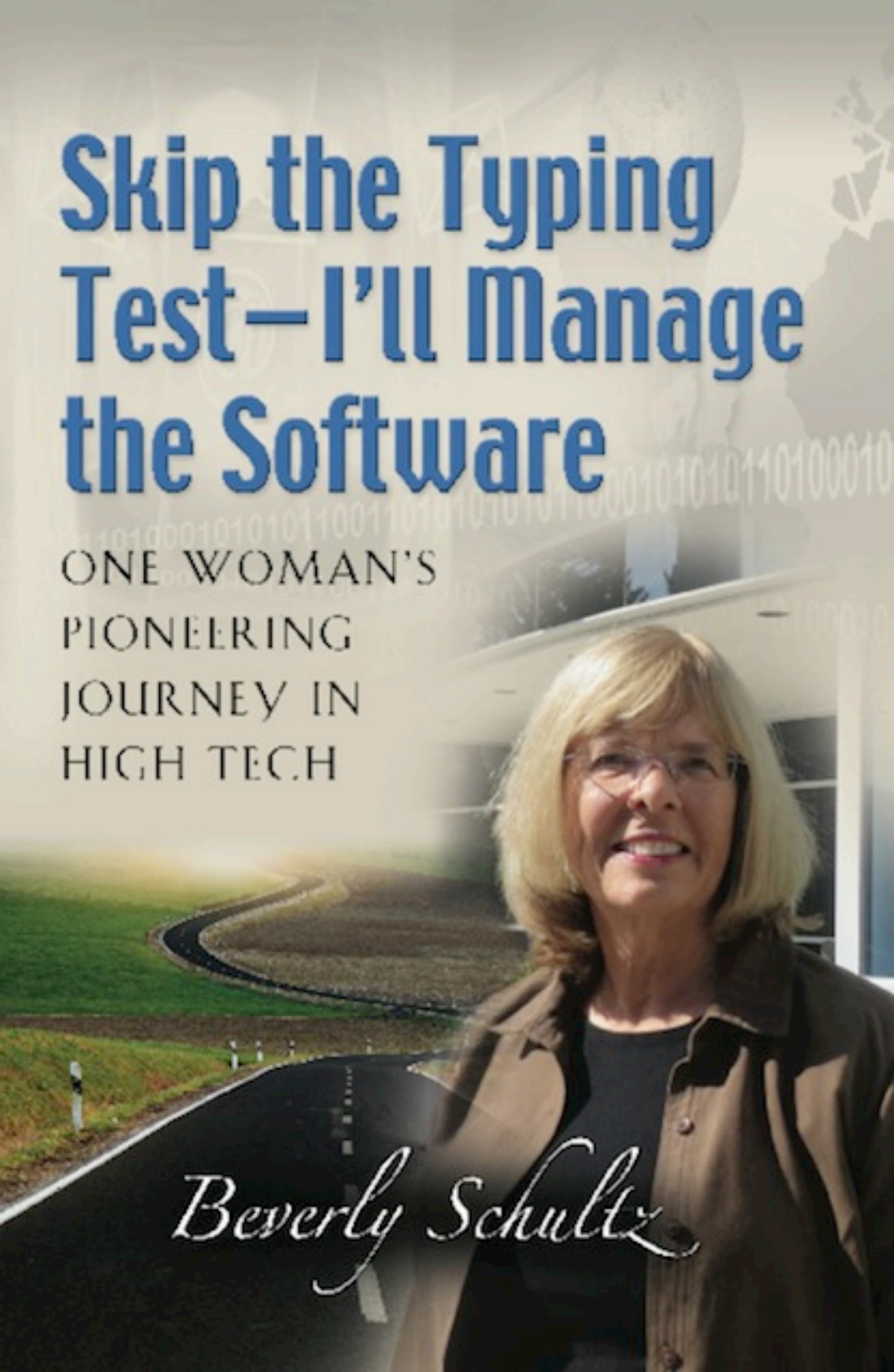
Skip the Typing Test—I'll Manage the Software

ONE WOMAN'S PIONEERING JOURNEY IN HIGH TECH



Beverly Schultz

Step into today's high-tech world with a pioneering female engineer in a male-dominated field! During the computer technology revolution that transformed the world's industries during the last quarter of the 20th century, Beverly Schultz's success story blends the challenges and fun of engineering at its best and worst, with specifics that engineers relate to and others treasure in this unique view of high-tech. Her engineering career tips resound across industries and occupations.

Skip the Typing Test, I'll Manage the Software

A Woman's Pioneering Journey in High Tech

Skip the Typing Test – I'll Manage the Software:

One Woman's Pioneering Journey in High Tech

Beverly Schultz

Hardcover ISBN: 978-1-62646-693-7
Paperback ISBN: 978-1-62646-694-4

Published by BookLocker.com, Inc., Bradenton, Florida.

Printed in the United States of America.

BookLocker.com, Inc.
2013

First Edition

the "boxes" on the SADT charts back in our design book represented software that didn't exist! It was fortunate that we now just had one set of modules to create. But it was a big one.

The security challenge was in allowing only certain people access to certain doors, and not to others. All employees had access to the main entry door, but some had access to labs, others to office areas only, others to multiple places. In addition, the software had to be written to allow certain employees to change that access *daily* as required.

We needed cards that the employees could use to show for access. SRL chose to use a new system: we bought chips that went into the intended walls and also bought cards with chips in them. The chips to be embedded in the walls just needed installing in the appropriate places, and the cards arrived in a box, so to make them work together was thought to be "just a matter of programming."

When an employee of Smith-Kline held his or her issued card at the door, the card's chip sent a signal to hardware we'd put into the wall. If the card showed the right access code, the door opened and the employee would be let in—without ever having to insert the card into a slot. (This was why they wanted it in downtown Philadelphia! No slots meant no vandalism and an always-working system.)

Of course it was more complicated than that. Because some employees were allowed into some doors and not others, and Smith-Kline could change access for any person at any time, the software had to be made to accommodate changes in a very timely fashion. Even the big "chopping gates" in the lower level parking garage were run with cards that would raise or lower the gates. This software had to be written not only to make the access cards work correctly, but also to work seamlessly with the SRL system and its little data-gathering computers, the Intel 8080 systems that were issuing actions, and the PDP-11s that did reports for security.

I didn't have anyone on my team available to design all of this software, so I did it myself. Actually, it was a joy to read the manuals and figure out how to design the security system using these cards. I hadn't done any design since I'd started working on this plant management system, and I found I loved designing. I was terrified, as I was putting my neck on the chopping block for my software expertise—skills I'd only been using sporadically to review the work of others. But I had a junior programmer who could code a lot of it for me, so I did all the design and discovered that what I gave him was easily coded, so I was even a good designer! Managers don't always get that wonderful feedback, or get to do fun parts of projects. On later projects, when software had to be designed for work I was doing, I wasn't averse to grabbing a bit of the design work for myself. It became my reward for juggling all the balls that a manager juggles.

Early "failures" in beta testing did not come from our software designs, but rather were the results of things we didn't think of. For example, I watched a woman standing in front of an entrance wall searching and searching in her *big* purse for her card. Meanwhile through the side of her purse, the card had already sent seven—or even a dozen—requests for the door to open. And access had been granted *every one of those times* even though she never actually found the card!

In another unforeseen situation, access cards given to some executives in our initial tests turned out to have a signal identical to one that a local box store used for their merchandise security clips. We discovered this embarrassing little glitch when one senior executive was stopped and searched at the box store because their security thought he was filching merchandise, when he was just carrying our card in his wallet!

I continued to have to defy, dispute and defend often when there were hardware problems with the systems we were building; the prevailing attitude at the time was that it was always the software that needed correcting. In 1980 software people were "weenies" and hardware people ran the show. They felt their hardware was right, so you had to

prove to them that their stuff wasn't working correctly—or just fix it in software, which they usually felt we should have done in the first place.

But the hardware-versus-software dynamic was beginning to shift. The industry was changing—for the benefit of small companies like SRL, and also for software engineers. In 1965, Gordon Moore, the co-founder of Intel Corporation, had made a statement that became Moore's Law. He predicted that computer processing performance would double every two years. The expanding complexity of the chips Intel was currently building attested to this. Customers delighted in the increased capabilities they were discovering, and used their processors for every purpose they could think up. But that meant customers depended more and more on software programmers to write code that made those processors do what was needed for business. And although software would not forever be the second cousin to hardware, it was my reality in 1980.

So I was in software and a female…why I was even left-handed! I had minority status up the wazoo. But I made myself fit. To do that, and so I could justify myself and prove my points, I always had to know exactly what I was doing. *That* made me a better engineer.

TIP #4: ENJOY THE GOOD STUFF.

Tell your body that this is the work you're built for, and revel in what you're doing. Help the body thrive, and absorb an attitude that says, "*This is good. I like what I'm doing."*

Step into today's high-tech world with a pioneering female engineer in a male-dominated field! During the computer technology revolution that transformed the world's industries during the last quarter of the 20th century, Beverly Schultz's success story blends the challenges and fun of engineering at its best and worst, with specifics that engineers relate to and others treasure in this unique view of high-tech. Her engineering career tips resound across industries and occupations.

Skip the Typing Test, I'll Manage the Software

A Woman's Pioneering Journey in High Tech