. OpenLaszlo

OpenLaszlo is an open source platform for creating zero-install web applications with the user interface capabilities of desktop client software. OpenLaszlo programs are written in XML and JavaScript and transparently compiled to Flash and, with OpenLaszlo 4, DHTML. The OpenLaszlo APIs provide animation, layout, data binding, server communication, and declarative UI. An OpenLaszlo application can be as short as a single source file, or factored into multiple files that define reusable classes and libraries. An OpenLaszlo application developed on one machine will run on all leading Web browsers on all leading desktop operating systems.

The OpenLaszlo SDK consists of a compiler written in Java, a runtime JavaScript library, and an optional Java servlet that provides additional services to the running application. The Compiler provides support for these features:

• XML UI Compilation
  The LZX user interface description is a declarative XML description of an application interface. The compiler, under the developer’s control, transforms these XML descriptions into either SWF bytecodes or DTHML that create an application interface when the application is executed.

• ECMAScript Compilation
  LZX user interface classes and instances are annotated with ECMAScript methods and event handlers. The compiler transforms them into optimized bytecode.

• Media, data, and font compilation
  Transcodes PNG, JPG, GIF, SWF, MP3, and TrueType font files, and embeds them into application object files (SWF only).

• Size Profiling
  HTML reports of application size.

The Servlet provides support for these features:

• Media transcoding
  Transcode a variety of media types into a set that the Flash 7, 8 and 9 players can display.

• Data transcoding
  Optionally transcode XML data responses to a compact binary format.

• Caching
  Optionally cache the results of requests to media and data servers.
• Proxying
  Proxy requests to remote media and data servers, using a server-side whitelist or blacklist.

• XML Services
  Back-end support for SOAP, XML-RPC, and JavaRPC requests.

• Logging and administration
  An administration console and declarative user interface allow text-based and GUI inspection of server state.

The Runtime Library provides support for these features:

• Components
  A rich library of user interface components, including web form components and configurable grid and tree, let you get prototype quickly.

• Layout
  A variety of layout managers position user interface components automatically when their sizes change.

• Animation
  A declarative animation system provides declarative or procedural animation for all user interface elements, with automatic ease-in and ease-out for design-quality animation.

• Constraints
  A declarative constraint system automatically update user interface attributes from the values of data sets, other user interface attributes, or user events.

• Data binding
  Automatically fills in user interface components values from XML data sets, and can make multiple copies of components that match multiple values.

• XML Services
  HTTP requests for XML, SOAP, XML-RPC, and JavaRPC services.

• Debugging
  An embedded command-line debugger displays runtime warnings and logging information, and interprets ECMAScript expressions within the context of the application.

Benefits:

✓ Improving of online user experience
✓ Rapid XML development approach
✓ Scalable deployment architecture
✓ Support for any web browser with a single code base in XML and JavaScript
✓ Deploy applications from any HTTP Web server in SOLO mode, or from a J2EE application server or Java servlet container running the OpenLaszlo Server under Linux, UNIX, Windows or Mac OS X
✓ Display applications in any Web browser enabled with the Flash 6 Player
✓ Complete documentation and community support
✓ Open-source license

http://www.openlaszlo.org/
7.5. Javascript Logging frameworks

7.5.1. Log4javascript

Log4javascript is a JavaScript logging framework based on the Java logging framework log4j. Its purpose is to provide JavaScript developers with a familiar, robust and flexible logging framework with which to debug JavaScript applications. This is becoming increasingly important as web interfaces become richer thanks to the recent appreciation of the possibilities of Ajax.

Log4javascript's main features:

- based on Java's log4j, implementing loggers, appenders, layouts and levels, providing a familiar interface;
- a powerful log console with severity filters and searching (including regular expression searches);
- a flexible Ajax appender that posts log messages back to the server;
- several layouts for flexible formatting of log messages, including XML, JSON, and Pattern layouts;
- crunched and stub versions of the main .js file included in the distribution.

Since a major use of JavaScript logging is in browser testing, log4javascript is tested and works across all recent major browsers, including:

- IE5+ (for Windows)
- Mozilla, Firefox, Netscape
- Safari 1.3+
- Opera 7.5+
- Konqueror 3.4+

Log4javascript is distributed under the Apache License, Version 2.0.

http://www.timdown.co.uk/log4javascript

7.5.2. JSLog

JSLog is a Javascript logging framework.

- JSLog's focus is on ease of integration and deployment, providing a logging infrastructure with minimal performance impacts, and allowing you to easily "productionize" you code (even with the logging statements still included), by changing one configuration setting.
- Open-source by Andre Lewis

http://earthcode.com/blog/2005/12/jslog.html

7.5.3. Log4Js

Log4js is a logging API for JavaScript. The main goal is to have a robust and solid logging API which is very simmilar to the Java logging API Log4j. There are several ways to log using "appender"s. The current available Appenders are:

- DummyAppender: log nothing.
- ConsoleAppender: open a new window in the browser or an inline div element and insert log messages in real time.
WindowsEventAppender: send log messages in the MS Windows event manager (Internet Explorer only).
FileAppender: write log messages in a local file on the client (IE and Mozilla).
AjaxAppender: allow to send log messages to a remote server with asynchronous HTTP request.
MetatagAppender: add log messages as meta data.
JavaScript Console Appenders for Opera, Mozilla and Safari

Many Appender can be used in a same time. To format the logs there are several "Layout" implementations available. Log4js is distributed under the Apache License, Version 2.0.

http://log4js.berlios.de/

7.6. Javascript XML frameworks

7.6.1. Google AjaxXSLT

AJAXSLT is an implementation of XSL-T in JavaScript, intended for use in fat web pages, which are nowadays referred to as AJAX applications. Because XSL-T uses XPath, it is also an implementation of XPath that can be used independently of XSL-T. XSL-T is a language for transforming XML documents from one language to another. An XSL-T style sheet would be used, for instance, to convert the unformatted content from an XML document into the a fully-formatted HTML document. AJAXSLT takes this process one step forward, by implementing XSL-T in Javascript and having it run in your browser. Thus, your web browser can fetch XML documents directly from the server, and perform the format conversion locally; thus saving time and bandwidth.

Utilizes Backbase AJAX engine - cross browser compatible
Declarative (XML-based) AJAX programming model.
Open-source license (BSD). By an search solutions company which refers to itself as "Google".

http://goog-ajaxslt.sourceforge.net/

7.6.2. Backbase XML/XSLT/XPATH

Backbase Enterprise Ajax is targeted at developers who need to create more compelling Ajax Web interfaces for any type of software application. It takes care of all the Ajax plumbing work and makes Ajax development fast and easy. It integrates with any type of technology that can output (X)HTML. Backbase Enterprise Ajax runtime is a robust and comprehensive Ajax engine that transforms any web browser into a rich presentation environment. It provides a large amount of generic and reusable Rich Internet features, such as single page user interfaces, drag-and-drop, animations, advanced keyboard navigation, real-time updates and validations, flexible user interface skinning and data binding facilities. Backbase Enterprise Ajax works like an application server inside the browser: it offers a runtime with all the Ajax “plumbing” you need to create enterprise-class Ajax applications. Developers can code in XML or in JavaScript. The framework facilitates smart code reuse practices with an architecture based on object oriented patterns and multiple inheritance support. In addition to the runtime, the Backbase framework includes over 250 widgets and functions that can easily be customized. The framework includes the ability to customize, extend components, and create entirely new components, without writing browser-specific code, and without experience in the details of Ajax. The framework is extensible and integrates directly with Struts, JSF, SOA, Portals and other server platforms. It includes an Ajax development plug-in for the Eclipse platform and a browser-based debugger.
Backbase Enterprise Ajax – Client edition consists of a client-side JavaScript runtime, Ajax Widgets, and developer tools:

- Zero install Ajax Runtime
- OO Enterprise Ajax framework
- Built-in cross browser support
- XML & JavaScript APIs
- 250+ Widgets w/ source
- Customizable components
- 3rd party components integration
- Browser-based Debugger
- Eclipse plug-in
- Accessibility Compliancy
- Complete documentation and support
- Open Standards and Free Community Edition license and Commercial license

http://www.backbase.com/

7.6.3. Sarissa

Sarissa is an ECMAScript library acting as a cross-browser wrapper for native XML APIs. It offers various XML related goodies like Document instantiation, XML loading from URLs or strings, XSLT transformations, XPath queries etc and comes especially handy for people doing what is lately known as "AJAX" development.

- Portable XMLHttpRequest creation.
- Portable XPath queries.
- Portable DOM manipulation.
- Portable XSLT.
- Portable serialisation to XML.
- JSDoc, tutorials and articles
- Cross-browser support
- Open-source (GPL 2.0 or LGPL 2.1, your choice). From various contributors.

http://dev.abiss.gr/sarissa/

7.6.4. xWire

xWire Framework is a mature, object-oriented, enterprise-class Javascript toolkit for building cross-browser applications.

- Mature toolkit (originally written in 2001)
- Simple API for cross browser functionality including XmlHttp, DOM, XPath, XSLT
- Completely object-oriented using Javascript classes
- Configurable logging mechanism
- Incredibly flexible event mechanism
- Includes some great widgets such as a hierarchical tree and notebook
✓ XML form binding for easily copying HTML form field values to/from one or more XML documents
✓ Working examples
✓ Frequent enhancements
✓ Well documented with meaningful and thorough comments throughout the code
✓ Compatibility: IE6, Moz/FF 1.x
✓ Apache License V2.0 by Jeff Roberts and Mark Nabours

http://xwire.solutionpioneers.com/

7.6.5. ZK - Ajax/XML Web Framework

ZK is an open-source enterprise Ajax Web framework that enables rich user interface for Web applications with little programming.

Features:

✓ With event-driven components and a markup language, developing is as simple as programming desktops and authoring HTML.
✓ No hazards of exposing business logic on the client.
✓ 1st support both Ajax and Mobile (JavaME).
✓ Server-Centric processing.
✓ Event driven model.
✓ 77 XUL and 83 XHTML off-the-shelf components.
✓ Macro components.
✓ Annotations and data-binding (access database in ZUML with zero Java codes).
✓ Server push.
✓ Script in EL Expressions and scripting languages, such as Java, JavaScript, Ruby and Groovy.
✓ No need JavaScript. (POJO)
✓ Integrated with other frameworks/solutions: JSF, JSP, Ext-JS(YUI), Google Maps, Google Calendar, Spring, Hibernate, Seam, Struts, Seasar, Acegi, JFreeChart, FCKeditor, amCharts, DOJ0, Liferay, Timeline, 188N, Twitter, etc.
✓ Demo and Tutorial
✓ Open-source license (GPL) by Potix Corporation

http://www.zkoss.org/
8. Bibliography

3. Building Web Applications with UML - Jim Conallen, 2002 Addison-Wesley
4. UML per il Web - http://java.html.it/articoli/leggi/2214/uml-per-il-web/
7. Programming Jakarta Struts - Chuck Cavaness, 2002 O'Reilly
13. Ajax in Action - Dave Crane, Eric Pascarello, with Darren James, 2006 Manning Publications Co
14. Pragmatic Ajax A web 2.0 Primer - Justin Gehtland, Ben Galbraith, Dion Almaer, 2005 The Pragmatic Programmers LLC
15. Build Your Own Ajax Web Applications - Matthew Eernisse, 2006 SitePoint Pty. Ltd
17. Ajax for dummies - Steve Holzner, Wiley Publishing Inc
27. OpenJacob Draw2D - http://draw2d.org/draw2d/
29. XML Tutorial - http://www.w3schools.com/xml/default.asp