IWW Website Proposal

Submitted by RadicalFusion
November 22, 2004

Table of Contents

Executive Summary

Primary Objective: Design and implement a complete web-based system including:

- a highly interactive and logically organized public website
- a secure and user-friendly web-based administration tool for managing dynamic content on the public website
- E-Commerce module to be leveraged by both merchandise sales and fee-based services, such as dues processing.

Secondary Objective: Build the website using technology and methodologies which produce highly flexible and extensible software, allowing for easy integration of new features not yet conceived and virtually eliminating the maintenance requirement that often makes professional websites cost prohibitive.

Estimated Project Duration: 6-8 weeks

Estimated Projected Cost: $25,000.00*

*Price based on client acceptance of all project components. Each component is independent and any component can be postponed for integration at a later date.

Technical Volume

Methodology

At RadicalFusion, we pride ourselves in producing high quality software by following industry standard software development best practices. Our methodology includes techniques and processes which are internationally recognized and have been developed by the some of the world's leading software development teams. Our approach is grounded in solid object oriented analysis and design techniques including use case development, Entity-Relationship and class diagrams with UML, interface prototyping, and disciplined unit testing throughout the entire course of development. Taken together, these object oriented techniques lead to more flexible and yet more stable code as well as higher rates of code reuse (which translates directly into greater code efficiency and ease of code maintenance).

System Design

The system design phase is arguably the most important phase of the project. It is during this period that most of the functional requirements are identified and translated into use cases. Use Cases are documents containing numbered steps representing the actions a user might take to achieve a particular result from the system. A simplified use case might look like this:

User Login

1. System displays login screen
2. User enters their username and password
3. System checks username and password against the database
4. System finds username and password, sends authorization cookie to browser
5. System displays welcome message

Each use case will describe the process of using one of the features of the system. Use cases help guide the development of the application. Use cases also serve as a convenient metric for project completion, since completion can be measured simply by testing the final feature against the original use case. While it is not usually necessary to document every conceivable use case, all of the major functionality will be documented in this way.

Use cases can then be used to develop user interface prototypes. UI prototypes can be expressed in software, but are just as good when written on paper. Essentially, the prototype is a model of what the layout of particular functional controls will look like in the final product. In our case, these will be HTML pages. Prototypes allow us to perform usability tests with the end users to gain a sense of how easy it is to use the system. They also allow us to cluster related functionality into logical groups.

Implementation

In object oriented design, software programs are not simply a set of routines and subroutines operating on an external set of data. Rather, the software is organized into objects encompassing both data and behavior specific to that object. Each object maintains its own information as well as the code to perform actions which are required of it, hiding the details of that implementation from other parts of the system. For example, an e-commerce website may contain a shopping cart object which holds several product objects, and is capable of displaying the names of the products it is holding and calculating a total. To add a product to the shopping cart we simply pass it the ID of a product. We need not concern ourselves with how the information is updated in the system—just that it works.

What we achieve by building software in this way is a highly modular system in which functionality needed by a variety of different features is kept in one place and accessed through a common interface. Every part of the system, from the login component to the e-commerce component will be built as a set of objects. Components may then be modified or even replaced without affecting the rest of the system as long as they keep the same interface.

One particularly helpful technique in building object oriented systems is unit testing. Unit testing is a method of writing tests for each object as you build it, and sometimes even before you build it. By continuously running tests on the system during development, as opposed to testing only at the end, you are more likely to identify bugs quickly and eliminate them. In addition, because unit tests are specific to an object or component, each time you add a new object, you can run the suite of tests in a matter of minutes to be sure that the new object has not broken any of the previous tests. What you end up with is software that is far more stable than that which is tested and debugged only at the end of the development process.

Deployment
In order to achieve the smooth launch of web-based system, it is imperative to compose a deployment model or plan. The deployment model includes information about the target platform, any expected technical constraints which may arise and even scripts used for migrating data and code. With a deployment model in place, we will be more likely to identify problems with the launch before they emerge.

The techniques discussed above are all fairly intuitive and tend to strike even non-technical readers as an intelligent way to design web applications. For this reason, it is surprising to us that many web development professionals and teams do not follow a similar methodology when building complex systems. Far too many simply fly by the seat of their pants, building websites without regard to careful planning and implementation. This may account for the fact that nearly 80% of all software projects fail completely, and the vast majority of those which do reach completion do so over budget and well past their deadline. RadicalFusion uses its development methodology to ensure this doesn't happen.

Functional Components

The IWW website will consist of several unique functional components. Each component is responsible for managing a different set of user or system-driven functions. These components are as follows:

Content Management System (CMS)
The current website consists of static HTML pages. Changes to the website are cumbersome, since they must be implemented by a staff member trained in writing HTML code. The primary component of the new website will be its content management system. All data from the current site will be migrated into a database. Web pages will be generated dynamically, using a server-side technology which retrieves data from the database, combines it with an HTML template and sends the generated page to the browser. This method enables any authorized staff to add, edit or remove content from the website using an administrative content management tool.

Data items such as articles and press releases can be typed directly into the tool or copied and pasted. Data may also be organized into categories which themselves may be added, modified or removed at will. Categories may be nested within other categories and data items may appear in more than one category. The tool's editor will not be limited to plain text, but will include text formatting capabilities similar to those found in MS Word.

Functionality for maintaining the inventory of merchandise for sale in the E-Commerce component listed below will also be administered through the administrative tool.

Interactive Forms (Petitions and surveys)
There are two general approaches for creating user input forms in web-based applications. One is to create hand-coded HTML forms for each form variant and create a processing script for each form variant. For applications which include a large number of forms (several different petitions, etc.), this approach can be both error prone and cumbersome. The other approach is to design software to build each form dynamically, from meta-data stored in a database or other storage medium (such as XML).

RadicalFusion is available to custom build each form as it is necessary. However, we have a feature for generating dynamic forms which can controlled through the administrative CMS.

Personalized Data Delivery
The website will include a method for users to receive targeted content of special interest to them. This content may be displayed in the form of a user's "homepage," and/or delivered to the user as a regular email bulletin. In order to access this website feature, users will be required to register on the site. Once registered, users may modify their "homepage" and email preferences by identifying topics of interest and indicating the frequency with which they want to be alerted. Again, the functionality for managing user preferences and content delivery will be housed in a single component. The implementation of homepage display and email delivery may be abstracted into subcomponents. Additional sub-components for content delivery to other medium, such as handheld devices, may easily be plugged into the system in the future.

E-Commerce
The website will require the ability to accept and process credit cards for a variety of purposes. IWW merchandise will be sold on the website, and may be purchased in a manner similar to any e-commerce website. In addition, credit card payments may be processed to accept payment of membership dues online. All purchasing functionality will be encapsulated in the E-Commerce component.

A necessary requirement of any e-commerce website is a merchant account and Secure Socket Layer (SSL) certificate. These can be purchased from a variety of service providers including Verisign, Geotrust, and Linkpoint. RadicalFusion is not responsible for acquiring a merchant account for IWW, however we can provide technical assistance and will integrate the payment gateway into our shopping cart software.

Technology
There are many software platforms available for building web applications. Generally, RadicalFusion makes use of open source technology except when specifically directed by the client to use a particular software platform. In our experience, open source software tends to be more stable, secure and reliable with fewer bugs than its proprietary counterparts. In addition, while open source does not necessarily mean free, it often is distributed without charge and with minimal licensing constraints. As a result, we incur virtually no overhead costs using open source technologies for our projects. Our pricing reflects only our labor time.

Standards
Open source software alone does not guarantee a stable and extensible system. We also thoroughly document our code and adhere to industry standards in developing our software. This allows the programmers who come after us to manage and modify our code without pulling out their hair in frustration.

Design Patterns and MVC
Design patterns are methods for solving particular design problems in software development which have been established over time and documented. We make use of a variety of design patterns in order to improve the efficiency of our code. One pattern we use quite often is the Model-View-Controller design pattern, which is a means for separating code for the presentation of data to the user from the code that runs business logic on that data. Using an MVC pattern enables us to build the application completely independent of the display medium, allowing for other developers to produce the front end design without interfering with the business logic code.

Application Frameworks
We sometimes write an entire application from scratch. More often however, we leverage the power of open source by using third party application frameworks. There are many application frameworks available and some are better than others. Where applicable, we will deploy a framework to avoid re-inventing the wheel. These frameworks can be thought of as a set of tools one might use to build a house. Rather than building our own hand drills and table saws, we use tools that has been previously

manufactured. Examples include the SimpleTest framework for unit testing PHP code and the PEAR::DB database abstraction layer for connecting to a variety of databases.

Pre-built Solutions
Somewhat different than frameworks are pre-built, or “off the shelf” solutions. Most are billed as Content Management Systems, and they come with an array of benefits and drawbacks. RadicalFusion has used many of these solutions, and we have found some to be to our liking. The difficulty lies in finding one that fits the requirements of the project. All too often, these solutions are rigidly built and difficult to modify, precisely because they are not all designed with flexibility in mind. Rather they are designed to get you up and running quickly with a generic set of features.

It is unlikely that the IWW website will benefit from one of these solutions. However, we are not opposed to using one, and should one prove appropriate for the project, we will use it. Also, should we find a component which we need in one of these solutions, we will happily attempt to integrate it.

Membership Dues
At this point we do not feel it is appropriate to include an estimate for membership dues payment processing in this proposal. Once the E-Commerce component is in place, a wide variety of goods and services can be processed. However, we do not currently have enough information about the new third party dues tracking database and its corresponding API to make an accurate estimate for integration. RadicalFusion would be happy to submit a separate estimate for this feature when we know more about the system. Suffice it to say, the website core will be built with extensibility for this and other features in mind.

Management Volume

Project Management

As mentioned in the “Methodology” section in the Technical Volume, RadicalFusion employs a thorough methodology in our software development projects. Complimentary to the technical processes listed there, we also adhere to some project management processes. First, we meet with client representatives regularly to review project status and discuss any issues which have emerged. Second, we provide our clients an account on our extranet, accessible via our website, where they can track the progress of the project, post bugs found during QA, and communicate with the development team.

Our standard Service Agreement includes a statement of work identifying the phases of the project with due dates for particular deliverables. Attached to the Services Agreement is a requirements document containing a complete list of functional specifications. Any change in the requirements document may require a comparable change in the payment schedule and changes in the due dates of particular deliverables.

About RadicalFusion

RadicalFusion was established in 2002, by Angelina Grab and Sam McAfee. Angelina Grab is our usability specialist and in-house graphic designer. She also serves as the business administration arm of the company. Sam McAfee is an expert in object oriented software development and serves as the technical lead on all RadicalFusion projects. RadicalFusion makes use of a network of highly qualified professionals in areas of graphic design, programming, database development and internet strategies to augment its array of services.

We believe that providing access to software tools which help make labor organizations more efficient internally while allowing them to make a greater impact in the community is imperative to ensuring the survival and growth of the Labor Movement. Angelina Grab and Sam McAfee have been activists in local and international economic justice campaigns for many years.

RadicalFusion principals are members of the Graphic Artists Guild, UAW Local 3030.

Budget Volume

The following is an itemized breakdown of the development tasks required for the project.

- Website Design $12,000.00 USD*
  - 3 design options, each with complementary internal page designs; choice of one for entire website (based on current IWW website, there will be upwards of ten internal templates necessary for new website)
- Content Management System $5,000.00USD
  - Administration Area for managing website content
  - Member Login and registration
  - Membership Group Categorization and Group Specific Content
  - Dynamic Form Building Wizard
  - Email, Newsletter Functionality
  - Centralized Membership Database
  - Content Search Engine
- E-commerce Area $5,000.00 USD
  - Shopping Cart
  - Dues Paying Area
- Data Migration $2,000.00 USD
- Content Organization $1,000.00 USD
  - Appraisal of all current data, organized into Parent and subcategories for self-generating navigation

Total Cost: $25,000.00* USD
Time Estimate: 6-8 weeks

* Web design cost based on $7,000 for creative, $5,000 for template development and integration. Although creative can be done by another party, template development cost must be included in overall cost.