TGI® Software That Drives Business Performance.SM

15 Questions for Every ERP Software Supplier

Technology Group International
6800 West Central Avenue, Building I, Toledo, OH 43617

Toll Free: (800) 837-0028
Office: (419) 841-0295
Fax: (419) 327-9017
Email: info@tgiltd.com
www.tgiltd.com

Fifteen Questions for Every Software Supplier

Introduction

Picking an ERP package is a difficult and time consuming thing to do. It is not something that organizations engage in on a regular basis (at least hopefully). As a result, they tend to spend most of their time concentrating on their specific business requirements and how the software packages meet these requirements. Almost no time is spent attempting to ascertain how the software supplier actually runs their business.

In the actual software selection process, finding out what an organization will be like to do business with is easily as important as the overall functionality issue. When you select an ERP package, you are selecting both the package itself as well as the organization that supplies it to you. In your current business operations, it is likely that you have formed some sort of partnership and / or alliances with key suppliers. When you select an ERP supplier, far more so than many actual supplier relationships that you have in your actual business, it is imperative that you have a true 'partnership' relationship with that supplier. You will be inexorably linked with them through the entire life cycle of the ERP system within your organization.

This document is intended to help you, as a buyer of ERP software, to insure that you ask all of your potential ERP software providers some of the really tough questions that will help you define whether or not you want to do business with them as an organization. The questions here are **not** functional in nature. Rather, they are designed to help you understand, prior to purchasing a system from them, how a potential supplier will be to do business with.

In addition to a listing of the question, there is also some brief discussion associated with each question regarding its relevance in the selection process. As you go through your selection process, assuming you decide to require answers to these questions, it is important to insure that specific examples are provided by the supplier. This should include sample customers that you can talk to regarding the answers to the questions as well as, where relevant, specific sample copies of documents used by the supplier in the process.

Whatever you do, **do not let the suppliers off the hook**. These questions are tough. Most suppliers will not want to provide good solid answers to them because they do not want to have to deal with the ramifications of the answers they would have to provide. However, they represent the best means available to you to ascertain ahead of time what doing business with a given ERP supplier over the long haul will be like.

Fifteen Questions for Every Software Supplier

The Questions

1. Will you provide full source code in the price of the software?
2. Will you provide a risk-free trial period on the software during which we can return it for a full refund?
3. Is the software written in a commercially available development language which is still being enhanced and supported by the supplier?
4. What is the cost for the first and subsequent years' maintenance with your software?
5. When a call is placed to your support organization, who is the first person we talk to and what is their background and experience with the software?
6. What is the average amount of time that lapses between a report of a non-mission critical bug and the 'fix' becoming available in the software?
7. How does your organization compare to other organizations in your industry relative to revenue per employee?
8. Does the system come with a fully integrated Warehouse Management System (WMS) as part of the software cost?
9. Does your software include Customer Relationship Management (CRM) functionality as a separate module?
10. Are there any restrictions as to which database, hardware platform, network, or operating system environment the software can run on?
11. What is the typical ratio you have historically seen amongst your customers of implementation cost to software cost?
12. What is the typical implementation time frame for a company of our size?
13. What is the methodology you use to track project progress vs. plan from both a work completed and a financial standpoint?
14. Will you provide a guaranteed maximum cost for any work you do for our organization?
15. What level of effort is required to install future upgrades of the software?

Will you provide full source code in the price of the software?

A surprisingly large number of organizations wonder why this is a major issue when purchasing an ERP system. Fundamentally, it is all about freedom. Once you have a copy of the application source code, you are no longer tied to the vendor who initially provided you with the software. What this means is that, should you choose to do so, you can use other organizations to provide you with on-going application programming, or you can do it yourself.

While the bulk of the organizations which purchase commercial ERP systems do not want the issues associated with doing their own modifications, having the flexibility to do so is extremely important. The percentage of companies that, over time, grow tired of the continual cost increases and poor responsiveness from their original software supplier is well over 90%. Having the flexibility to either go elsewhere for the services, or to do them yourself, is a simple and easy way to insure that the software supplier remains continually interested in providing you with the highest quality service at a market competitive price.

Many ERP vendors tell you that they will 'escrow' the software source code for you at some third party location so that, in the event that they terminate business activities, you will have a copy of the source code for your use. This **is not** the same thing. Having software in escrow means that you still do not have access to the code and that you are forever tied to the supplier for any work you might want performed. Having software in escrow is the same as not having a copy of the source code at all. Moreover, escrowed source code is almost never kept in sync with the supplier's most current release of the software.

The second most popular non-answer to this question is the organization which will provide you with a copy of source code, but for an additional cost. What this really means is that the supplier really does not want to give you the source code, but will if you give them a sufficient amount of money so that they get most of what they would have gotten in service revenue anyway.

Finally, you need to be fully aware of the ramifications (as specified by your software supplier) associated with an organization other than the original software supplier doing any modifications to the software. Typically, when you, or an organization you designate, modify a program, you invalidate the support contract **for that program**. As a result, you assume complete responsibility for the functioning of the program itself (including those portions of the program that are part of the originally delivered application code), as well as any down stream data impact that the program may have.

Will you provide a risk-free trial period on the software during which we can return it for a full refund?

The first thing you have to realize when you are buying an ERP system is that the person on the other side of the table is trying very hard to **sell** you an ERP system. ERP sales personnel can basically be broken down into three categories:

- Blunt and direct
- Embellishers
- Flat-out liars

In a very large sense, buying an ERP system is not unlike buying an automobile. The sales representative at the dealership can tell you everything about the car, but you really do not know how much power the car has until you test drive it to verify that it feels like it has a V8 and not a small 4 cylinder engine.

What all this means to you is that there is a high probability that there will be a substantive difference between the software functionality you thought you bought based upon the demonstrations and RFP responses and the functionality of the software that actually gets installed. To an extent, this happens with every ERP installation. Despite the best efforts of all parties, the expectations on the part of the buyer are always higher than the reality of what they actually received. In the vast majority of cases, the simple fact is that you have no alternative other than to continue with the implementation and make the system you purchased do the things you need it to do.

In order to get around this potential problem the buyer needs to have a period of time, following the installation of the software at your site, during which you verify that the functionality of the software which you purchased is what you expected it to be based upon the demonstrations and the RFP responses. This is commonly called an **acceptance period**. During this period of time, if significant functionality ‘gaps’ are identified in the software between what you have and what was committed that you would have during the sales process, you should be able to return the software to the supplier for a full refund. In our automobile analogy, this is a lot like the test drive when we get to verify that what the sales representative told us is actually what we have received.

Note that the purpose of this period is not to verify that you received everything you wanted, but that you received what the vendor said you were going to get during the sales process.

Is the software written in a commercially available development language which is still being enhanced and supported by the supplier?

When you really take a close look, you would be surprised to learn the number of ERP packages on the market today which either: (1) are written in a proprietary language (i.e., not a commercially available development language which is available for purchase separate from the ERP application); or (2) are written in a commercially available language which is no longer being enhanced by the supplier. Some of the biggest names in the ERP software industry have software written in a proprietary language. A substantial number of other suppliers use a development language which is either no longer in existence or no longer being enhanced by the language supplier. For the potential ERP system purchaser, these are extremely important issues.

In the case of an application written using a proprietary development environment, there are three primary risk factors. First, a proprietary development language means the availability of a relatively limited number of individuals experienced in the language. This, in turn, means the cost associated with using one of the few individuals around who is experienced in the proprietary language will typically be 2-5 times greater than the cost of an individual experienced in a commercially available language. Second, since the development language is tied directly to the software package itself, there is a high probability that the language will not be adequately robust to handle other peripheral applications which you might wish to develop. This simply means that you will have to use some other development platform for these applications. A mixed development environment will result in duplication of function, and thus increased cost, through the information technology function. Finally, this environment results in organizations having to pay exorbitant salaries to individuals who have some degree of expertise in the environment. It is not uncommon for even entry level programmers with minimal experience in these proprietary environments to command six figure starting salaries.

Over the past decade, a number of 'development platforms of the future' have come and gone. Some have stood the test of time, but the majority have not. Buying an ERP system written in a language which either has gone completely out of existence or is no longer being invested in by its developer is an extremely risky proposition. You could easily find yourself in a situation in which the ERP application you install today is already a technological dinosaur. Before buying an ERP system, make the ERP vendor supply you with the name of the company that created the development environment the application is written in, along with a phone number you can call to find out the current status of the environment and its future development direction.

What is the cost for the first and subsequent years' maintenance with your software?

This is a biggie! Long term maintenance revenue is one of the most coveted things in any application software environment. It is particularly true, however, of the ERP market. The reason for this is quite simple. If you have a relatively high quality product (i.e., one in which the number of systemic 'bugs' in the core application is minimal), then the revenue generated by the maintenance contract will outpace the cost of providing that maintenance by potentially an order of magnitude. There are software companies that generate 100% of the money they spend on R & D as a result of the profit generated from maintenance contracts.

This is no more true than in the first year after installation of the application software. For a significant portion (and, with most ERP suppliers, all) of the first year after software installation, your organization will be in the software 'implementation phase'. In other words, you **will not** be actively using the software to run your business for most or all of this first year following software installation. If you pay a maintenance fee during this first year, therefore, essentially no real maintenance type activities are being performed. You are, therefore, effectively paying for something you are not getting.

Most ERP suppliers on the market today charge maintenance fees ranging between 10% and 25% of the license fee of the software. In some cases, this percentage is based upon the **list** price of the software while in others it is based upon what you actually paid for the application. In either case, paying a maintenance fee during the first year is really equivalent to paying between 10 and 25 percent more for the software than you think you did.

The solution to this problem is simply to not pay any maintenance fee during the first year following software installation. While this may seem like a radical approach, you must remember that the software supplier knows they are not going to give you much in the way of maintenance services while you are actually in an implementation mode.

In subsequent years, it is common for ERP software suppliers to have a COLA (Cost Of Living Adjustment) clause relative to maintenance cost escalation built into the software purchase contract. Your best bet is to either negotiate a five year fixed annual rate for maintenance, or to negotiate payment for four years of maintenance up front and get five years of maintenance for the four year price. You should never put yourself in a situation in which long term maintenance costs for the software are not extremely well defined as part of the actual software licensing agreement.

When a call is placed to your support organization, who is the first person we talk to and what is their background and experience with the software?

One of the most important aspects associated with any software supplier, but particularly with a supplier of ERP software, is what type of support do they provide to you when you have a problem. In most cases, people do not call into a support line unless they are currently having a problem that they do not know how to deal with. Usually, they need some form of help immediately.

The vast majority of ERP software suppliers utilize some form of a help desk concept. In this mode of support, your initial call is answered by someone in the support organization who typically:

- Logs the call
- Assigns a case number
- Records pertinent information regarding your question/problem

The big question is what happens next. The typical scenario would be that the person you initially talked to is not, in fact, someone capable of providing you with support. Rather, they are a recorder of information. As a result, your call will be forwarded on to a support engineer to handle. The problem is that the support engineer is most likely busy. You will be either asked to hold for the next available engineer or will be told that an engineer will return your call as soon as he / she is available. When you do get in touch with the engineer, there is a high probability that they have been in their job for less than six months. As a result, if their existing artificial intelligence 'debugging' system is not equipped to handle your question, this level of support will be of nominal value to you and your problem will need to be escalated to the 'supervisory' level. This, in turn, typically requires additional wait time, further delaying the resolution of your problem.

In **every** software organization, support is regarded as purgatory (not quite hell, but pretty close). As a result, the most senior members of the organization, who are the ones that have the greatest depth of knowledge about the software, seldom, if ever, spend any time on support. Instead, the support desk is manned by the most junior members of the organization who have the least amount of knowledge about the software.

The software suppliers who provide the best support are those who:

a) Have a support person answer the initial phone call
b) Have seasoned development personnel manning the support desk

If your ERP software supplier does not do this, then you are in for a lengthy and frustrating support experience.

What is the average amount of time that lapses between a report of a non-mission critical bug and the 'fix' becoming available in the software?

When a company buys an ERP system, they naturally assume that the response of the software vendor to the report of a 'bug' in the software will be to fix the problem immediately and make the 'fix' available to the user community. For 'mission critical' problems (i.e., those which directly impact an organization's ability to conduct business), this will, in fact, be the case for essentially every ERP provider on the market (if they did not do this they would find themselves rapidly out of business). As a result, the aggressive resolution of a mission critical problem is an expected process on the part of both the software purchaser and the software provider.

Non-mission critical 'bugs', on the other hand, are quite a different story. A non-mission critical problem is one for which there is either a procedural or system 'work around' that, while potentially annoying, does not impact an organization's ability to conduct business. An example of a non-mission critical bug might be the need to key the payment terms of an order manually rather than have it default in based upon the customer entering the order.

It is a common practice in the software industry to say that non-mission critical 'bugs' will '**be fixed in the next release of the software**'. If you have never heard this from a software supplier, then you are one of the fortunate few.

Unfortunately, waiting until the next release (unless it is going to be tomorrow) is a fairly ineffectual option if the work around you end up having to use allows you to continue to transact business, but at a significantly diminished rate. The simple fact is that a problem is a problem. While it is true that some have a greater impact than others, this does not diminish the fact that a problem exists. When looking for an ERP supplier, it is important to determine exactly how the organization handles these non-mission critical issues.

A quality organization will treat all problems reported as serious. As such, even the non-mission critical problems receive immediate and direct attention by the support organization. Most of these problems should be resolved by the support team of the software supplier within a few days.

When evaluating ERP suppliers, make sure you talk to their existing customers and find out if they have ever heard that a problem will be 'fixed in the next release'. Ask their customers how long they have to wait, on average, between the report of a non-mission critical problem and the successful resolution of that problem fully implemented within their environment.

How does your organization compare to other organizations in your industry relative to revenue per employee?

When you are considering purchasing an ERP system, you want to make sure that there is a very high probability that the organization you are dealing with will be there for the long term. When going through the evaluation process, almost every organization looks at revenue as being the guidepost for organizational financial health and probable longevity. Unfortunately, revenue is almost never a good indicator of financial health of an organization within the software industry. In the past five years, there have been numerous organizations who supply ERP systems with annual revenues in excess of $250 million that have effectively ceased operations. Revenue is almost **never** a good indicator of the health of a software supplier.

There are other organizations which look at 'profitability' as a measure of financial viability. While profit is certainly an extremely important element of financial viability, it ignores two key measures of organizational performance. First, it does not appropriately scale the performance of the organization. For example, if an organization makes a profit of $1,000,000 in a year and has 1,000 employees and a second organization makes the same $1,000,000 annual profit but has 100 employees, which organization is in better financial shape? In other words, profit alone is deceptive in that it does not reflect the investment (in people, etc.) required to generate that profit. Secondly, the amount of profit an organization makes can, and in the software business absolutely should, be directly impacted by the amount of money the organization is re-investing in further development of the application.

Within the software business, the only real indicator of financial health and viability is to measure **revenue per employee**. There are many reasons why this measure alone is the most significant single indicator of whether an organization has solid long term survival prospects. First, it scales both the revenue and profitability variables to be reflective of the size of the organization. Second, it recognizes the fact that, within the software industry, the level of compensation to developers, project leaders, etc. is fairly consistent throughout the industry. In other words, the cost per employee does not vary greatly between one software company and another. As such, if the cost per employee does not vary much from organization to organization, then revenue per employee becomes a strong measure of both latent profit (profit prior to re-investment in the business) and real revenue (revenue scaled based upon the size of the organization). In the software business, an organization doing less than $150,000 in annual revenue per employee should be regarded as suspect.

Does the system come with a fully integrated Warehouse Management System (WMS) as part of the software cost?

When looking for an ERP system, the vast majority of organizations do not differentiate between the concepts of inventory control and warehouse management. Unfortunately, most software buyers assume that if an ERP package can make use of RF (Radio Frequency) and Bar Code equipment within their system, then they have an integrated warehouse management system. As a result, when they ultimately realize that simply using these technologies does not constitute a warehouse management system, the bulk of these companies will end up spending a substantial amount of money to subsequently acquire and integrate a warehouse management system into their chosen ERP package.

A quality inventory control system will provide the following capabilities (using RF and Bar Code technologies):

- Cycle and physical counting
- Inventory moves
- Picking and shipping
- Receiving
- Inventory locator
- Primary pick and putaway locations as well as restocking rules

In addition to these functions, a warehouse management system adds (at a minimum) the following capabilities:

- Establishing ‘pick’ rules (minimum quantity, order value, etc.)
- Different pick ‘schemes’ (e.g., wave picking, area picking, etc.)
- Automated direction of the activities performed by individuals (pick, putaway, moves, etc.)
- Productivity measures for individuals in the warehouse
- License plating (putting multiple of the same or different items on a pallet / skid and identifying the entire pallet using a single ID)
- Infinite nesting of license plates (license plates containing other license plates and / or license plate and individual product combinations)

If an ERP system that you are considering does not have a true warehouse management system integrated as part of the package, and included in the price you have been quoted, then you are not getting the full functionality you will ultimately need from the ERP system. While you may not perceive the need initially, a significant portion of the ROI from an ERP system will ultimately come from management of the warehouse.

Does your software include Customer Relationship Management (CRM) functionality as a separate module?

Customer Relationship Management (CRM) is one of the latest industry buzz words. Everyone talks about CRM and its overall importance in the entire supply-chain process. Knowing and understanding your customer, their organization, what they buy, when they typically buy it and in what quantities is one of the clear bench marks for future success of your organization.

While CRM encompasses a wide range of features and capabilities, a system which includes CRM functionality contains these (minimum) basic features:

- A complete contact management system, including an infinite number of contacts, along with the role they play in each organization you interact with, as well as an infinite number of notes for each contact
- Ability for sales reps and / or customers to directly access information and / or process transactions regarding customer orders, invoices, accounts receivable, etc. remotely (generally via the internet)
- Ability to identify customer buying patterns (what products, purchase frequency, etc.) and proactively prompt customer service and / or sales when these patterns are not being seen
- Ability to both manage customer inventory consigned to one of your facilities as well as perform vendor managed inventory (VMI) of your products at one of their facilities, including an 'invoice when consumed' capability
- Ability to track your penetration with your customers and identify opportunities for incremental capture of share with that customer

Beyond a simple list of features and functions, however, the nature of the CRM offered by the software is extremely important. If the CRM functionality in the software you are considering exists as a separate module, especially if it is provided by a 3rd party, then this is a clear indication that CRM concepts **were not** designed into the application, but were included as an after thought. The importance of this cannot be over emphasized. For CRM to work properly, it must be a concept that was imbedded within the application from its conceptual design stages. Thus, having CRM as a separate module is, in fact, a **bad thing** since it means that CRM concepts were not considered in the original design of the application.

Are there any restrictions as to which database, hardware platform, network, or operating system environment the software can run on?

Perhaps the single item that most significantly limits the viability of an ERP system is the restriction on what operating environments the system is capable of utilizing. The table below shows a reasonably complete listing of the different client, server, and network operating systems that are available on the market today. For simplicity, we have eliminated the concept of a single computing device providing application support through traditional 'green screen' character based system (this is 30 year old technology which is [generally] no longer being invested in by application software suppliers).

Database Server Operating Systems
- Windows (2000/2003/NT)
- Unix (AIX, Unix, Solaris, etc)
- Linux
- OS/400 (AS/400 standard Operating System)

Client Operating Systems
- Windows (95/98/ME/2000/XP)
- Linux

Network Operating Systems
- Windows (NT/2000/2003)
- Novell

Client-Server Connectivity Methodologies
- Thick Client (PC running application on each person's desk)
- Thin Client (Desktop 'appliance' connects as 'virtual' PC) using
  - Terminal Services or
  - Citrix

When selecting an ERP system, it is important to choose a system which **maximizes** the number of possible operating configuration options which are available to you. This insures that your future flexibility is similarly maximized through the entire life cycle of the ERP system. Any software package which you would consider should be capable of operating in **any** combination of environments. For example, if you have 100 users, you should be able to have some using Windows 95, some Windows 98, some Windows XP, some using 'dumb' terminals, and still others using thin clients with a combination of Citrix and Terminal Services. Without this capability, your future ability to utilize technology to its fullest potential will be severely compromised.

What is the typical ratio you have historically seen amongst your customers of implementation cost to software cost?

Within the ERP software industry, there is an adage that goes something like ‘Whatever we give away on the software sale, we will more than make up for on implementation services’. What you really have to recognize is that no matter how good a job you think you do in communicating your requirements to the software supplier, you are going to either miss something entirely or not communicate the real depth of the requirement. The net result of this miscommunication during the sales process is that, despite the best efforts of both the buyer and the seller, there is always something unexpected that arises during implementation which results in (usually) incremental costs being incurred on the part of the buyer. This is commonly referred to as ‘scope creep’. This can range anywhere from minimal to outrageous. In one of the most extreme cases of which we are aware, the company spent around $6 million for application software, with a budget of $40 million to implement the software across the entire enterprise. They ended up spending over $350 million to implement, and did not implement it throughout the entire organization. The key is that a quality ERP software supplier knows and understands the concept of ‘scope creep’, and can fairly accurately include this within their project cost estimates. As an ERP software buyer, you need to be aware that there are many ERP software vendors who understand scope creep and will use it to their advantage by not incorporating this factor into their cost estimates.

The actual ratio of implementation cost to software costs is heavily dependent upon the size and sophistication of the implementing organization. With smaller, less sophisticated organizations, this ratio is typically quite high (implementation costs of 5 to 10 times the cost of the software). This results largely from the fact that the cost of certain implementation tasks (e.g., training, configuration, etc.) do not scale linearly with the number of users. Exactly the same thing occurs in very large companies. Not only does implementation cost not scale linearly with size, but these organizations have the added complexity of size being a hindrance to decision making and thus project progress. Implementation costs for large organizations commonly range between 5 and 10 times the cost of the application software.

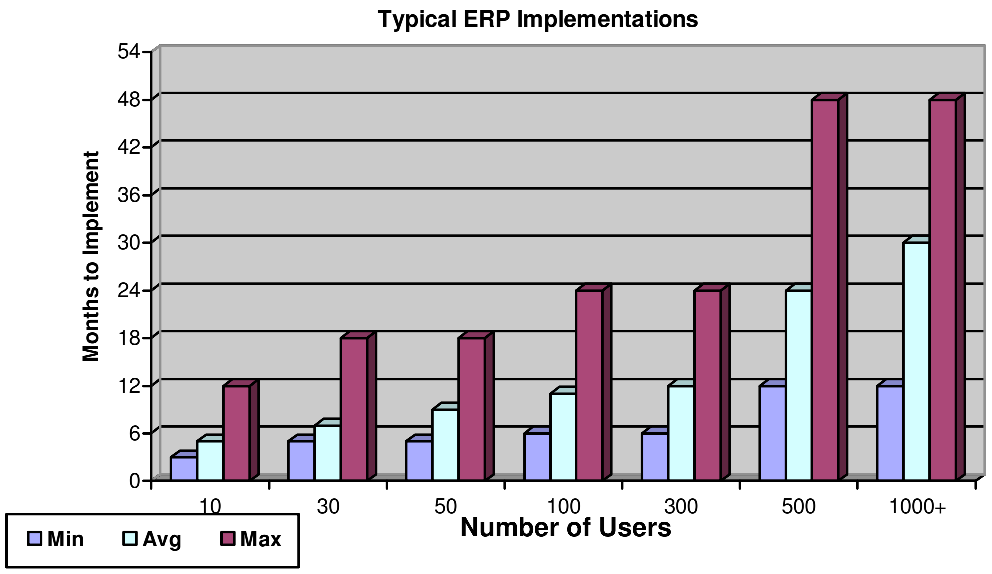
For organizations in the very broad mid range (30 – 150 users), implementation costs typically range between 2 and 5 times the cost of the application software. A few organizations, whose processes are extremely well defined, can do implementations whose costs range between 1 and 2 times the cost of the application software.

What is the typical implementation time frame for a company of our size?

Just like implementation costs, implementation time frames vary widely based upon size and sophistication of the organization. The most important element in how long it takes to implement an ERP system is the level of commitment on the part of the senior management of the organization which purchased the ERP system.

While there are a wealth of reasons why ERP implementations take substantially longer than originally anticipated (‘scope creep’, data conversion issues, etc.), the single largest issue in implementation is the staff of the implementing organization spending the necessary time to actually make the implementation happen. This is difficult because in most mid-sized companies, implementation activities are typically performed by people currently working for the organization that purchased the ERP software (as opposed to larger organizations which typically use external sources to augment their staff during an implementation). In most of these mid-sized companies, these individuals are already working a solid week without the added requirements placed on their time by an ERP implementation.

An ERP software vendor should, based upon experience, be able to tell you how much time your organization will need to spend during the implementation process and how long the entire process should take. For purposes of comparison, the chart below shows some typical ERP implementation time frames taken by organizations of varying size.



What is the methodology you use to track project progress vs. plan from both a work completed and a financial standpoint?

Broadly speaking, almost every organization that implements an ERP system uses some form of project tracking system. In a very large number of cases, this will be Microsoft Project, or a very similar package. The question, however, is really much deeper than this. What is really important is the degree to which you are kept informed by the software supplier of where you stand relative to the original (and / or revised) project cost estimates.

In the vast majority of cases, the way a buyer keeps track of the costs incurred on an ERP project is to flag payments to various suppliers as being associated with the project. They then produce a report which shows what has actually been paid and / or invoiced to date on the project. In some cases, the buyer will actually keep track of what is spent on various categories within the project. What is of relevance here is that it is typically the buyer who is keeping track of the project against budget, not the software supplier.

When you get into an ERP implementation, you need to be involved with a software supplier who is as interested and concerned about cost containment as you are. One of the ways to determine the level of the supplier's concern is to define whether they have an active program to track and monitor not only percent of work completed but also project costs versus budget.

A concerned supplier should be capable of providing you with:

- A detailed project plan of tasks with responsibilities and dates
- A high level estimate of costs to be incurred by category by month which is integrated with the project plan
  - Installation
  - Project Management
  - Data Conversion
  - Training
  - Configuration
  - Modifications
- A monthly report tracking, by category, projected versus actual costs for the current month as well as project to date.

During the software selection and evaluation process, most organizations ask the software supplier to provide a sample project plan. While this is important, **it is insufficient to address the real requirement**, which is for an integrated project plan including the above items. You should ask any potential supplier to provide you with copies of these items used in a recent project.

Will you provide a guaranteed maximum cost for any work you do for our organization?

As discussed earlier, the single largest source of cost and time overruns during an ERP implementation is ‘scope creep’. For purposes of the discussion at hand, it really does not matter why it occurs; the fact of the matter is that it does on essentially every ERP implementation project to some extent. As a buyer of ERP software, you want to insure that the potential software supplier has a well defined process in place to help you to identify, up front before any work is begun, what the financial magnitude and scope of the ‘scope creep’ will be.

During any ERP implementation, there is a process which is designed to define the ‘fit’ between the business process used by the organization and the processes as defined in the base ERP system. Discrepancies between the current business processes in use by the organization and those internal to the ERP application are also identified. This process is commonly called ‘gap analysis’. The result of this process is a series of ‘gaps’ which exist between the way the business handles certain processes and the handling of these same processes within the ERP application.

The ‘bridging’ of these gaps is accomplished by one of two means. First, the business processes in use by the organization can be altered to be consistent with the process internal to the ERP application. This is highly desirable from both a cost and process standpoint. One of the major reasons organizations implement new ERP systems is to insure that they are using ‘best practice’ processes within their organization. In the event that the gaps between the functionality of the system and the processes used by the business are simply too expansive, then the second alternative is to modify the system. Care must be taken to insure that tight control is maintained on the number and scope of application modifications.

When making modifications, the software supplier should provide you with a written specification which, in detail, defines exactly what the modification entails, including process logic, screen shots, field logic, etc. In addition, the specification should provide a cost to perform the modification as well as install and test it. This cost **should not** represent an estimate. Rather, it should represent the **maximum** amount you should have to pay for the defined modification, assuming no subsequent alteration in the requirement that precipitated the modification. If the modification takes less time than anticipated, you should pay less than the quoted cost. If it takes longer than anticipated, you should only have to pay the quoted maximum. Your software supplier should be required to provide you with sample modification specifications, including the cost estimate as part of the selection process.

What level of effort is required to install future upgrades of the software?

The ability to easily upgrade an ERP system to take advantage of new functionality and / or technology included in future release of the software is one of the major reasons many companies decide to implement a new ERP system. In addition, each year you will pay 15 – 25% of what you originally paid for the software to maintain an active maintenance agreement with the supplier. Most companies would like to believe that this money is being spent to actually derive value from subsequent releases of the software rather than simply provide the software supplier with a constant revenue stream.

The issue of applying software upgrades provided by subsequent software releases is of importance only when the base software package has been modified to fit the requirements of your business operating environment. Since essentially every ERP implementation involves customization in one form or another, upgrade ability is a serious issue. As a result, the real question here is "**In order to take advantage of a software upgrade, do we have to re-incorporate all of our modifications into the new release?"** Stated another way, do you have to pay a substantial fraction of what you paid originally to perform the modification to have the modification incorporated into the new release?

In order to answer this question, you really need to understand, at a conceptual level, how the software developer has constructed the application. There are several things to look for that will tell you that the application has been designed so as to facilitate relatively easy re-integration of modifications into new releases. First, many application developers create application 'exits'. These are points in the code where a customization can be performed and automatically re-incorporated into the application. The second option is to look for an application framework which is 'project' oriented. In this system, a 'project' exists which defines the actual location on the hard disk of the computer of each part of the application. When you customize the application, you simply point the project to a different location on the hard disk where it can find a customized version of a specific portion of the application. When new releases of the application become available, the project still points to the specified location where the customizations can be found so they are automatically incorporated into the new release. Finally, modifications can be made directly to the underlying application code. If this is the methodology used by the software provider, then your ability to cost effectively incorporate your customizations into new software releases will be extremely limited.

About TGI

TGI is an industry-leading enterprise software solution provider to small and mid-market manufacturers and distributors. TGI's exclusive focus is on the development, implementation, and support of Enterprise 21, the company's fully-integrated business management software solution. TGI is a privately-held organization with one of the highest revenue per employee ratios in the ERP software industry.

Find out more by visiting our website at www.tgiltd.com or by calling us at (800) 837-0028 or (419) 841-0295.