



Make Your Software Idea a Reality

White Paper

Abstract

This paper describes the methods of taking a software idea from conception to reality.

Make Your Software Idea a Reality

Index

- ❖ Introduction – Your Software Idea
- ❖ A System, a Method, a Plan
- ❖ Getting Started – the Needs Analysis
- ❖ The Requirements Definition Stage
- ❖ Scope
- ❖ Design
- ❖ Specifications
- ❖ Development
- ❖ Testing
- ❖ Roll out.

Introduction – Your Software Idea.

When I was a youth, my dad always told me that I got a lot of “million dollar ideas”. Well, we all get great ideas, sometimes for new products or software applications but the real challenge is the process of making those ideas turn into a tangible software product that can be sold or implemented profitably. But then every software package we use from Excel to the game of Myst to the Internet itself was an idea at some point that changed into a reality.

The good news is the process of defining and building software is a project management method that is quite well defined. There are a lot of things to think about when making software come to life. It is worthwhile to do some research on whether the idea has been developed already and if not how to protect your idea once it gets into development, especially since if no patents exist, many times you can use ideas and features from competitors. This white paper will primarily focus on the technical process of taking a software idea from concept to distribution, i.e. making it a reality. By using many of the methods described in this paper, the process of software creation will be much smoother, more economical and be less prone to costly pitfalls.

We will discuss what you will need to be successful at building your software project. This paper assumes that the project is being developed within an already existing corporate structure like Tometa’s custom software development offices. The process of software creation by nature must occur within a setting where there are development resources, programming computers and compilers, management tools and test beds and of course, programmers. These resources can be hired, trained and purchased; but a viable option to get production underway quickly and efficiently is to outsource the work through a quality contract development provider with impeccable references such as Tometa Software, Inc.

A System, a Method, a Plan

The processes and steps described below can be used in any software development effort whether it be a windows game, a new web site or an application to be used in the corporate world. It is a development methodology that goes back several decades known as the Systems Development Life Cycle or SDLC for short. Now over the years tools and resources have been created to streamline the process. One such tool is *Microsoft Project* which has excellent resources for coordinating the process of software development. However, the process used here can be done with nothing more than a word processor and a spreadsheet. The key is to proceed in a structured fashion from step to step to assure that the project is built in such a way as not to waste resources and cut down on the risks of development.

There are risks in building a new software application. The process can be quite costly and lengthy. A typical software development cycle can take anywhere from three months to three years to complete. If the design of the software is not thorough before the first line of computer code is generated, it can cost huge sums to correct in the testing phase or cost even more if the software goes to market or into production flawed. The higher level of expertise and experience that can be put against the preparation process, the more that risk can be mitigated. Utilizing a trusted business I.T. firm such as Tometa, can assure that the greatest skill can be placed against the areas of the plan that are most at risk without incurring overhead for the entire project development process.

The 3 “P”s of software development are Prepare, Prepare, Prepare.

Getting Started – The Needs Analysis

The preparation for software development consists of several dependent steps each of which must be successful for the next to occur. The first step is *The Needs Analysis Document*. Needs analysis asks the blunt question: “What need does this software solve?”

The needs analysis phase feeds the design process by defining the true nature of the need. The outcome of the process is a *problem statement* which defines to the designers, the users and the developers just what this software is supposed to do. If we are able to realize a thorough understanding of the problem then the needs analyses document will contribute tremendously to the rest of the development effort.

It is not always easy to gain a realistic problem statement. One of the most common errors that consultants encounter is users asking for a solution that does not solve the problem they have. It pays to dig a little deeper, ask some relevant questions about exactly what the problem is.

Now if your software idea is being developed to be sold commercially, there may not be a “need” for the software but there may be a potential market. In that case a *market analysis* serves the same purpose to address the question “is there a market for my software?” In some cases, an outside firm might be needed to conduct the market analysis. If that is expected, a line item in the project budget should be introduced early in the project life to fund the study.

The last step in this process is the *User Sign Off*. The document is reviewed by the customer and if the step has been successful, both developers, managers of the development process and the users sign a document stating that the step was complete. By signing off each step of the way, the team agreement is documented which streamlines development and focus as the product begins to

take shape. A common unfortunate interference to sign off can be internal corporate politics. A skilled project plan accounts for that danger. In some cases the use of an outside objective project team member such as the skilled project leadership team leaders at Tometa can provide a way to reduce the impact of politics on the crucial sign off phase.

The Requirements Definition Stage

The Requirements Definition Stage defines “What does this software do?” The requirements are stated as *Functional Business Requirements* of the software. Even if the software is not being prepared for the “business community” the requirements should be stated in non technical terms, in terms that can be understood by the user.

The requirements definition will be in greater depth than the needs statement. For example, for a web page, we might have a Business Requirements Design, or BRD, addressing security, another addressing adding a shopping cart, another about the search engine optimization issues of the site and still another about the data infrastructure supporting the site which will not be visible to the user at all. This list of requirements will feed the generation of a specifications document which gives the developers what they need to build the software.

The key word is “requirements”. This document will define what the software must do to achieve its stated objectives. There may be additional functions and features that the software “could do” but those should be listed as an addendum.

Scope

The Scope Document then is the outcome of all of the hard work that has gone on to date. The Scope will spell out in detail just what is “in scope” and what is “out of scope” for this project. The Scope document expresses in a clear narrative:

1. The software approach
2. What the resulting product will look like?
3. What market or business need it will satisfy?
4. What components will be needed to make that design a success?

Along with core design concepts, the scope will also deal with interfaces, infrastructure issues such as date base specifications that must be observed, communications protocols, language preferences and the like.

As before, in the final step of the scope document is for all project participants, key team leaders and the clients and sponsors of the project will “sign off” on the scope signifying that all is clear to develop the project exactly as it is described in the document.

Much of this documentation and signing off may seem somewhat unnecessary. A developer will often think “let’s just program the thing” and get moving. However, when dealing with a software project of significant investment and time in development, these steps are absolutely critical to the success of the final product. Long experience has proven that the developers will be very glad these documents have been prepared when the rigors of development get underway. Above all, a thorough completion of these steps satisfies our challenge which is to satisfy the three P’s – prepare, prepare, prepare.

Design

The Design Document is not the final step before the project goes to the programmers but it is an important linking step. The Scope Document defines the project in lay terms understandable by business leaders and management. The Design Document is the first bridge to the actual technical infrastructure of how the job will be done.

In the Design Document, a flowchart showing how the software logic flows from step to step, or web page to web page if appropriate is included along with the narrative discussing that flow chart. If there are web pages or Graphical User Interface screens, mock ups of those will be included in the design document. Further details concerning the technical infrastructure requirements are presented including Data Base Management System, or DBMS, system requirements, table schemas, communications expectations and hardware or component specifications that have become a requirement of the design. In preparing the design document, take full advantage of technical expertise. In addition:

Take advantage of modular and object oriented design concepts. By designing the product in a modular fashion, common functions can be reused throughout the life of the project. By implementing object oriented design concepts at this early stage, appropriate hooks and interfaces that are even internal to the project can be placed into the design which will keep the development team from losing time later on due to redesign of the product to accommodate a modular approach.

Don’t forget about interfaces. In addition to internal interfaces between objects or modules, if the product must interface to external software products, place the design of those interfaces in this document and begin to identify where in the program flow for these will occur. If there are needs for technical interface information from an external vendor, the design team should gather that information so that when the final specifications are handed to the developers, much of what they need is already complete so speeding up the development.

Plan Ahead for Testing. The design document will serve two roles in the project life. It will lay down the product design at a greater technical level to begin to facilitate the developers but it will also lay down the hardware and environmental needs of the project for the rest of the cycle. As such, testing must be planned for. If the testing is to occur as the product is being developed, a schedule of development should be defined so modules are turned over to testers in a logical and streamlined fashion. A section of the design document should address testing categories and hardware needs which will eventually be used to design the testing regime when that part of the project gets underway.

Specifications

The detailed documentation that brought us into the development stage should make generation of specifications much smoother. The specifications document is what the programmers, data base subject matter experts and other technical team members will use to create the actual software product. It is written in technical language tailored for each developer. The logic programmers will have a set of specifications as will the web developers, the DBA team, the hardware and communication staff, etc. The document should be prepared in sections to address each technical team's orientation and include all the detail they need to begin work on the software production as quickly as possible.

Development

Once the development is underway, time must be allowed for the technical team leaders to do their work. There will be clarifications on the intent of the specifications and that is where the design, the scope and even the needs analysis will be helpful.

Concurrent with the systematic program execution, the project leaders will have placed a schedule into the planning documents that will detail the development time table.

The *traditional development plan* will include begin and ending date for the development and testing phases. This is a linear approach that defines development as a dedicated time during which design is done but testing has not begun. In that way, it is clear to the team and if done well results in a good solid product.

A second approach that might be of value is an *interactive development cycle*. Using this approach the design, development and testing phases occur concurrently. This is a much more dynamic approach in which "versions" of the software might be in design, in development and then on to testing while other parts or revisions are in other phases as well. The iterative approach is a good method to quickly get the developers to work and to develop rapidly and to give the design process a longer time for their efforts as well. However, it places a

greater demand on the managers and coordinators and should be approached with fore thought to assure the success of development.

Testing

There are four phases of testing each of which has a particular focus and satisfies a specific need. They are:

Unit Testing

This is the testing the developers do as the programs are written and made functional in the “programming labs”. Unit testing is ongoing until the developers are done. Even if subsequent tests send programming modules back to be fixed, the unit testing discipline is completed and documented.

Strategic Testing

Once the developers have tested their work, the project team tests the product in line with the design document and what the software is “supposed to do”. While the developers will focus on technical function, the strategic tests look more closely at user functions but still run through all logical paths of the software to find, document and find fixes for any program problems or “bugs”

User Acceptance Testing

The final phase in which a body of future users of the product give it one more run through before it goes into full production or to the market. These tests should not produce “bugs” but rather surface any functional flaws that must be corrected for the software to be viable.

Roll Out

Early in the design process, a roll out approach and plan will have been developed by the team leadership. If the product is going to a marketplace, that plan will involve mass production and distribution. However if it is going to a large number of corporate users and particularly if it is upgrading an existing software, plans must be carefully thought out for a seamless release of the product. There may be cause to consider a *limited roll out strategy*.

Another popular method of extending the testing cycle into roll out is to release the software in the form of *Alpha and Beta releases* which are actually testing releases. In this way the software gets into the hands of the users but there is still opportunity afforded for refinements and improvements. This is the method that Tometa prefers.

A methodology of this nature will always be most beneficial to the development process if done under the leadership of team leads or consultant talent that has back ground in the craft. If such talent is on staff, utilizing them in this way will greatly benefit the organization. If not, consider using outside help by contract. Tometa's trained and experienced project team members can provide any missing skill sets needed to make the SDLC method serve your project well. Whether it be to help with just a part of the process, the research and preparation of the needs analysis or requirements definition, or if it is the application of skills with tools such as Microsoft Project, Tometa can tailor the assistance to work with all or part of the process and assure that your development plan is executed efficiently and successfully.

This document is for informational purposes only. Tometa Software, Inc. MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

© 2004 Tometa Software, Inc. All rights reserved.