Should You Modify an Application Product?
The good the bad and the ugly

By Olin Thompson

It would be nice if off-the-shelf application products would satisfy every business need of every business. However, be it SCM, ERP, CRM, BI or any other category of application products, this is not the way it is. Many businesses find that some needs are unmet; they cannot live with a plain vanilla version of the product. If that is your situation, should you modify the package to meet those needs? Is the pain worth the gain?

What is a modification? A modification is any code written to get an application to perform differently from what the vendor intended. It may deal with writing code within the programs shipped by the vendor or it may involve code external to those programs. Sometimes, modifications include adding new fields to the database. Do "cosmetic modifications" count? These are changes to the screens, reports or workflows that do not impact the logic or database of the product. If these cosmetic changes mean that you have to do some work to redo or test the cosmetic changes when a new release arrives, the same issues apply … they count as modifications.

Let's start the discussion with a fundamental idea … modifications are to be avoided. Modifications mean that the vendor's support may be more difficult or even without value. Modifications mean that accepting new releases from the vendor will be more difficult, in terms of both time and cost. Modifications mean that the system is more error prone. Modifications mean that your long-term cost of ownership will be higher. Modifications should be avoided.

Looking at modifications in financial terms is looking at an issue of liability. This is not the type of liability that ends up on the company's balance sheet. However, it is a liability that is just as real. Time and money will have to be spent in the future for a modification decision made today. Defining how big the liability is going to be and thinking of it as if it were on the balance sheet helps management in these important decisions.

Many implementations start with an absolute policy of NO MODIFICATIONS! That's great if it can be adhered to. However, these are a small minority of all implementations. The absolute policy ends up with a few exceptions. The result of the absolute policy is still very positive, it minimizes the number and complexity of those modifications that do end up being made.

The Why of Modifications

So why would an enterprise consider modifying an application product? The answer is that sometimes it has to be. But why it has to be is not black and white. What is an absolute "must-have" for one enterprise would not even be

considered at another. What are the various types of needs that may result in a decision to modify and how can we prioritize them?

Some business processes or conventions are industry practice. These processes or conventions are the way business is conducted in that industry and not having the ability to follow them would mean not being able to conduct business. These business processes or conventions are mission critical. First, if this type of problem exists, maybe you did not select the right application product. If a vendor's "focus" on your type of industry is anything beyond a web site or brochure, they should provide standard features that deal with the industry practice. But if you find your selected application product does not meet your industry specific needs -- a modification is more than justified, it is imperative. This type of modification must be completed before the application can be implemented.

A business process may be unique to your business and therefore not provided by the application product. Why is this process unique to your business.

If the process provides you with a strategic advantage, then it may be worth the pain of modification to preserve the advantage. But this is not a low level decision but one that deals with competitive strategy. The value of the competitive advantage versus the long-term cost of the modification must be evaluated. Some of these processes are mission critical and many serve as essential advantages in the marketplace. Usually, this type of modification must be completed before the application can be implemented.

If a process is "just the way we do it" then the value can be questioned. Most application packages are combinations of best practices gained from working with many different enterprises. It is rare for the process unique to one company to be truly more valuable than those available in the standard application package. Often, these types of situations become the most difficult to decide. Some powerful user or group of users insists that the modification is required. A successful approach to dealing with this issue is not to say "no" but to say "later". A strategy that implements the standard product and differs all non-critical modifications for a period of few months after the go-live date will result in many of these "just the way we do it" modifications falling off the list. The users will start using the tools available in the standard package and after a few months of use, what was important to them will have changed. If after a few months of use these demands still exist, perhaps they are critical to the business.

The How of Modifications

If a decision is made to modify an application package, how it is undertaken can have a major impact on the long-term cost of ownership of the modified package.

Whenever possible, modifications should take place external to the vendor's code or database. The use of user exits, API's etc. or the implementation of pre or post processing programs isolates the impact of the modification. Utilizing database fields provided by the vendor (User defined fields) or tag-along files can

also be considered external modifications. When a new release arrives, the external modifications must still be tested, but the cost and risk is relatively low. In terms of modifications, if they must exist, this approach is a good one.

If the external approach proves to be unfeasible, then something must be done within the vendor's code or database. If the modification is limited to adding a block of code to the vendor's program, the effort in accepting a new release becomes on of adding the code to the new release and testing. Adding a block of code is not good, it is bad, but sometimes it is unavoidable.

The ugly of the modification world is actually changing the vendor's code or files. In general, application products are very complex. They must meet the needs of many different businesses and therefore have complex logic to determine how a program will function in a specific enterprise or implementation. What one program does, and how it does it, can impact the way other programs work. This complexity leads to high risk in any changes to the code. When a new release arrives, reapplying the changes may require a complete analysis of what the vendor has done to the program. It may mean a new way of changing the code to get the same result. It means very extensive testing, not just for the program in question but also for the entire system. Changing the vendor's code is ugly.

Summary – Have Your Eyes Open

Are modifications bad? The answer is yes. Are modifications justified, the answer is sometimes. The decision process should focus on the business benefits versus the long-term liability of making the modification. An absolute policy of NO MODIFICATIONS is a working starting point, but the needs of the business often create exceptions to that rule. Make exceptions with your eyes open. Open to the needs of the business and open to the long-term cost of ownership.

About the Author

Olin Thompson, a principal of Process ERP Partners, has over 25 years' experience as an executive in the software industry with the last 17 in process industry related ERP, SCP, and e-business related segments. Olin has been called "the Father of Process ERP." He is a frequent author and an award-winning speaker on topics of gaining value from ERP, SCP, e-commerce and the impact of technology on industry.

He can be reached at OT@OlinThompson.com