From Program to Product: Turning Your Code into a Saleable Product

ISBN-13 (pbk): 978-1-59059-971-6

ISBN-10 (pbk): 1-59059-971-3

ISBN-13 (electronic): 978-1-4302-0614-9

ISBN-10 (electronic): 1-4302-0614-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editors: Jonathan Hassell, Dominic Shakeshaft
Technical Reviewer: Martin Reid
Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Kevin Goff, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Beth Christmas
Copy Editor: Ami Knox
Associate Production Director: Kari Brooks-Copony
Production Editor: Kelly Winquist
Compositor: Dina Quan
Proofreader: Lisa Hamilton
Indexer: John Collin
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `http://www.apress.com/info/bulksales`.

CHAPTER 1

Who Wants to Be a Millionaire?

Most of us harbor a secret fantasy about being in business for ourselves and raking in a lot of dough. Some of us—like myself—are doing it and are halfway there—we're in business for ourselves.

In the last fifty years, rock star aspirations aside, a popular dream is to be a software entrepreneur who writes "The Next Great Killer Application"—a program that will cause a river of easy money to flow into your life.

What you have is an idea for a computer program that you think would be a popular product. The state of your program may be anywhere from a half-baked scheme to a smoothly functioning program.

What you don't have is a way to get from idea to reality.

You might already have crossed the finish line and be out in the free market hawking this gem everywhere you can think of, and are perhaps not meeting with a great deal of success.

Turning an idea into a professional-looking, saleable product involves a lot more than just writing a good program. There are packaging, and marketing, and legal, and organizational, and technical issues to consider.

And you are probably feeling that you don't even know the right questions to ask.

So Why Did I Write This Book?

I'm a programmer. I have been since I was a teener. Oh, I've had lots of job titles, but at heart I'm just a geek programmer. Right now, I write custom databases and applications for a variety of small businesses using Microsoft Access exclusively, under the company name of Beach Access Software (`www.bchacc.com`). It's fun. It's satisfying. And having been doing this on my own on a variety of platforms for nearly 30 years, I'm pretty much unemployable.

One day I got a referral from a colleague to a fellow in Dallas—Jack Stone. Jack is a patent and trademark attorney at Scheef & Stone who had developed some software in-house to help keep the legal matters under control.

Then, being both smart and ambitious, Jack decided that this program would be attractive to a lot of law firms like his, so he hired a programmer and turned his attention to generalizing the application—making his highly customized, home-grown code into a program that would be usable by a lot of other patent and trademark attorneys, and maybe even other kinds of law firms. The program is called DocketWorks™ and, as you can see, Jack has it already trademarked. (Jack and I will talk about trademarks in a later chapter—what they are, how to get them. Jack will give you his business card.)

This, of course, is how many, if not most, new software products begin life—as a highly customized application that the creator wants to make into a best-seller.

After a couple years of development, Jack had become disenchanted with the programmer he had hired, and so Jack was looking around for a replacement. At my request, he e-mailed me the current database and application so I could give him my opinion of the state of the program and what it would take to get across the finish line.

After I looked at it, I had some good news for Jack. The database—the tables where the actual data was stored—was pretty well designed. And in the program itself there was a lot of functionality present.

But the bad news was 1) a lot of buttons weren't hooked up yet (that is, there was much functionality that was not yet implemented—doors leading to rooms that hadn't been built yet), and 2) the forms and reports were not very attractive. They were functional but, well, ugly. No color, controls all crammed together—hard to read and not intuitive.

By the way, I did not charge Jack for this analysis. To me, the first look at a potential application is a job interview. So if you find yourself in the position of having to hire a programmer, my first piece of advice (among many to come) is this: you can make your own judgment about programmers who charge their regular hourly fee just to see if they want to work for you.

My judgment is that it is not appropriate and should raise a red flag in your mind. Maybe they're so successful they need to charge for sales calls just to keep the riffraff away. From that kind of programmer you may not get the attention or timely response you need. Maybe they think that every word that drops from their lips is a pearl of great price. In any event, be cautious about paying someone just to look over your stuff.

Anyway, after looking over the database that he had had designed and the state of the program that he had had developed, I asked Jack for a kind of roadmap document on the design and implementation plan for the product to get a sense of just how far he'd come and how much road was ahead. Jack didn't have that, but he had a database he developed himself of some 150 action items—things the program needed, with priorities, subpriorities, and status. But this was more of a task list of unrelated items—some small, some big. And it wasn't the high-level plan I was looking for.

So I began to ask Jack some probing questions:

- What is the target market? Exactly who needs your product?
- What problem does your product solve?
- How are you going to price your product?
- Will you package it or distribute it electronically?
- How are you going to reach the people who would want it?
- How are you going to protect your software from illegal copies?
- Who is the competition?
- What advantage do you have over them?
- How will you track your leads and sales?
- Will you be asking for annual support fees?
- Speaking of support, how will you provide it—both technical and operational?
- Do you have a manual—planned or in process?

There were a number of technical issues as well:

- Will you use fixed screen resolution or form resizing?
- Will you require users to have Microsoft Access, or will you give them a runtime version?
- Do you have a consistent look for all forms and reports?
- Do you have a logo that appears in the same place on every form and report?
- Will you be using floating forms or full screen?

On and on it went.

And the answers to these questions were mostly that he hadn't yet really thought about these things. His focus was on making the software work. That's not a bad motivation, but it's kind of like starting to build a house without an architect and figuring to solve the problems as they arise. (If you find yourself in this position, you're reading the right book. You will need to reverse the process—to go back and do some serious design work before writing any more code. Keep reading.)

So I opened up a Word document and began to organize an outline of all the action items I could think of that Jack needed to do or think about in order to turn this raw program into a saleable product.

I kept that Word document open for three days. Every time I thought of something Jack needed to consider, I added it to the document. And at the end of that, we had a kind of rough roadmap to the finish line.

Looking over that document, I began to think about all the pain I suffered learning how to ask and answer the many questions that Jack hadn't considered when designing a software product for sale. I thought about all the great ideas for programs that never made it to the product stage or failed aborning for lack of good answers to these questions. I thought of all the folks out there who have a custom application they think would make a great product or just an idea for a piece of software they think would be a winner. And I thought about the path from idea to product and the big black box in between—a box that includes product design, development, testing, and packaging.

I looked over Jack's outline again and thought "Wouldn't this make a great book?"

Why Am I a Programmer?

As I noted at the beginning of the chapter, I'm a programmer. I was a hacker before it became a criminal activity. A geek. A propeller head. I got my first exposure to computers as a high school student. But in those days, there were no computers in high school. To get near a computer, I had to take a weekend course at the Illinois Institute of Technology in Fortran programming. But I got to write and run simple Fortran programs on the university's massive second-generation IBM 7090 computer. It was behind a glass wall in a temperature-controlled environment. And it had a small fraction of the computing power of that three-pound notebook in your briefcase.

I wrote the programs for it on punched cards with 24-hour turnaround on each run. A misplaced comma meant a lost day. It was, viewed from today, awkward, expensive, clunky, and inconvenient.

But I thought I'd died and gone to heaven. There was something magical about the whole computer thing. I was starting to get hooked.

The college I went to had a computer, too. One. A lovely old IBM 1620 with actual hard drives and a printer the size of a Volkswagen that sounded like it was breaking concrete. It was replaced in my sophomore year by an IBM 360/40—a big box with about a thousand blinking lights on the front. I hung with the IBM engineer, and by the time school opened in the fall, I was the only one on campus who knew how to start and run the beast.

So the director offered me the job of systems manager, apologizing because it didn't pay much. "Pay?" I thought. "They'll pay me for what I'd do for nothing?" Now the hook was in but not yet set.

In my junior year, I decided to go for a proficiency grade in one of the school's few computer classes—do a big project, show you know the stuff, and skip the class work. So I wrote a program. Never mind what it was. It was just

a utility program providing a feature for Fortran programmers that IBM hadn't provided. And I got my grade.

Then, I guess because I was a business student and kind of in training to be an entrepreneur, I decided to see if I could sell a couple of these jewels. But I couldn't afford to advertise, being a student, you know. So I sent out a press release to the few computer magazines that were in print at the time.

It got picked up by *Datamation*, and the staff gave me a reader service number so people could circle it if they were interested in my product (no Web at that time—it was 1968 and Al Gore hadn't thought of it yet). They would send me the leads (on sticky labels). I followed up with a letter and an order form.

For $40 (which was a couple of weeks' worth of student groceries in those days) the buyer got a deck of IBM punched cards with my program on them.

Imagine my surprise when people began sending me money. The hook was finally set. I got to play with the world's best toys, and people pushed money at me for doing it. How could it get any better?

Well, it did. My dream was to be rich enough and powerful enough one day to own my own computer. You're laughing. Yes, I now have seven, and a few more in the garage. Computers have evolved from expensive, remote, highly specialized machines, to appliances that almost everyone has and needs. In fact, they've become so ubiquitous, they are now an environmental waste problem.

But as soon as Radio Shack came out with that TRS-80 Model II computer around 1980, I moved from being gainfully employed to being a full-time, independent, self-employed slacker playing all day with microcomputers and thinking about how to get people to send me money as a by-product of all that play time.

The first thing I did was write a program for my wife, who wanted to start a business tracking sales leads for companies and sending out their product literature. That worked out pretty well. I was beginning to think I'd never have to have a regular job again.

Then, the fellow I was sharing office space with—a guy who, like me, was pretending to work while playing with these new personal computers—decided that a micro-based (we called personal computers "micros" in those days, until IBM came along and standardized our language) critical path project management system was something the world needed. Boy was he ever right about that.

The first products were sent out on 8-inch floppy disks and included a manual that was printed in uppercase on a dot matrix printer, with no real packaging. The program was called PMS-II (a lot of software in those days had the -II appended to it—a tribute to the pervasiveness of the Radio Shack TRS-80 Model II microcomputer). And we charged $3,000 a copy.

Those were heady days. There were no set rules for product creation and distribution. We made them up as we went along. So we couldn't really make any mistakes.

And the checks began to arrive.

Of course, eventually, Microsoft came out with Microsoft Project for $69, and the game was over.

But the hook was finally set so deep it would never come out.

Over the years, I have developed and sold other applications. A couple of them I'll use as examples in this book to illustrate points I'm trying to make.

The Lone Ranger Rides Again

One product I'll often refer to is a venerable old manufacturing system named E-Z-MRP (`www.e-z-mrp.com`), designed for small manufacturers. Originally programmed in a DOS-based language on the old IBM PC, it was reinvented several years ago as a Windows-based product written in Microsoft Access. So most of the examples I use in this book are based an applications developed in Access. However, this book is not about programming but about creating a product. So, regardless of the programming language you use to make your product, this book will be both useful and pertinent.

By the time I decided to leave the world of gainful employment, I had, through no deliberate plan of my own, gained a lot of experience in manufacturing systems. One of the painfully obvious things about manufacturing systems was that only larger operations could afford the investment in dollars and time to implement manufacturing software.

But after the project management adventure, the rules had changed. I thought to myself "If a simplified interface could be designed so that the complex problem of controlling manufacturing could be understood and used by the lowest-level person in a manufacturing business, I could go after the bottom 90 percent of the market where there's no competition."

It took real hubris to create a product that everyone in the business said was impossible. But then entrepreneurs are not known for their humility. Or their aversion to risk.

Now, to create a good software product generally takes two skill sets. You need someone who knows how to code up a smooth, graceful, effective program. And you need someone who understands the application area.

In the case of E-Z-MRP, I happened to know both sides of the game. So I went alone on that one.

It Takes Two to Tango, Sometimes

Another product I'll refer to throughout this book is The Sleep Advisor (`www.thesleepadvisor.com`)—a program for consumers that identifies sleep problems and provides remedies for solving them.

As I noted before, to create a good software product generally takes two skill sets—someone who knows programming and someone who is an expert in the application area. In the case of E-Z-MRP, I could sit on both sides of the table. But I didn't know squat about sleep or sleep problems. So as you can guess, it wasn't my idea to create this program.

In 1994 a colleague called me from Tucson with an interesting proposal. He was a clinical psychologist who had developed a unique expertise in the field of sleep disorders.

"Rocky," he said, "I'm doing these sleep consults every day." He explained to me, "I'm asking the same set of questions. I'm coming up with a predictable range of diagnoses, and offering people a finite set of recommendations for solving their sleep problems. Couldn't we write a computer program to do this?"

Being a software developer, I knew this would be much more challenging than it sounded. Being somewhat of a smart aleck, I responded, "Sure. Just tell me what the questions are that you ask, what the conditions are that you diagnose, and (here's the kicker) what the links are between the questions and the diagnoses. And violà! We've got ourselves a program."

So I went over to Tucson, we spread out the butcher block paper, and started to diagram what would eventually become The Sleep Advisor. After two days, we gave up. Translating the qualitative, intuitive approach of a clinical psychologist into a computer program was just too daunting.

But the idea wouldn't go away. We kept coming back to it. I had my doubts that it could ever be done—that a computer program could accurately identify sleep disorders based on the answers to a questionnaire. But we kept at it, off and on, for over ten years. And we finally got it to work. You can see the result at `www.thesleepadvisor.com`.

Partnering has distinct advantages. If you have expertise in the field that your program is made for, but you're not a professional programmer, you should try to team with a good programmer. If you're a programmer, you'll be way ahead of the game to join with someone who is an expert in the field.

Teaming has its obvious downside. Lone rangers don't have much problem with interpersonal conflicts. A partnership is like a marriage in many ways. You have to like each other, have good communication skills, and be willing to compromise what you think is right or best to achieve the goals of the partnership.

Of course, you can always buy the skill set you're missing. As a programmer, you can consult with experts in the field and pay them for their advice. If you're an expert in the field, like Jack Stone, you can hire the software design and programming skills you need.

Who Are You?

So, do you want to be a millionaire? Do you have the dream—like Jack—of taking that custom application you developed (or haven't developed anywhere yet but in your imagination) and turning it into a product? Do you have a vision of sitting back and opening the morning mail and sorting out the checks? Do you see a river of money flowing into your PayPal account?

But are you stymied, intimidated, or otherwise reluctant to start down that long, dark road?

You are not alone—in either your dreams or your misgivings. But as the old Chinese saying starts "A journey of a thousand miles . . ."

One of the beauties of software is that you don't have to mortgage your house to develop it. It's largely sweat equity. But, in my experience, more people fail to realize their goals because they fail to start, rather than fail along the way. One of the dark sides of being a software entrepreneur is that you have to be self-motivated to be self-fulfilled. There will be no boss other than yourself who drives you to meet the goals and deadlines.

So you need to start. And I'm hoping that this book will give you the information and confidence you need to guide yourself from an idea to a product in-hand that's ready to sell.

What's the Big Idea?

As I pointed out earlier, at this point in my "career," I happen to favor Microsoft Access as a development platform. I like the kind of Goldilocks applications that it's good for—not too big, not too small.

But programs come in all sizes. You might have an idea for a simple $39 utility program. Or your idea might turn into a $39,000 enterprise application. It may be a *horizontal* application—one which is used by many people, consumers or a wide variety of businesses, crossing many professional disciplines. Or it may be a *vertical* application—targeting a specific group of users.

Selection of your development platform—the language you choose for your program—is up to you. C, HTML, Access, SQL, Oracle—they're all good choices for the right program.

But regardless of the application, all software products have many things in common. And it is those common, generic topics that I have tried to cover in this book—answers to the kinds of questions that I asked Jack Stone. So that regardless of what the size and nature of your program is, and regardless of where you are in the process—not even in the starting gate or already out there flogging your software—this book will help you to reach a successful outcome and steer you around the many hazards and pitfalls that lurk unseen on the road to your first million.

What This Book Is Really About

This book is really about getting to your first day in business.

It is about taking that raw idea you have for a program and creating a saleable product. And setting up the support systems you need to make your business hum.

If you're going to do this thing, you have to be ready for some tough days. But nothing worthwhile ever comes easy. Sometimes what it really takes is stupid, blind obstinacy—the drive to keep going, stubbornly solving one problem at a time until you get the brass ring.

Once I had a partner who had one of those ghastly motivational posters hung in our office. You know, the kind that promotes leadership and teamwork and quality through the use of tired old clichés and kitschy art. This one was no exception. And yet its corny McCuen-esque message stuck in my mind and has provided a handy boost to me many times over the years. It was a picture of a three-masted sailing ship threading its way between rocky outcropping and small islands. And the message:

Obstacles are what you see when you take your eye off the goal.

How They Did It: An Interview with Software Developer and Entrepreneur Al Vanderpool

I partnered with Al Vanderpool at the beginning of the personal computer revolution and had a hand in the development and marketing of what was then one of the first commercial applications for personal computers, a critical path project management system named PMS-II—the name reflecting the popularity of the Radio Shack Model II computer, which was the state-of-the-art machine at that time.

Although he is no longer developing and marketing software, Vanderpool's depth of experience in the software industry creating profitable products and businesses makes his a voice worth listening to. Pay particular attention to the advice he has at the end of the interview for the wannabe software entrepreneur.

Smolin: You're a software developer.

Vanderpool: That's correct.

Smolin: And, how did you become a software developer?

Vanderpool: Well, I've been in computers since the early '60s with General Electric and I was doing a lot of technical support, and then I had the opportunity to go on my own, and lo and behold I wound up writing software. So it was not a conscious decision, it was an afterthought.

Smolin: So, the first commercial program you did was . . .

Vanderpool: PMS-II.

Smolin: And that was in, if I recall, around 1980?

Vanderpool: Yes, late '70s, early '80s.

Smolin: And that was a critical path project management system.

Vanderpool: Right.

Smolin: Why did you want to create a critical path project management system? Where'd the idea come from?

Vanderpool: Interestingly enough, I had bought a Radio Shack Model II computer when they first came out, and I was looking for something to play with on it, just something to do, and we decided to track our fertility cycle in the hopes of creating a girl.

Smolin: You and your wife.

Vanderpool: Yeah. So I wrote some date functions and some methods of calculating forward and backwards from dates and coming up with the right things. And then I thought, "Gee, this is the heart and soul of a critical path project management system," which I'd had a lot of experience

with at General Electric and Honeywell in the past. Using, not creating. That was the nub of the idea, and from that I started using it in my consulting business for very simple scheduling. And then it started to grow and started to get embellished. I created reports from it, and I was using it myself in my consulting practice. So that's how it all came about. And then I met Rocky Smolin, and he published an article about it in one of the computer rags at the time called *Information Age*, and the phone started ringing, and we decided to go ahead and make it a full-fledged product.

Smolin: So you wrote this in which language originally?

Vanderpool: Originally it was written in Microsoft Basic, and then it was converted to a Digital Research language so it could be compiled to protect the source code.

Smolin: C-Basic.

Vanderpool: C-Basic or CB86, a 16-bit BASIC. And then to make it all happen, because languages and supporting software and utilities were pretty crude and pretty user-unfriendly at the time, I added a library of functions to it, replacing the original stuff. That library was from a company called Minnow Bear, or something like that. It had all kinds of expanded capabilities to it which really made the program capable of doing what it needed to be done, which was managing construction projects for larger construction firms, RCA being the first user.

Smolin: So this was the first product you commercialized?

Vanderpool: This was the first commercial product that I ever created.

Smolin: Who was the target user? Before you started to sell it, who did you have in mind to sell it to?

Vanderpool: The original idea immediately went to the construction industry because the need was there and they didn't have any tools, or all the tools they did have were extremely expensive and very hard to use, and so they didn't do what they should be doing with them. So the thought was to give them a lower priced, easy to use desktop computer capability so that they'd be able to do a much better job. That was the target market at the time.

Smolin: And the Radio Shack Model II was the state of the art at that point.

Vanderpool: That was the Ferrari of the day.

Smolin: So you sent the product out on 8-inch floppy disks?

Vanderpool: 8-inch floppies, yeah.

Smolin: How long did that product run, how many years did you flog that jewel?

Vanderpool: Probably, easily ten to twelve years of product life on it. Mostly because of the add-on capabilities and additional products that it gave birth to that were supporting it—resource management, graphics,

production support, materials management, reporting systems, that type of thing.

Smolin: As the technology evolved, did you change platforms, or was it always C-Basic DOS?

Vanderpool: It was always C-Basic DOS, with more capabilities as Windows came online, more features were added to the library to take advantage of what was available in Windows, to whatever extent we could. Windows was pretty crude at the time.

Smolin: What eventually stopped the sales, or why did you stop selling it?

Vanderpool: I stopped selling the product as a conscious decision because the support revenue from it was exceeding the sales revenue for it, and I saw no sense in spinning the wheel of advertising for zero to negative gain. So by stopping the product sale and concentrating on end-user support and relicensing and annual support contracts, it produced a revenue stream for many, many more years until I finally just told everybody, no you can't renew, and no I'm not going to answer the phone anymore. That was the end of it. That was probably mid '90s before that ended. So it had a complete run of maybe fifteen, sixteen years, which is extraordinary for sure.

Smolin: You went on and did a second product?

Vanderpool: One of my PMS-II users from the Los Angeles area who had taken the course at Brigham Young—they used to teach PMS-II at Brigham Young University graduate curriculum for construction management—and this young guy was really a cost estimator for a construction company who saw the program, and he called me one day and said, "Hey, I took your class. Is there a way I could come down and talk to you? I have an idea." So I said sure.

He comes down and he starts talking about the problems that the construction companies have in soliciting bids and getting responses from bids. Most of the general contractors had stopped doing anything of a specialty work, they were just contracting. And they needed some support with cost estimating. So he laid out this idea for a fax machine–driven application, and I said that doesn't sound too tough. Give me a fax card that goes in a computer and I'll figure it out.

About six weeks later, we had a product that allowed a contractor to specify what trades were needed on a particular project and then go through their database of subcontractors, select all of the local subs that met the criteria, and send them a fax inviting them to bid on this project. That product was called BidFax. It was a wonderful title. It became the Coca-Cola of the industry. People would say, "Did you say get a BidFax on that?", and that's when you know you've done something right. It was a very, very good product for us. I sold that to McGraw-Hill about 1998, I think.

Smolin: You said it was about six weeks to get it from conception to where you were ready to sell it.

Vanderpool: Essentially, get it from conception to Beta testing so that we could take it to a live construction company and let them use it on a limited basis.

Smolin: How long was it in Beta?

Vanderpool: I'd say probably about two months with revisions and enhancements before we felt like we had a saleable product.

Smolin: What language was that one in?

Vanderpool: That was also written in CB86—the same thing that PMS-II was done in, utilizing the libraries that I'd already developed there, so it was fairly easy to construct a new application using the framework that was already there.

Smolin: Did you do all the programming?

Vanderpool: I wrote the entire thing, yeah.

Smolin: And PMS-II as well?

Vanderpool: Yep. Well, not all of the stuff in PMS-II, there were a few contributors along the way, but I did the primary work on it, yep.

Smolin: How much did it cost you in dollars, to get that, either one of those, ready?

Vanderpool: Well, PMS-II was a, I think, early-on product in the microcomputer revolution if you will. I put $30,000 into the company to start with, and I think I spent $7,000 of it before we had positive revenue, and we had positive revenue from then on out. So the real cost of it was quite incidental to the overall value of it. The second product, BidFax, I would say I probably invested, not counting my time, but just actual out of house investment, probably another $20,000 into that product for the packaging, and you know, all of the legal protection, etc., to put it together. But again, doesn't count my time, I'm cheap [*laughter*].

Smolin: Were either of those products copy protected? Or how did you enforce deterrence against illegal copies?

Vanderpool: Interestingly enough, BidFax was attacked by that very thing. There was a rip-off copy of it that was created. It was called . . . some two-word acronym.

Smolin: Was it a knock-off, or was it actually your code that they stole and made copies of?

Vanderpool: The core of it was my code, and then they put a different face on it and just the display part of it was changed just to make it look different. And they advertised it—they even copied our advertising, etc. We had a lot of hassle with them. They . . . [used] negative marketing, which is something no one should ever do; their marketing message was, "Don't buy that; buy this because that's bad." It came around and put them out of business.

When I sold the BidFax product line to McGraw-Hill Publishing to add to their contractor estimating systems, that was the end of that product line, took me a year to get out of it but the copy protection was . . . well almost have to go back . . . BidFax had a different method of distributing. We didn't distribute it via diskettes. It was distributed online. I think it was probably the first online software distribution that had an online update distribution in the industry that I'm aware of. I don't remember anybody doing this before that. But because of the capabilities of the onboard fax cards that the customer already had to have installed, it was a very simple thing for me to send them a key for that computer, once they'd paid for it, and then it would come to my computers and say "Here's my key. Do I have a valid key to get software?" So it was essentially very much like the licensing agreements you see today. Then it would download the system, install it, and run it. Every month it would self-check itself, come back to my system, see if there were any updates, and download those automatically for the user. So they found great value to always having fresh, up-to-date, bug-free software automatically on their system.

Smolin: So the problem with somebody giving it to somebody else, could they do that?

Vanderpool: I never became aware of anyone copying it as a user and reusing seven copies in their company, because if they did, their key wouldn't work again. I had a secure key that was hidden in a system file on their computer which they didn't even know about. If it wasn't there, then the update service wouldn't work, and it would just turn itself back off. So that was a real crude, I guess you could say that's a very crude, early beginning copy protection system that proved to be pretty effective.

Smolin: Was PMS-II copy protected in any way on the user side?

Vanderpool: I don't recall ever doing any copy protection on that. The industry was so young. It was so new, you know microcomputers and selling software for microcomputers was so early in the game that I don't think people even thought about stealing things off of computers at the time.

Smolin: So how did you price PMS-II?

Vanderpool: I wish I could say, you know, that it was a deliberate act and that there were some value considerations going on there, but basically in the early conversations we had with RCA they said, "Well, we want it," and I said, "Well, I don't want to sell it," and they said, "Well, we'll give you $1,000 a copy for it," and I said, "Well, how about $2,000," and they said, "Well, how about $1,500." And we agreed at $1,300, and that's how it got priced, $1,295, and it stayed that price for its entire life.

Smolin: And the BidFax, what'd you sell that for?

Vanderpool: Well, BidFax was, you know, after Broderbund or whatever their name was, the people that came in and said sell everything for $99 type of thing. BidFax pricing was a different consideration. It was a logical

thought-through value analysis type of thing and considered all the distribution costs and all of the things we had learned in the prior years and came to a price on it of $595, which was an extremely good value for people at the time. They were spending more than that every time they wanted to do a bid just in clerical support to send out all of the notices. Postage was that much. So they just, it was never even a moment's thought process. It was never a decision point. It was, "OK. Here. Here's the money. And away you go."

Smolin: For PMS-II you had ongoing support revenue stream.

Vanderpool: Right.

Smolin: Did you have that for BidFax as well

Vanderpool: Yes, yes. That was the other side of it. I figured the way I came out with the original number was all of my distribution costs amortizing the hardware necessary to do it electronically and all of that sort of stuff came out to about what we were charging for the product plus the commission for the people that were selling it. So that was a break-even as far as I was concerned and recapturing the investment for what I perceived to be a much smaller market than what actually turned out to be. My revenue stream on it was online support and continuous support. For that we charged $400 a year, and I still have people trying to buy it today.

Smolin: So both of these applications were single user, not networked?

Vanderpool: Basically, yes.

Smolin: OK. So you sold it single-price for a license?

Vanderpool: BidFax, we did eventually have a multiproduct package for it. But it was simply multiple keys, you buy six copies of the single at a discounted price, and that was fine with everybody.

Smolin: Did the original PMS-II program give rise to variations on that product?

Vanderpool: No, but it had ancillary products added to it.

Smolin: You didn't have a PMS-III.

Vanderpool: No.

Smolin: But you had add-ons.

Vanderpool: We had add-ons that expanded its capabilities into manufacturing, into facilities management, in different areas where scheduling was still the core but resource management and material flow management were just as important. They had to feed into the schedule at a particular time. So I wrote modules that added those capabilities down the road. Plus, presentation graphics, plotters, things like that the bigger firms needed.

Smolin: Where did the ideas for these add-ons come from?

Vanderpool: The place that all good ideas for products should come from—from your customer base. These were needs that were identified, verbalized by customers, that you take and create a satisfaction of that need. When your customer base is already established, and they have a need that's not being satisfied, it's almost automatic. It's predefined.

Smolin: And the BidFax, same kind of evolution?

Vanderpool: Well, BidFax continued to grow, and the only add-on products that became available for it were packaged databases, geographical databases for subcontractors by specialties. Using the Construction Specification Institute's coding structure, we were able to take [data] from existing clients that wanted to get involved in a cooperative process, and these were usually the multiple office type construction companies, the Bechtels, the Fluors, those types of companies, where they had operations that were all over the world. They had these databases, and then essentially we made arrangements to use them.

Smolin: So you interfaced BidFax with third-party databases?

Vanderpool: Third-party databases. Of subs. Contractors. And that then became a new licensing revenue source without any production costs to it at all. Which is why McGraw-Hill wanted to buy this system from me because that's what they did, they licensed databases. So it was a natural to fit right along with BidFax.

Smolin: It sounds like you didn't face any big problems bringing PMS-II to market. The market kind of drew you in once you presented it?

Vanderpool: More than anything, I think, when the original article was published, it was incredibly surprising when people started calling and went to the trouble of finding the office phone number and everything to call and say they wanted to buy it. Marketing that kind of a product became much more complex as the industry grew, to the point where the advertising cost far outweighed any other cost.

Smolin: I remember at the time it was exciting because there were no channels of distribution. There were no precedents. There were no pricing models. There was no competition.

Vanderpool: It was a blank blackboard, fill it out yourself.

Smolin: So we had to make up our own rules.

Vanderpool: Yep.

Smolin: In case of the BidFax, did you have any problems getting that one launched?

Vanderpool: Well, BidFax had its own defined distribution channel because the guy that came to me with the original idea where there was no concept problem, he wanted to get involved in software distribution himself. So I sublicensed the system back to him for distribution, and that's where the $495 came out. That covered all of those costs, and I didn't

worry about it. He didn't take any part of the ongoing revenue; he just did the original sales. He did a very good job of using the existing distribution channels that were out there. Some of the independent stores were still in business then which are now all gone. But there were also outlets that specialized in different products for the construction industry that just added this to their bag and took it right in. It went national in two weeks. Two weeks to it spread[ing] all over the country.

Smolin: So you didn't have to do any research on the market or the competition?

Vanderpool: Well, there was no competition. It was a noncompetitive environment. There was nothing to even begin to compare to it.

Smolin: Did you trademark both of these products?

Vanderpool: Yes. All of it was trademarked.

Smolin: Did you write a manual for BidFax?

Vanderpool: The BidFax system came with a printed installation card which was two-sided, three-sided at one time, triple fold, that just showed them how to get online and how to get hooked up and get it going. The entire manual was built right into the software so it was distributed right along with the code. The entire user guide, training guide, and everything was online.

Smolin: You did have a manual, but it wasn't a hard-copy manual?

Vanderpool: Not a hard copy.

Smolin: All electronic. And how long did that take you to write?

Vanderpool: Probably about five times as long as it took to write the software [*laughter*]. That's a big chore that only the people that have done it have any idea how much resource it takes to create a reasonable user guide, because people are not very intelligent that use software and everything has to be triple supported; otherwise they don't get it.

Smolin: Was there a PMS-II manual?

Vanderpool: Yes. It was a huge PMS-II manual. It was like the *Webster's Dictionary* type of thing.

Smolin: As I recall, it originally went out in uppercased dot matrix, paper that was torn apart.

Vanderpool: [*laughter*] Yeah, that was the original.

Smolin: Continuous tractor feed paper, yeah, and three-hole punched.

Vanderpool: You got it, by yours truly. It finally evolved into about a 3-inch-thick three-ring binder with tabs and dividers, and that was just the PMS manual. Each of the add-ons had a 1-inch manual that went with them. The cost of distributing a hard-copy manual and the cost of maintaining the information that's in a hard-print manual is . . . unbelievable until you've experienced it.

Smolin: I guess those days are over.

Vanderpool: I hope.

Smolin: It's pretty much twentieth century technology.

Vanderpool: Yeah, oh yeah.

Smolin: How often did you release updates? Was it sporadic or scheduled?

Vanderpool: PMS-II had a pretty rigid schedule; I would say at least once every forty-five days there was an update to the system. We kind of worked on that kind of a schedule for it. BidFax, because it was online for distribution, and it was updated automatically through the user's system, that was an ongoing process. We kept that [schedule], pretty much, do a feature, release it, do a feature, release it. Bug fixes are obviously something that has to be factored into the equation early on.

PMS-II was very bug prone when we first released it. But after a couple of years of maturity out there, it was as close to a bug-free piece of software as I've ever seen anywhere. BidFax, being built on that base, pretty much hit the road pretty close to bug free. It did have some hardware glitches with the fax card in its early days, but all of that got resolved with the better drivers, etc. It was about as solid as a Sherman tank going down the road.

Smolin: So you didn't use a commercial installer for either one?

Vanderpool: There weren't any commercial installers at that point in time. And again, going a different route, looking at the experience that we had with PMS-II and sending things out by post, etc., versus electronic distribution, it just made no sense to do anything beside electronic. Again, it was painless for the end user. They just answered the question that was asked and put in the code and bang, bang, it came up for them. So that's kind of the model that we're on today I think.

Smolin: How long did it take your typical user to learn to use PMS-II?

Vanderpool: To learn PMS-II, sometimes it required them coming in and actually going to school. It was the subject itself. The use of the software was incidental to understanding the technique and how to apply it to the business model that they were in. So, that was the bigger training issue there, how to schedule, not necessarily how to use the software to schedule. It was pretty much straightforward if you understood what you were trying to accomplish. Then how the software went about doing it was pretty straightforward.

Smolin: Two, three days of training? Or a week?

Vanderpool: At max, yeah.

Smolin: And BidFax?

Vanderpool: BidFax was originally created to be done the way they were doing it manually, so the transfer of information, the approach they took,

the step they took doing things manually were just automated and simplified. And they took to that like a duck to water. It was like almost no training period whatsoever. Learning the terminology and setting up their database took a little bit more assistance, but to use their software once they got subcontractors set up in a correct database was incidental.

Smolin: So implementation time for PMS-II, I know it could take forever, but optimally?

Vanderpool: I would say it was probably a two-month minimum cycle for them to come up to speed in PMS-II.

Smolin: And how about BidFax?

Vanderpool: Maybe a day, two at the most.

Smolin: What kind of learning and implementation problems did you encounter with people and what did you do about them?

Vanderpool: I think one of the things that most software writers still make a mistake with is putting their system in their terms as opposed to their users' terms. I think a lot of the implementation problems that I've experienced and had to come back and solve were due to terminology that was familiar to me but not familiar to the end user—BidFax being an excellent example of that. There's a Construction Specification Institute and everybody knows that as the Contractor Code. And I, of course, called it the Construction Specification Institute's code, and nobody knew what the heck I was talking about. So it was the technical versus the real-life usage of terminology.

Smolin: Did you do custom modifications for either of these products?

Vanderpool: PMS-II got a lot of custom modifications coming in from real users across the spectrum from small to huge corporations. Many, many, many parts of its features came from the user base themselves. Custom modifications were actually paid for by the contractor that wanted them with the understanding that they would then be supported as part of the full release. So the exchange for them was this gets supported for me, and everybody else gets to use it too, but I get it first.

Smolin: How did you handle the problem of having custom modifications that weren't appropriate for all users? Did you support multiple versions, or did you make it invisible somehow? How did you finesse that problem?

Vanderpool: Because so much of the changes to the project management system were techniques and existing improvements in the technology of managing projects with critical path network methods, everybody could take advantage of the features. If they couldn't, if it wasn't a general appeal feature, I made it cost prohibitive to them, and they opted to not do it that way. Or I gave them an alternative that was not cost prohibitive but everybody could use. So a little bartering, a little meeting midway. I think if you have something that does something for everybody, then you have something that doesn't do anything for anybody.

Smolin: So all of the custom modifications made for PMS-II became generic or part of the standard package?

Vanderpool: Part of the standard offering.

Smolin: How about BidFax?

Vanderpool: BidFax didn't require very much of that at all because we had a much more extensive Alpha-Beta cycle on it. So it pretty much was ready to go and didn't get much change. The only changes that occurred to it in general were things having to do with the hardware support itself or fax cards as they matured as a piece of hardware that we were relying on better, faster, you know, smarter. Custom mods, other than changing terminology, not that much.

And it also had a much shorter lifespan. BidFax was a four-year product because by the time it had reached its maturity point, technology had already replaced it. The Internet became what we know it is today. It became apparent that that was the way everything was going to go. And in fact, the guy that had the original idea created a company called Buzzsaw, which was going to be huge and everybody got involved with it. But unfortunately they weren't paying attention, and all the money went away [*laughter*]. The dot-com bubble burst, and it got swept aside.

Smolin: No business model but plenty of hope.

Vanderpool: Lots of money coming in but no revenue.

Smolin: If somebody came to you and said, "You know, Al, I've got a great idea for a product. I've got half a program written. I'm going to do this," if you had only one piece of advice to give them to keep them from going off the rails, what would you tell them?

Vanderpool: That's an interesting question because I just recently had that question come up using the Nextel push-to-talk telephone systems, which are now available on a lot of carriers. They came to me with a product idea, and I said, "Define your market. Give me a market definition. Who is it and how are you going to reach them and why are they going to buy it from you?" They couldn't do that, and after four years of the owner and his friends pouring their money into it, mortgaging their dad's salvage company, etc., etc., etc., they went away. It was a great idea, a fantastic product idea, but they never defined their market. The market is more important than the product. Market definition.

Smolin: It was a solution in search of a problem?

Vanderpool: Absolutely. And if you don't know how you're going to get that solution out there, then you have no business whatsoever being in this business. Because this is an expensive business to be in these days. Great rewards, great risks. You can't do it in the old style that I've done it. In today's environment, market definition is the number one thing that I would say. If you don't have that, I'm not interested in the rest of your conversation because you've got nothing.

Smolin: I see by the old clock on the wall that our time is about up here. In fact, I just finished the questions.

Vanderpool: Great, great. I hope it was useful information.

Smolin: I hope it's on the little recorder thingies here.

Vanderpool: Might want to check.

Smolin: No, I don't want to know.