Choose the Right Platform for Your Mission Critical Computer and Software System

by Bonnie D. Huval

Do you need to build a major system to handle mission critical work for your business? Are you starting with a blank slate, so the first thing you have to ask your technical team is what platform and tools to use?

Selecting the right foundations and tools are among your most crucial decisions. Your starting point is probably not hardware. Think about what your system must do, and the resources you have for building it.

You must decide which trade-offs make the most sense for you. Strictly speaking, some of your choices are not directly about the demands of your project. Some involve the future availability of upgrade paths, spare parts and expertise. Plenty of expertise is available in the marketplace for the tools that are most popular right now; but if those tools are not suitable for your needs, lots of readily available labor will not compensate for shortcomings in the tools.

So let's begin by looking at the foundation of your system--the operating system.

What operating system is best suited? That will narrow the hardware and tools you can use. A few options:

* If rebooting once or twice a month to install patches is acceptable, you may want something ubiquitous and comparatively inexpensive like Windows.

* If processing speed and low overhead are important, and you have access to the right technical expertise, you may prefer UNIX.

* If the system must handle enterprise-class communications and data volume for business applications, you may want AS400 on a mainframe.

* If downtime is intolerable, you may need a special disaster-tolerant platform.

Sometimes Old Tools Are Just Right

While you consider this, try not to restrict your thinking to the newest products.

Why? Here are a couple of examples:

When the World Trade Center came down in 2001, CommerzBank's headquarters were located less than a hundred yards away. CommerzBank was running its most essential banking applications on a properly configured OpenVMS wide-area cluster with shadowed (mirrored) disks. In an OpenVMS cluster, multiple computers act as though they are a single computer. Processing work and storage are spread across and shared by the member computers. The bank could lose a data center without a noticeable blip, so it had no interruption in service. With appropriate hardware and

communication lines, computers in a cluster can be up to 500 miles apart--great for disaster redundancy.

OpenVMS is about 30 years old. It is highly secure and so robust that its computers commonly run for years between reboots. Many recent graduates have no idea what it is, cannot conceive of running five years without rebooting, and have never seen a "more modern" operating system capable of doing what the CommerzBank cluster did.

By contrast, due to Tropical Storm Allison in June 2001, flooding and a fire in Houston, Texas, took down both the primary and backup power sources for a major electronic transfer system. The system services a large percentage of automatic teller machine and point-of-sale transactions in 22 states. Those computers ran a more popular, lower cost operating system. Although disaster recovery plans were in place and a disaster data center was ready in the Dallas area, bringing the secondary data center online and resuming service took days.

University Education and Real-World Experience

Now you know why you should double check the opinions of your staff when you ask them to recommend the operating system, programming language, database engine, and other tools to be used in building your system. Double checking is especially important if your staff is young or has a tightly focused range of experience.

There are distinct limits to how much anyone can learn in a specific period of time, and computing is a vast field. No university can teach its computer science graduates everything about computing. For that matter, even the most experienced top notch computer expert can't know it all, either--but the experienced expert has been learning a lot longer. Independent experts have often seen a wider range of technologies and ways of applying them than people in typical jobs, because independents get to see how more companies and groups approach their projects and how the projects turn out.

Fair warning: Some distinctly technical talk is next, but stick with me. This real life story will help you see what you need to find out before making your foundation decisions, and why.

At one project, the group I worked with tied itself in knots trying to take care of a procedural application written in C. That is a relatively low level language, not far above machine code. It would have been clean, easy code to maintain in a higher level procedural language such as FORTRAN. On the computer and operating system we were using, on average each line of C generated only 3 lines of machine code. Each line of FORTRAN generated 8, which means FORTRAN took care of a lot of details but C required the human programmer to do nearly all the details. That meant C left many more chances for the programmer to make mistakes.

To make matters worse, a portion of the software had attempted to make the procedural language C behave like the object oriented language C++. It was convoluted and difficult for even the most senior programmers to update.

Eventually, the boss of the group told me that he was a programmer not long out of college when that software was conceived. He knew C and C++ from his degree studies, so that was what he recommended. He thought C++ was great, but the

company was only willing to provide C. He personally wrote the especially ugly portion that tried to mimic C++.

He admitted those were mistakes, especially the C++ imitation. It should have been done in a higher level procedural language, where mistakes would be less common and the code would be easier to understand. Now he had to live with the budget and resource strains that came from his decision. He made choices which were harder to maintain than other choices that were available, simply because he did not know about the other options and did not think to inquire about them.

For well designed, well built software, you can reasonably expect about 20% of its lifetime cost to go into creating it. About 80% of the cost is maintenance and enhancement. By making a less maintainable choice, he increased the lifetime cost of the system substantially.

He shook his head. "I didn't know any better. That was what I knew from college. All of us were right out of school. Nobody else knew any better, either."

A Little Outside Opinion Goes A Long Way

You can still use a young team to build your system. There are some good reasons to want youth on the team--energy, creativity, comfort with some of the latest software. Besides, if your team is young when they build the system, you have the potential to keep original expertise on hand for a long time.

Don't skip the double checking, though. It's worth your while to bring in a consultant to look at what you want to do and give you some advice. Even if you don't keep the consultant involved for the whole project, at least the consultant's experience can help you make sure you're going to use the right foundation.

When you're tempted to skip that step, just think about how much that shortcut can cost in the long run.

About the author: Bonnie D. Huval has been a consultant since 1992 helping companies make more money, especially with their automation and transaction systems. Her USA and UK business interests also include real estate, property management and a restaurant. Successful projects include cutting time to ship product from two days to two hours, and cutting downtime for product introduction by 40%. Go to http://www.seneschal.biz to get such consulting help for your firm. Go to http://www.makesureyougetpaid.com for her materials to help small businesses be more successful.

Article Source: http://EzineArticles.com/?expert=Bonnie_Huval