# Why I Like Vanilla

Hugh R. Alley - 12/9/2010 4:29:00 PM

Three years ago I joined a small manufacturing firm as Operations Manager. Two years earlier, the company had selected and installed a new super-flexible, highly customized enterprise resource planning (ERP) software package. Since I joined, I have reached the conclusion that I like vanilla. A lot.

**Background Flavor**
I first heard the term “vanilla” used in reference to computer software during a consulting assignment I had several years ago. I was helping an organization do a project review for a software installation that had gone sideways. The team described their recently adopted strategy as “vanilla, if at all possible.”

I had to ask what they were getting at. Their reply was that when they implemented software, they wanted to make as few changes to the software (out of the box) as possible. They did lots of configuration, to the extent that it was permitted by the underlying structure of the software, but very, very few changes to the software.

Why did they take this approach? Well, they described the history of their organization’s ERP system. An earlier version had been heavily customized. Then, with budget constraints (don’t we all have them?), they deferred upgrading the software as new versions came out, because of the cost of upgrading the customized portions and the extensive testing that would be necessary. The situation never got better, and pretty soon they were stranded: support for the version of the software they were using was about to be dropped by the manufacturer.

Now they were wrestling with the monumental project of transitioning to the newest version. Now they needed to keep it plain and simple. “Vanilla” was the new code word.

**Why You Should Keep It Simple**
There are, I think, three reasons to go for vanilla.

*1) Your business isn’t as different as you think.*

The operations of most businesses aren’t really that special. There are some specialized requirements in some industries; heat tracking in metal distribution, complete traceability in food and aviation, and so on. But even these are now handled natively by the many different specialized software offerings. I think there is some hubris in people’s belief that they’re special, different, unique. Often what that means is that the firm has grown up with unusual or obscure processes. In reality, those processes aren’t essential to what the organization does, but they have become identified as such. Choosing software because of existing practice is rarely a good decision.

The power of most ERP solutions is now high enough to allow you to do most of what you want to do. And if you can’t find one that will do, I recommend you think long and hard about whether what you want is really a necessary function. (As a side note, 10 to 15 years ago, when the power of ERP systems was much less, this was a much less convincing argument.)

*2) Documentation, training, and reporting are often afterthoughts.*

There are some common failings that occur when a company chooses super-flexible or highly customized software. Two are critical. The first is that documentation and training must be driven by the user, and this is rarely done well.

We have this situation where I work now. I am amazed by the staff’s resourcefulness in finding ways to get

their job done in spite of the apparent obstacles the system puts in their way. But when I dig into each situation, more often than not I find that they simply haven't been trained in what our ERP system can do for them.

The second is that reporting is often omitted. The developers work to get basic functionality in place. But between time pressures to get the system in and budget limitations, the reporting functions get left to a later phase. And like my former client's version updates, that phase often never comes.

At my current company, even five years after our software is in, I still do not have a simple report that shows me how much of a certain *stock-keeping unit* (SKU) we have sold in any given period. I can get the information by faking out the system, or by doing extensive manipulation of data extracts in off-line spreadsheets, but it isn't easy. It makes forecasting difficult, and that hurts our inventory management and our supply chain reliability. Big problem. By contrast, most out-of-the-box systems come with pretty good reporting, and as long as you don't change the code, the reports will give you what you need.

*3) Upgrades are hard with customized software.*

That's what my old client finally learned. If there are no changes to the underlying structure of the software, then upgrades are easy. The upgrades give immediate access to new capabilities, improve security, and reduce bugs. These are all good things. But with customized software, upgrades are more complicated, more expensive, and more risky.

**Vanilla Essence**
In my experience, the common argument given for choosing super-flexible or customized software is misleading. The argument says that with better flexibility, you will have the ability to do whatever you need. But it doesn't work out that way. The resources to make changes are limited, and the result is that while all things are possible, little of what is possible is actually accomplished. It doesn't seem to matter what size of organization you work in.

So the next time you go looking at ERP systems, think long and hard before straying from vanilla. Ask yourself whether those features you just can't live without are truly essential to carrying on the business of your organization. My guess is that often enough, you'll discover you have those requirements because you've always had those requirements, not because they're at the essence of your business.

Yup. I like vanilla. A lot.

**About the Author**
Hugh R. Alley, PEng, is the Operations Manager at Alco Ventures Inc, a manufacturer of aluminum railing systems and retractable screen doors. Hugh is responsible for the entire supply chain of the firm, including purchasing, logistics, manufacturing and distribution. Alley has been doing process improvement work for 20 years, and has been advising management about IT system requirements and implementation for 15 years. Educated at University of Waterloo (Systems Design Engineering), Cornell University (Resource Economics), and Vancouver School of Theology, Hugh has taught at both Simon Fraser University and University of British Columbia. He writes and presents frequently on lean manufacturing and lean processes, decision processes, and project management. Contact him at halley@alcoventures.com.