

---

# 4D for Fun and Profit

## Should You Create a Vertical Market Product with 4D?

by Walt Nelson

---

### Should you create a vertical market product with 4D?

As you know, the theme of my regular *Dimensions* column is “4D for Fun and Profit.” In this month’s column, I hope that I can persuade you to seriously consider increasing your 4D fun and your 4D profit by creating a vertical market product based on one or more custom applications that you have developed in 4D.

What started me to thinking about this subject? A couple of weeks ago, I realized that a large percentage of my professional life is currently involved with vertical market products based on 4D: I released Docu-POWER! a few weeks ago, and I am currently working on two other vertical market products: one for a client, and another for my own company. Although the process of creating and maintaining vertical market products is not easy, I must say that overall, doing these products in 4D V6 has been a pleasant experience.

### Vertical market product: a definition

When I say “vertical market product,” I mean a product that is sold as a specific, packaged solution for a specific, targeted market segment. Some examples of specific targeted market segments might be: an Optometry Practice Management System, a Property/Casualty Insurance Agency Management System, or a Documentation Management System for Software Developers.

It is very important that a vertical market product have a specific target: a market for which the product solves specific problems that are not being solved—or are not being solved as well—by other competing products.

With that definition of a vertical market product in mind, we will discuss three major ideas in this article:

- Advantages of a vertical market product to the customer
- Advantages of a vertical market product to the developer
- Advantages of 4D as a platform for creating vertical market products

### Advantages of a vertical market product for the customer

Steven Covey, the author of “The Seven Habits of Highly Effective People,” says that for any product or service to enjoy long-term financial success, that product or service must be based on the concept of win-win. The deal must be a good deal for the customer and a good deal for the vendor. That makes a lot of sense to me; so, before we look at the “win” for you, let’s look at the “win” for the customer. We will be assuming here that the customer has already looked at the off-the-shelf software available on the market, and the customer has decided that their needs are so specialized that a general off-the-shelf program, such as an Accounting program or a Contact Manager, will not be sufficient. The customer needs a specialized solution; the question now is how to get it: vertical market product, or custom development? If such a product is available, a vertical market product will probably be a better choice. Here are the reasons why I believe this is so:

- A vertical market product takes advantage of the knowledge and experience of previous customers. Although it is true that every organization is somewhat different from all other organizations, it is also true that organizations may be grouped into categories, and the organizations in each category have certain common needs. If your product has already been tailored to meet the most of the needs of the customer’s category—because of development and enhancements for previous customers—then that product adds value that might not be available in a custom development project.
- A vertical market product is ready for use right away. Custom software takes months or years to develop; in the meantime, the client’s special needs are not being satisfied.
- A vertical market product is less expensive for the client than custom development. Since the development has already been done, you can afford to charge less for a vertical market product than you would have to charge for custom hourly development. (By the way: you may be thinking that a vertical market product is not always the least expensive solution; an off-the-shelf generic package costs less. Remember: we are assuming that the customer has already looked at generic software packages and decided that none of the generic packages meet the need.)
- A vertical market product usually has fewer bugs than a custom solution, simply because you have had more time to

test and debug. This can be a significant benefit for the customer.

- As the provider of a vertical market product, you have a better understanding of the customer's business than a developer who has not spent as much time studying the customer's category of business needs. This, again, is a benefit for the customer.

## Advantages for you

Now let's look at your side of the win-win equation. When you start with an application that you have already developed—and then turn that application into a vertical market product—you gain several advantages:

- *A vertical market product gives you residual income.* Robert Allen, in his course "Multiple Streams of Income," makes the point that in order to become financially secure, we need residual income—income that keeps coming in while we are out doing other things. He calls this residual income, or "making money while you sleep."
- *A vertical market product makes it possible for you to take time to build multiple streams of income.* As a consultant, you definitely need multiple streams of income. Nothing is forever; consulting jobs begin and consulting jobs end. Multiple streams of income are the only way that you can guarantee yourself financial stability during those periods when you are between contracts.
- *A vertical market product gives you an opportunity to develop some specialized expertise.* Specialized knowledge about a particular type of customer's needs is usually more valuable than general knowledge about general customer needs. If you know a customer's business and you can walk into their office and "hit the ground running," you are far more valuable to that customer than a developer who did not know their business. For that developer, that customer would have to spend time explaining the business and how it works.
- *With today's advances in technology—writable CD-ROM's and the Internet—producing and supporting your product is far easier than it was in the past.* You can produce a very professional-looking, cross-platform CD-ROM in your home office. The Docu-POWER! CD-ROM, which is hybrid Macintosh and Windows with a multi-color labels and jewel case jackets, was produced in my home office with a \$500 CD-ROM duplicating machine and a \$250.00 color printer. Internet support is even more economical. For a \$200 initial investment and less than \$30.00 a month, I set up my own virtual-host web site ([www.WaltNelson.com](http://www.WaltNelson.com)) that gives me up to ten e-mail addresses, and allows me to maintain up to 30mb of files. This hosting plan allows customer downloads (demo's, updates, order forms, etc.) of up to one gigabyte per month. If I need more space or more download capability, I can upgrade my web hosting plan for a few more dollars per month. Because of these advances in technology, there

has never been a better time to create and support your own vertical market product.

- *A vertical market product can be simpler to support than a custom development project* for two reasons:
  - a) The application has fewer bugs than a custom developed application.
  - b) You already know most of the questions that will be asked, because you are already familiar with many of the challenges of the customer's line of business.
- *A vertical market product can be easier to sell* for several reasons:
  - a) You already know quite a bit about the business, so you can speak the customer's language.
  - b) You have the best kind of references: other organizations with needs similar to the needs of the prospect.
  - c) The underlying product (for example, 4D versus Visual Basic) is not as important an issue as it could be with a custom development project. If the vertical market product is specific to the customer's needs, then the issue is not nearly as important as it might be in custom development situations. If you have something that solves their problem, that is likely to be the key factor. Period.
- The total cost to you is far lower than custom development, because you have already been paid for a large portion of the time that you put into the project.
- If the customer wants to further customize a vertical application, you can do that—and in the process, you create another income stream!

## 4<sup>th</sup> Dimension as a vertical market product development tool

I hope that I have convinced you that creating a vertical market product—based on work that you have already done—might be a sound idea. The remaining question is this: "Is 4D a good choice of a tool for vertical market product development?"

Several weeks ago, when I realized that more than 85% of my 4D time was being spent on projects that involved vertical market products written in 4D, that realization was something of a shock. In the past, I had never intentionally spent such a large percentage of my time on products (as opposed to consulting). So I thought that I had better look at 4D from the vertical market perspective, and ask myself if this platform (4D) was suitable for the job.

I was pleasantly surprised at how well 4D fits the description of an ideal tool for vertical market development. Here are my observations:

- *4D is powerful and flexible.* In terms of relational database needs, there are many things that 4D can do, and there are

very few things that 4D cannot do. This means that you can provide features that will satisfy a variety of the user's needs. (Granted, it takes several years of experience to get to the point where you can take full advantage of the power and flexibility of 4D. However, the same is true for any powerful software development tool.)

- *4D provides several rapid application development (RAD) tools.* The form wizard, the picture library, the form editor, the object alignment tools, the quick report wizard, and 4D Insider are just a few of the RAD tools in 4D. Why is this important? Because the longer it takes you develop a product, the less the net profit. Also, if the development cycle is too long, market conditions may change—making the product obsolete—before you can bring a version to market!
- *4D scales up relatively easily from a single user up to several hundred users.* This means that you can offer your vertical market product to the full spectrum of the market: from a small one-person shop, up through an organization of several hundred employees. This also means that you can grow with your customers. As companies like Quicken, Microsoft, Oracle, and Apple have clearly demonstrated, the “real money” is in repeat sales: upgrades and additional sales to existing customers. This repeat business is a profitable form of residual income: the cost of sales is close to zero, and therefore the profit margins are very high.
- *The company behind 4D is stable.* Like most companies in the software business, ACI has had its challenges. But after more than 12 years and several market cycles, the company is still around and again making a profit. That in itself is an accomplishment in this volatile software business!
- *4D is constantly being improved and updated.* Sometimes we take this factor for granted, sometimes we even complain that things are changing too fast. But consider the alternative: one law of nature is that a thing is either growing or dying. If the rate of change and growth in 4D ever stops or slows down considerably, that will be the beginning of the end—as it was for dBase, VisiCalc, Omnis, Comuserve, and many other products and services that were once dominant names in this business.
- *At least once every two years, we get a stable version of 4D.* This is actually pretty good, when you consider the fact that 4D is a highly complex product, and it is constantly being upgraded and improved. Historically, the stable U.S. versions of 4D have been 1.0.6, 2.0.11, 2.2.3, 3.0.5, 3.5.3, and 6.0.5—six versions in 11 ½ years, or approximately one stable version every two years. If you think about it, you will realize that the two-year time frame for a stable version of a large application is not uncommon. Think about the Windows operating system, the Macintosh operating system, Sybase, Oracle, and other complex software that you use, and you will probably see a similar pattern: a stable release about every two years.

- *4D data automatically adjusts to structure updates.* Even though you might only release major versions every two years, you should release minor updates every few months. 4D developers take the automatic updating feature of 4D for granted, but other competing products require export/import of data and structure objects whenever you send an update to the customer. This is just one more thing that can go wrong—something that you don't need when you have customers spread out around the country or around the world.
- *4D is Cross-platform: Macintosh and Windows.* Over the past couple of years, while Apple Computer was on a downtrend, the cross-platform capability of 4D was not looking like a competitive advantage; however, with Apple now rising from ashes like the Phoenix, cross-platform capability might once again become a major selling point. This capability is an advantage because one customer has the ability to run your application on both platforms; it is also an advantage because you can sell your product to customers on both platforms. If they like Windows, you can sell them a Windows version; if they like Mac's, you can sell them a Macintosh version. If they like both, you can sell them a cross-platform version. You do not have to try to impose your platform preference upon the customer. There are very few vertical market products that can make that claim.

## Some do's and don'ts

The arguments for a 4D-based vertical market product are very strong, aren't they? Yes, I think so too. To make this a profitable strategy for you, here are some suggestions that you should keep in mind:

- **Do** let someone else fund the initial development. Unless you have lots of experience and lots of money, you should not try to create a vertical market product from scratch, with your own money. Start with one or more applications that you have already developed as custom applications, and build from there (Note: make sure you don't violate any contractual terms or implied agreements that you had with those previous clients!).
- **Do** target a specific market. Don't bite off a larger market segment than you can chew. Remember: the more specifically your product solves a customer's problem, the easier the product will be to sell. You need to find a balance: a product specific enough to be attractive to the customer, but a market large enough to justify the cost of producing the product.
- **Don't** expect too much. A common mistake is for a developer to get excited about a creating a vertical product and, in his or her enthusiasm, “throw out the baby with the bath water.” When the inevitable happens, and the actual sales do not meet those unrealistic sales projections, the developer becomes discouraged and gives up. Wrong, wrong, wrong. The correct way to approach this vertical market opportunity is this: think of it as another stream of

income to add to your existing streams of income. If you can earn (for example) three thousand dollars a month with less than 20 hours a month devoted to the project after the product is released, then you can consider it a success.

- **Do** keep your expenses low, low, low. The ideal strategy would be to spend no money at all—just your own “sweat equity”—but that is not a realistic goal. You need a duplicating machine to produce the CD’s (assuming you are selling your product on CD); you need a telephone line; and you probably need a web site. What you do *not* need, in the beginning, is a payroll expense. One of the quickest ways to eat up the profits of a small business is to hire employees. Structure this side business so that if you need help, you hire contractors—not employees.
- **Do** make it a habit to be very conservative about upgrading to new major versions of 4D. You should release the first version of your product using a tried-and-true, stable version of 4D, and then continue maintenance releases based on that stable version. If you have to wait the full two years for the next stable version, do it. ***This is a good rule, no matter what tool you are using: C++, Visual Basic, Powerbuilder, Java, or whatever.*** What you do not need is a set of support headaches that are beyond your control, because they are the result of a less-than-stable version of your development tool. If you jump into a new version of 4D because it has some “cool” features, and that version turns out to be unstable, don’t blame ACI; blame yourself. You knew that stable versions can be as much as two years apart; I have told you in this article, and I backed it up with historical fact. If you don’t listen, then I have no pity for you.

## Summary

I hope that I can persuade you to seriously consider increasing your 4D fun and your 4D profit by creating a vertical market product based on one or more custom applications that you have developed in 4D.

Assuming they do the job that the customer wants done, vertical market products offer you several advantages that make them a very good source of income—in addition to your custom development revenues. Indeed, the best of both worlds would be to find clients who retain you for custom development; then, without violating your contractual or implied agreements with those clients, create vertical market products that serve the client’s specific target market.

Finally, to “close the loop,” sell your custom development services to the vertical market customers. In this way, you can build multiple streams of income and “make money while you sleep.”

## About the author

Walt Nelson is a 4D developer, trainer, and troubleshooting consultant. He has been working with 4D since the Silver Surfer beta test project of 1987 including four years at ACI US as the Manager of Developer Services and the Director of Technical Support. He is the author of 4D for Fun & Profit: A Handbook for Professional 4D Developers, and of the Docu-POWER! Document Management System. He can be reached at [walt@waltnelson.com](mailto:walt@waltnelson.com).

## On this issue’s examples disk

*Y2KIssues* — A folder containing text files of the methods shown the article “Y2K Issues” by Ed Heckman and Joe Batts.

*ServerTools* — A folder containing 4D v6 Macintosh and Windows example databases to accompany the article “4D Open Access Control” by Peter Bozek and Miloslav Bystricky.

*GenericPicker* — A folder containing 4D v6 Macintosh and Windows example databases to accompany the article “A Generic Picker” by Mike Kerner.

*TabsinTabs* — A folder containing a 4D v6 Windows database to accompany the article “Tabs in Tabs” by Martin Bjorkman.

*ACI\_Video.sea* — A archive containing a 4D v6 Macintosh example database to accompany the article “Simplyfing Report Selection for Users” by Dani Beaubein and Guy Algot.