Designing Applications That Sell

by Greg Grisham

Whenever a programmer completes a new application, (barring extreme custom code) the natural inclination is to wonder how many more times he or she might be able to sell the same application. Sometimes it can be a turnkey application written for a particular business need, other times it might be a unique, critical part of an integrated set of programs that the original vendor overlooked or didn't address. In any case, the dream quite often has no bounds and expands to that piece of code becoming the next Quicken or Microsoft Excel. There's nothing wrong with that outlook, particularly since such a vision is what drives a programmer on those long bleary-eyed nights to deadline.

But what defines the applications that get scooped off the shelf at the local computer store? The answer is elusive, most likely because it is ever-changing. There are some criteria, however, that have remained constant over the history of commercial software, and there are also some emerging characteristics in the latest successes.

1. Address the immediate needs of the target market.

Obviously, if there were no mice, it would be reasonable that consumers wouldn't be shopping for mousetraps. In the same vein, it is important to solve the problem and not just present a partial solution. Imagine an Accounts Receivable package that couldn't print an Aging report. The first rule of any automation tool is to relieve or resolve some business need. Simply put, make that business function easier to perform. Quite often in the business community this results in the development of what has historically been referred to as a vertical application. The issue here is that the code may address a particular market, but it can constrain the scope of the marketable customer database. You can truly code yourself into a corner.

2. Make it scalable.

Don't turn away small OR large customers with your design. Woe is the developer that creates an Accounts Receivable application with a customer code limited to three-digit numeric input. It is important to be forward-looking in both your data structure as well as your application design. Remember, your customers have dreams too. While they might have 50 or 100 customers today, they will most assuredly hope to have many times that in the future. Make sure your application design doesn't preclude you from marketing to large enterprises. They buy software too.

3. Ensure usability.

Application ergonomics are more critical now than ever before. In the past, having menu-controlled software was considered "user-friendly" and sufficed. In today's world, taskbars and toolbars have replaced those menus to become a commonplace method of navigation. You need to have one-click access (or close to it) to the data, or the users feel they are working too hard. In a nutshell, GIVE THEM SOMETHING THEY CAN USE!! I use the emphasis here because if this gets overlooked, the rest is pointless. Remember, applications are tools, and the most popular tools are not wrought with gadgetry but are simplistic, reliable and effective.

4. Utilize Enterprise Information Portals.

Get personalized data via an Enterprise Information Portal. This expression is new and the concept is gaining wide acceptance. The basic premise here is that your data is accessible in such a fashion that the user can customize his or her queries and extract personalized data specific to the query. This doesn,t equate to a monogrammed report for every level of management, but rather it means customized extraction is possible with little or no specific interaction from the user. Your enterprise may have vast volumes of data, but your user has smooth, fingerprinted access.

5. Maintain market expertise.

Seek continuous training in your vertical market. In most any profession, there are continuing education programs. These are designed for the professionals in that arena to stay abreast of their particular market as well as the policies and laws affecting it. It follows then that application software designed for specific professions needs to be sensitive to the same changes and react accordingly. No one wants to reinvent the wheel every couple of years, but maintaining your code to keep it contemporary, and consequently competitive in the market, is critical. It is important to be sensitive to the nuances and issues of your chosen marketplace.

6. Understand technology and how it relates to your customers.

Simply put, their job is their job, yours is yours. Your customer might know everything there is about making and selling widgets, but they need to rely on you to deliver the application tools. A simple analogy I was given a number of years ago is still pertinent today. A baker will know everything that goes into constructing a cake but will have little or no idea about the value of an automated inventory system. This is really the definition of "value added" in the expression Value Added Reseller. Your customers want to focus on the cake and trust you to focus on the system.

It is important to be forward-looking in both your data structure as well as your application design. Remember your customers have dreams too. While they might have 50 or 100 customers today, they will most assuredly hope to have many times that in the future.

7. Practice quality assurance.

The expression "bug-ridden code" is hopefully a thing of the past. It is definitely an expression rarely used when describing popular software. Whether you purchase an application for your home or business, you expect it to facilitate that part of your life. You don't have time to be constantly uncovering flaws in the design or code of that application. Years past when a company purchased an application, it was expected that the company would maintain a close relationship with the vendor and quite often hired its own code-maintenance individual. Spaghetti code has gone the way of the Spaghetti Western. No longer is it acceptable in a programming shop for programmers to develop their own techniques. Object-oriented programming, with its focus on standardized internal interfaces to data and the way it is manipulated, has helped immensely here. Quality Assurance departments are now as prevalent as development groups.

8. Get "Web enabled."

This is clearly the most popular of the current buzzwords. Developers worldwide face a query on this constantly. End users are rapidly becoming a web-savvy group and want to know how an application they may purchase will interact with the Internet. Other users will speak directly to the features linking to their intranet. In either case, Web connectivity, as a bell, whistle or integral feature, is something all future applications must address.

In summary, it used to be that you had to bring the user to the application. Any function within an application that they couldn't readily perform was classified as a training issue or software bug. In today's world, you need to bring the application to the user. The computing world is evolving on a number of fronts that will either *drive* future application features or *pull* applications into the 21st century.

Joining BASIS in 1999 as a Product Marketing Specialist, Greg Grisham aids the company in all aspects of customer relations and helps to determine the position and focus of BASIS products in the marketplace. Greg began developing Business Basic applications 19 years ago. Since then, he has headed technical support departments and designed, developed, purchased and implemented business systems and applications in facilities across the United States.