SonicSpider LLC.
www.sonicspider.com
P.O. Box 483, Bonsall CA, 92003
Ph# 619.955.6380
Fax# 760.453.2177

Custom Web Development Guidelines

Introduction

Unlike shrink wrap software, custom software development involves a partnership between the architect/programmer/developer (SonicSpider) and the owner/testers/users (Your Company) of the system. The software evolves from a general specification to a prototype, a growing "work in progress", then ultimately, a finished system. There are a number of issues that should be discussed from the beginning that will help make the process successful. Not all of these issues may apply to a given project. This is a guideline to inform all members of the development team of what issues to expect to encounter and what the expectations are of every member of the project. It will hopefully stimulate discussion and act as a core methodology for a successful project. This document also acts as a description of SonicSpider's core procedures and protocols for custom development.

Development Process Overview

Custom development involves three basic phases: Design, Programing, and Quality Assurance (QA) (and Documentation and/or Training if specified). Of these phases only the programing phase can be quantified with a cost estimate to any exacting degree, and ONLY if the design phase was through enough to allow for a detail specification to be developed.

It is tempting to minimize the design phase. The typical attitude is that “we will quantify the design as we go” or “..it is so simple that it should be obvious” or variations on those feelings. Interestingly, at the same time an “exact” or “good approximation” of the cost is expect. It is very simple; ONLY if there is a detail specification can there be anything approaching a “good approximation” of the cost. Detail specifications are developed during the design phase.

The “rule of thumb” on both the time and cost of a project is: “For every hour of programming, there will generally be “at least” one hour of design and two to three hours of QA and documentation. Ironically the programming phase depends on the design, and that phase is difficult if not impossible to estimate. This is because it is dependent on needs and requirements that are only discovered as the planning is being developed. It sounds like a vicious circle, which it can be at times, hence it is often referred to as the “bootstrap” phase of the process.

Design and Planning – Needs Assessment

Would a house be constructed without a detailed blue print? Maybe a small tool shed, but not a house. Architects expect to get paid for their designs and engineering. Software is no different. On occasion, the desired project centers around an existing application that is being re-written and expanded. On other occasions, the needed application is completely new. Either way the documentation of the expectations of the targeted users and the business needs of the company must be clearly presented. The key concept is that "**If it is not written down, it will not happen**". There are to be NO assumptions of "commonly accepted functionality". The obvious must and should be documented clearly. Some projects are expected to "evolve" and features are added and documented in increments. Other projects require a clear "feature complete" specification. Either way, each feature, each change, each enhancement must be clearly documented. Typically, one quarter to one third of the project time will be spend in some form of "Design and Planning" or "Needs Assessment". Which term best suits the needs of your project will depend on the type and scope of your project.

It is important to remember: "What is not clearly specified can not be cost estimated". Even if the project is meant to "evolve" each design step must be completed before the next increment of the project can be estimated.

Use Cases

A very specific part of the design process deserves special mention. Most web applications are intended for people to interact and accomplish specific tasks. Because of this fact, the development of "use cases" is the key to insuring that the applications will be "useful".

Use Cases are short "stories" that outline in a "dialog" fashion how a typical user will accomplish a task. These "use cases" allow for the validation of page sequence, field sequence and instructions found on any given page. During the QA phase these same "use cases" will be used to guide the testing and validate that the final application does in fact: "work-as-designed".

This is a critical step and will seem obvious and trivial. Unfortunately it has been SonicSpider's experience that how a typical user will use an application is rarely that obvious or trivial.

Programming

Once the planning and specification has been completed for either that phase of the project or the entire project, the programming portion can be more accurately estimated. The degree of accuracy is directly proportional to the degree of specificity of the design. Often this is the phase where there is the temptation to "throw manpower" at the project. The thinking is that if the programming specifications require 300 hours of programming and you put 30 programmers to the task they will be done in 10 hours. This is an exaggeration to make a point, and the point is that it is absurd. To understand the issues on this is beyond the scope of this guideline, but a very good book "The Mythical Man Month" by Frederick P. Brooks who is considered the "father of the IBM System/360" that clearly outlines the fallacy of throwing "bodies" at a programming task. The size of the project determines the number of programmers that can be effectively involved. Generally for very large projects ($100K or greater) three to four man teams are best. For small projects (under $50K) one programmer is best, two is possible only if the system is design in a modular fashion so that each programmer can work independently and not materially affect the work of the other.

Quality Assurance (Testing, Debugging and Documentation)

All testing and documentation is done on the system during prototyping and development, and continues during every phase of the project. The expectation should be that testing and documentation will involve a minimum of half of the expected project time. In addition, the first 6 months to a year into the normal use of the application, will require bug fixes and alterations. All of which are part of the normal cost of developing custom applications. The system will be placed in a "development domain" and links will be provided to all persons that will be testing and validating the system. Generally issues discovered should be outline by the tester and submitted in written form. If requested for larger projects, a development forum can created so that features, bugs and other issues can be discussed and documented. Also, ad hoc design and training conferences can be conducted with minimal cost and loss in development time, using the forum.

The following points apply in all cases:

1. The primary testers should be the same as typical users. These testers have the responsibility to thoroughly test every aspect of the system during the development of the system, and to report problems in a timely manner via email or in larger project in a forum or project management system, setup for this purpose.
2. It must be understood by the testers that finding and correcting bugs is a normal part of custom software development. The testers can minimize the time, cost and later surfacing of problems by addressing the following:
   2.1. Thoroughly testing every release and reporting in writing all problems in a timely manner.
   2.2. Making sure all modules are tested thoroughly, and by several targeted users. (Different users tend to find different problems.)

2.3. Rechecking all "fixed" problems and verifying the correction.

3. All bugs or problem reports MUST be in writing and can be faxed, emailed or entered in a designated forum or project management system. This insures proper documentation of bugs and problems and maintains clear communication between SonicSpider and the testers. It is the testers responsibility to insure that all reported bugs are confirmed and fixed as reported. The testers must be sure to retest all reported bugs or problems after receiving a corrected release.
4. SonicSpider's responsibility will be:
   - 4.1. Respond in writing as to the status of all documented problems, via email reply or as a reply to a post on the test forum or project management system.
   - 4.2. Address all problems/bugs or provide alternative solutions with the approval of the CLIENT within five (5) working days unless agreed on differently in advance. (Some issues may be deferred due to agreed on priorities.)
     - a) A "bug" is a problems that:
       - Is NOT working as designed.
       - Prevents the use of documented features. (i.e. Crashes)
     - b) Some "problems" are not bugs, but "features expected". Remember, if it is not specified in the working proposal it is NOT a feature, it is an "enhancement". Enhancements will require additional cost.
   - 4.3. Provide ongoing documentation of all corrections and their resolution.
   - 4.4. Place a priority on "blocking bugs" that inhibit the continued testing of the current release.

Training.

Training, if applicable, begins with the first release and is an integral part of QA. A sampling of targeted users should be using and practicing on the each release.

Release Definitions

Web application development is handled by specified release milestones and is defined as "any module that is complete enough for a tester to test the usability and requested functionality". Releases are provided on a frequent basis so as to provide a tight feedback between the development of the application and the user's needs. Any given release will frequently be missing features and is NOT intended at any time to be a complete and final application but a "work in progress". This evolution of releases insures that the user's needs are met and avoids surprises in the final application.

Though there is no hard and fast rules as to the types of releases that will be available and in what order, the following is a guideline as to what to expect. In all cases the developed "use cases" and design specifications are used to validate the deployed release version.

Generally there are three categories of releases:

1. Alpha Release – This is a release that may not actually “work”. It is meant to illustrate “structure” or layout issues.
2. Beta Releases – This are releases in which some portion or module is working sufficiently that it can be tested. When a Beta version is released it will also contain some form of instructions as to “what is to be tested or reviewed”. There will be ANY number of Beta Releases
3. Release Candidates – These are “feature complete” releases, where any one of the releases may be deemed “finished”. There will be ANY number of Release Candidates.

Following are some of the possible release scenarios:

- Prototypes of web pages. The first release (Alpha) is generally a collection of prototypes that graphically illustrate the general layout of fields and other user interface elements. This will provide an opportunity to model work flow, and the layout of fields, pages and other interface objects. User friendly, and usability of the interface should be reviewed and changes documented. It will also allow for discussions of work flow issues, use cases, and sequence of data entry.
- Preliminary working models of individual forms/pages (Betas). These will be released one at a time (or in logical groups) and will contain about 50 to 80% of the designed functionality. This begins the serious testing of usability and the assumptions made during the prototype phase.
- Integration of forms/pages. Once the primary functionality of the user interface is complete the individual forms or web pages are completely integrated and tested as a whole. At this point the testers should be attempting to input real data into the system so that any problems or other issues will surface.
- _ Reports design and testing.(if required) Once the underlying data structure is stabilized and its is certain that all needed data is being input, any report construction begins and is subsequently tested. Printing hardware/software problems are tested and output is approved.
- _ Parallel Testing.(if applicable) At this point the application has been approved by the testers as being "feature complete" and a final installation deployment is done. The application is now used in parallel with the old application for several weeks. This is one of the most challenging stages because it will involve extra work on both the testers and SonicSpiders part. SonicSpider must fix all problems within hours to insure continued work flow and the testers must allow extra time to input data into both systems. If a serious problem is encountered in the new system the old system will serve as a

backup. This is very rare, but it can happen.

The sequence of releases in each category and what they address can loop and repeat a number of times on different portions of a project. Therefore it is not a linear process but a "feedback loop" process that involves the close teamwork of SonicSpider development staff and you, the ultimate users of the application.

A Final Word

It is hoped that this document helps the reader understand the process required for a successful project. It has been documented frequently in many articles and books that “on the average” only about 10% of all projects are “successful”. This may seem depressing at first, but when studying the projects that succeeded vs. those that did not, it was the adherence of much of the contents of this guideline that differentiated the success or failure of a given project.

This process may seem complicated at first glance, but after nearly 20 years of experience in application development, this guideline fairly represents all of the components that would be required to successfully complete a project. Anytime some part of this process is side tracked, short changed, or omitted, it has often spelled the doom of that project. The SonicSpider team can ONLY assure you that your project will be successful IF this process is followed in a manner that is appropriate to the size and scope of your project, of which the SonicSpider team reserves the right to determine the appropriate level of adherence for your project. If you chose to take short cuts, SonicSpider can not be held accountable for the resulting software and success of your project.