enVista®
enabling enterprise excellence®

Risk Mitigation for ERP Implementations

Nigel Cox, CFPIM, CIRM, MBSP
enVista Enterprise Solutions

Contents

enVista®
enabling enterprise excellence®

What is ERP?

ERP stands for Enterprise Resource Planning, but really it's interpreted as the system or systems (people, processes and computer systems) used by all members of the back office to run the business. This typically includes:

1. **Financials –** General Ledger, Accounts Payable, Accounts Receivable.
2. **Procurement –** Buying, including such areas as Vendor Onboarding, Incoming Inspection, Return to Vendor.
3. **Selling –** Sales order taking, including promising, but not including Customer Relationship Management (CRM). While CRM should really be considered part of the integrated solution, it is often treated as a separate system, with hooks into the ERP system, to access corporate data.
4. **Operations –** Including Manufacturing, Warehouse, Quality, HR, Payroll.
5. **Other –** Such as Fixed Assets, Projects, Product Configurator, Budgets, etc. as they may apply.

Implementing ERP generally means selecting, installing, preparing and going live with a new computer system, or interfaced systems. In modern times, these have become so fully featured that it is often possible to use one fully integrated system to handle all back office systems. However, sometimes it makes more sense to link best of breed systems for a particular business. Perhaps the distribution requirements are so intense that a Tier 1 Warehouse Management System has to be interfaced to the ERP engine. Or, the customer facing activities are so critical that a best-of-breed CRM system is indicated.

I will be using the expression "system design" a lot in this publication. All ERP systems should be started with a clear intent to limit customizations. Most are. But many end up regretting how much they resorted to customizing the solution. It has always struck me how often these changes happen to make the new system look like the old. But these are man-machine projects, and we're fallible.

My definition of "system design" does include customizations and custom reports, but I'm mainly referring to the way the solution is configured and set up, as well as the definition of the business processes that use the ERP system as the tool – design for how it's used.

What is ERP risk?

What stands out here is the shear breadth and depth of the scope. Many departments are impacted and the new approach means looking again at the way literally everything is done on a day to day basis.

Because the best systems are so well integrated, it can take only one group of dissenters or one department to undermine an implementation.

It's not out of line to say that a business implementing a new ERP is betting that business on the outcome. The cost can be many millions of dollars, and even for a small business, it is no less than several hundred thousands of dollars. The time can be multiple years if the implementation is spread over phases or multiple businesses, but it rarely takes less than six months. Rushed implementations can result in months of expensive triage.

Often the buyer of a new ERP system is doing so because a prior one either failed or was botched enough that a second attempt is needed. It really is a buyer beware situation. Not surprisingly, savvy buyers look to ways they can mitigate the risk.

Let's look first at a list of risks. I've listed them in order with what I consider the highest risk first:

Executive Buy-in

Most people put executive buy-in at the top of the list for any business project. A key difference with ERP implementations is that more executives have more at risk. If the team members see that their executive isn't behind the program, frankly, Sayonara! This kind of project takes a long time and inevitably has its setbacks and ups and downs. There's always conflict between people committing to this and also getting their daily jobs done. If your boss doesn't really want you committed, it's pretty hard for you to be consistently committed for the duration of the project.

It is best practice that one executive takes the role of project sponsor. It's important that the sponsor is:

- Committed to project success.
- Visible and engaged with the project team.
- Respected and trusted by their executive peers. The sponsor is seen to be representing them.
- Seen as someone the team can turn to for solving bandwidth and conflict issues.

When changing from one system to another it is inevitable that some things will work better and some things will work not so well. Since the decision was made to change, there will generally be more things improved than things that worsen. But when the team member complains to his recalcitrant executive boss about how painful those few things will be, those will be what get attention. You can see that this may also lead to the new system being customized to look just like the old one.

Often the new system is different because the whole underlying principle IS change. An example might be that planning was done very poorly on the old system, so a myriad of expedite-type reports and inquiries were added over the years. The new system, likely, has much better planning tools. A good goal for the new system is to morph from reactive shortage-driven expedite to proactive planning where exceptions can be caught before they become critical shortages. This would mean a purposeful transition from looking at things by order line to looking at them by item. Why expedite the same thing on ten orders when it could have been looked at once?

Here, the danger is that the executive may insist on the inquiries and reports being replicated. The result will almost certainly be a perpetuation of expediting and an undermining of the transition to managing by plan exceptions.

Unrealistic Expectations

Frankly, my experience is that ERP implementations frequently start with expectations too high for scope, budget and, perhaps worst, far too optimistic a schedule.

I put this down to unrealistic promises by the consulting and/or software vendor. It's a real problem in this business. This is an expenditure that the client almost always puts out for bid. It's a lot of money and can be a risky venture. It can pre-occupy the best people in an organization for many months. The sales cycles are often long and always competitive. Unfortunately, sometimes the most aggressive vendor, rather than the most reliable vendor, gets the business. That most aggressive vendor also puts huge pressure on all the other vendors to sharpen (or over-sharpen) their pencils. Classier vendors will under-promise and over-deliver, but, in my experience, these are the exception and they run a huge risk of losing business that they should win.

Customers know this is risky. Frequently they have been burned before, either in the same company or from prior employment. This is a key driving incentive for clients to limit risk and seek guarantees.

Core Team Buy-in and Bandwidth

The core team members are those employees that will work with the consultants to design how the new tools will be used. They need to be key players, most likely the same people most needed for the day to day success of the business. The project demands more of their time than any other players, close to full time, ideally. From project to project what the core team does versus what the consultants do varies all the way from the consultants doing all the design and the core team signing off, to the core team doing all the design and just getting product training and advice from the consultants. Clearly in the latter case, core team buy-in and bandwidth are critical to success. But even in the former case, relying on the consultants, I have consistently observed one of the following:

1. The core team isn't really engaged until the consultants present a delivered system. They have had no skin in the game and no real incentive to back the solution that someone else came up with to dictate their working lives for the forthcoming future. The approach is rejected as not meeting their needs and either:
    - The project is halted.
    - They go live anyway but the core team hasn't become familiar with the new solution and as the inevitable teething issues emerge, people resort to old ways, perhaps even trying to keep using the old system.
    - They go live anyway but depend on the consultants being tenured as their expensive ongoing support.
    - The consultants are reprimanded or replaced and asked to perform a do-over.
    - Or, the consultants are reprimanded or replaced, and we start over with the core team now instructed to get more engaged in the design and testing.

There's a big risk that second time around that the design is going to look a whole lot like the legacy system by expensive force-fitting and customizations.

2. Alternatively, the consultants actually do a pretty good job, but the core team hasn't been close enough to be able to support after go live and again, either:
    - The inevitable teething issues emerge and people resort to old ways, perhaps even trying to keep using the old system.
    - Or they depend on the consultants being tenured as their ongoing and rather expensive support.

Bottom line: a weak and/or disengaged core team can be a major threat to project success.

Inadequate Testing

Inadequate testing can result from a weak core team, weak consulting advice, or from pressure to meet unrealistic deadlines. Surprises discovered close to or after go live present real and significant risks to overall success.

Client Subject Matter Experts Buy-in and Bandwidth

The Subject Matter Experts (SMEs) that emerge during the project are usually what I call the sub-core team. These are the delegates that the core team train within their design to validate it, create test scripts, perform tests in successive cycles and provide job-specific training to their end-users. SMEs are generally more numerous but part-time on the project. They are first line support once live.

In a healthy implementation, these are the doers. The quality of their work dictates the quality of the implementation. If they cannot or choose not to invest their effort, then scope, duration and budget are all threatened.

Funding

Funding can get pulled for a lot of reasons the implementation team has no way of mitigating. The whole enterprise can be out of business. The company can get sold to another business that has a different solution they have made their standard. A new executive can arrive and decide to pull the plug. Just keep in mind that a project that is going well and is running within budget is much less likely to have its funding pulled.

Funding may also have been inadequate from the start, of course.

Team Training

Modern back office staff is comprised of knowledge workers. Employers increasingly realize that it's sub-optimal to just tell people what to do and then manage them to do it. More and more businesses foster creativity by empowering their team and leveraging grass roots creativity to continuously improve processes and competitiveness. Running a successful implementation shares many characteristics with those needed to run a successful business. I suggest that you want creative, empowered, knowledgeable client staff on the implementation team. They know their business. They know where they already do well and where their legacy systems don't work so well. I'd also suggest that one of the most important things the consultants can do is to educate and train the incumbent staff on how the new tool works. Yes, it's great if they can ADVISE on how it might be used and suggest some best practices, but sooner rather than later, if we are successful, they are leaving. Critical as it is for the consultants to quickly understand the client business, I'd argue that it's much more important to educate the client on the tool than educate the consultants on the business. But both are important, of course.

Clearly, poor training will increase risk of failure. This training includes:

- Training the core team on the full breadth of the product so they can design (with help).
- Training the sub core team (SMEs) on the designed solution for refinement and validation.
- Training the end-users on how they will use the new system to perform their duties (and perhaps one upstream and one downstream).

Ill-Defined Scope and Missed Requirements

Some would put this risk higher on the list. The inherent scope of ERP is huge. It's too easy to discover important requirements too late in the project, and the greater the scope, the greater the risk. This is much less likely if the client team is heavily engaged – much more likely if they are depending on consultants for the design. Executives frequently understand the need to limit scope and instruct the troops that they won't get everything they want in phase 1 and that we have a zero-customization policy. But as the team dives into the product weeds, frequently:

- The temptation to make the solution look familiar, just as it was in legacy system, is immense. This actually is two-edged. It's very costly, and generally, since the legacy system fell short, why spend all this effort to reproduce it?
- Non critical but appealing features often get added to scope, perhaps surreptitiously. On occasions, this can be entire modules being added to phase 1. This is scope creep.

End User Buy-in

If the core team and the SMEs all bought in, I've never seen an issue with end-users. You still need to do a good job of training, as mentioned already.

Turnover

Clearly this depends on who we lost! Losing the executive sponsor generally presents the highest risk, followed by core team members and sub-core team members. Losing a consultant SHOULD NOT be so important. The consulting firm ought to have another one, hopefully waiting in the wings.

Consulting Quality

By now, you will probably guess that I will declare this to be less important. And you'd be right! But perhaps for a different reason than you might expect. A good consultant, one who really knows the product, who can articulate best practice and has good implementation project experience, can make a dramatically positive difference to a project. The relevant point is that a bad consultant should be quickly identified, and it's not hard for the client to pressure for a replacement. Indeed, it's not unheard of for them to replace the consulting vendor! In other words, a bad consultant can damage a project, but it's relatively easily fixed.

Software Fit

This is not the risk it used to be. ERP software has matured to be much more comprehensive, flexible, integrated and reliable. Yes, there are differences in how well it accommodates a specific industry and there can be surprisingly significant differences in prices and cost of ownership. I've seen projects hurt badly with some gap being missed until they were live and even with some bug emerging after go live and proving difficult to isolate and fix. But I've never seen a project fail for either.

Software Quality

As just stated, it's rare that software quality threatens project success. More likely, a buggy new release or new module could incrementally demoralize the implementation team and exacerbate other risks. As ERP software has matured, this has become much less a source of risk.

Over a period of 40 years, I have seen many combinations of the above risks in action. I can tell you that in all those years, no two projects were the same, even in their design, let alone their risk profiles. I'm happy to say that the only complete failures I've witnessed only terminated the project well before the go-live and were outside of the control of the project team. One was the client company going out of business and another was funding being pulled because of a downturn in the economy/industry.

If there has been a pattern I have observed, it's been that the more troubled projects were always associated with low client ownership. Sometimes this has been because the client staff could not (or would not) allocate time so the consultants were asked to fill in. Sometimes the implementation was even inappropriately sold as turnkey so the consultants were required to do the whole thing. On smaller projects, generally looking within a single department (such as implementing a warehouse management system), you can get away with savvy professionals coming in, designing a solution and even implementing it, training the client as part of the go live process. I've NEVER seen this work in ERP, and I don't expect to see it in my lifetime. It's just too dependent on too many people needed to SUSTAIN it later. If consultants do the work then consultants own the result. The people who now have to use it and make it work, simply don't know the tool well enough. What's much worse is that they have put no

skin in the game and have nothing to personally lose if it fails. I've also never seen a go live, that didn't demand some blood, sweat and tears afterwards - no matter how well the implementation went. If it's easy to simply throw your arms up and blame someone else's design, that's quite likely to be exactly what happens.

If client staff doesn't OWN the solution by go live, it's going to hurt. I would suggest that the earlier the client takes ownership the better the chances of success and the quality of the solution. That's why I'm so keen that the core team is thoroughly engaged in and committed to the design of how the system will be configured and used.

Now take another look back at the list of risks and ask yourself which of these are diminished by strong client team ownership and which are made even riskier by low client team ownership. My point is that there is nothing more important than the client taking ownership to ensure project success. Indeed there should be nothing more important to the consultants than doing everything they can to gain that client ownership. Please note that for some software vendors and consulting firms this can be counter-intuitive. They want more business, not less. For too many that means fostering dependence on them. For me this guarantees missing scope, missed dates and running over budget.

Risk Mitigation

There are several things the client can do to mitigate risk. These include, in no particular order:

Retaining Experts

I've been implementing or helping others implement ERP solutions since 1972. Back then, I had just moved from Production Control to head up IT. Recognizing they needed an ERP system (it was called MRP2 at that time), they sent me off to be educated by George Plossl, one of the three founders of APICS. This turned out to be a career changing event for me. We formed an internal team and looked at several alternative systems. We selected MAPICS, an MRP2 system developed and sold by IBM. We formed an internal team, studied the manuals, embraced APICS training and implemented MAPICS without any consultant help.

So it can be done without consultants, but you need some smart, motivated and knowledge hungry people on your team - and probably extra time to get it done as well as some tolerance for some false starts. Almost all modern day implementations are done with the help of experienced consultants. Often these are people who used to have functional roles in businesses and participated in one or more ERP implementations. The implementation process has been refined and improved over the years, and there are many consultants who have been engaged in 10 or more implementations. I've done six as an employee and close to 30 since then in the 17 years I've been consulting.

Just like all human endeavors, experience helps, but retaining a few experts close to full time for six months or more gets expensive.

Multiple Location Phases

This is the "big bang" debate. You've all heard the answer to how do you eat an elephant – one piece at a time. This mitigation means implementing in pieces, or phases. The challenge here is that the primary benefit of a new ERP system is how integrated it is. Most businesses have already gone a long way optimizing within each department, and the inefficiencies and mistakes are more to do with how the departments interact. So how can we break it down to smaller, more manageable and less risky pieces, without losing the benefits of an integrated solution?

One of the easier break-outs is by company if the client operates in more than one legal entity (e.g. international locations). You implement in one company, fully integrated, and use it as the template for the next.

Multiple Functional Phases

One of the more common approaches to phasing is to implement Financials (GL, AP, and AR) first. This was more appropriate when the legacy financials were also somewhat decoupled from Operations. The drawback is having to write an expensive but throwaway interface between the new financials and the old operations platforms. I've never seen this done the other way around, but I guess it's feasible.

You really can't separate selling or buying from inventory, with one exception: if the intended goal includes a separate Warehouse Management System (WMS) for warehouse execution interfaced with ERP planning, then I have seen clients implement the WMS first (temporarily hooked to their legacy ERP system) and follow with a new ERP go live.

I have seen where selling and buying were implemented and production came later, but that was about 30 years ago. This is less practical these days with increasingly sophisticated and complete sets of functionality.

One client I assisted came up with the term "Base Camp" to describe the approach we took there. The principle was to get to an integrated platform with the least possible functionality to run the business and ditch a sub-par legacy system as fast as possible. The CEO declared that he would do this even if it sacrificed some legacy capability. Go live would then be followed by, what he called, ROI trips to the summit. Rather than the functional phase breakouts just covered, this approach to mitigating risk is very appealing to me.

Parallel Running

Believe it or not, I had a client insist on a parallel run for a couple of months after go live. They got away with this because the legacy system was so manual and excruciatingly painful that the relatively small effort of also transacting through the new, much more automated system could be handled with overtime. This was a great opportunity to flush out any issues (especially process issues) that testing may not have already surfaced.

But there are very few businesses that can afford to transact everything twice, even for a limited period. These days, most enterprises are running too lean to be able to do that.

Formal Readiness Assessment

A good project plan will include at least two milestone events where readiness is assessed. It's better to recognize a show stopper and delay cutover than to go live and then halt, perhaps having to resort to the old system. It makes sense to do this readiness assessment right before cutover dry run and right before go live. This might be as simple as getting all the stakeholders in a room and asking everyone, "Are we ready?"

But "ready" needs to include:

- Hardware infrastructure
- Software stack
- Defined business processes
- Everything tested including interfaces
- Trained end users
- Support in place
- Secure
- Compliant
- Disaster plans in place

Heavy Training

It's hard to be over trained. These are all knowledge workers who need to be knowledgeable about the new tool and how they will use it. Clearly a poorly trained team adds to risk.

Heavy Testing

Test, test and test again. This is tedious, and it's human nature to avoid tedium. But this is one of the surest ways to reduce risk. I often describe the whole implementation as a succession of test cycles:

1. **Proof of Concept.** This might be conducted by the consulting company and may even be performed by their pre-sales team. It's aimed more at proving the base software than a fully configured solution, but some attempt should be made to make it relevant to this client's requirements.
2. **Conference Room Pilot 1.** This immediately follows the core team's design process and may also be the first introduction of the system to the sub-core or SME team. Typically it's a happy path (simplest scenarios/use cases) through the main process flows:
    - Quote/Lead to Cash
    - Procure/RFQ/Requisition to Pay
    - Plan to Produce
    - Financial Closing

    The test will use keyed in data for master data such as items, customers and vendors.
3. **Conference Room Pilot 2.** This is the first testing being done by the sub-core team using scripts they have developed within the core team's design. Ideally, it will use master data migrated from the legacy system(s). This will expose gaps that need to be fixed. None or few of required customizations and custom reports will be in place. Security may be wide open.
4. **Conference Room Pilot 3.** This is the second testing being done by the sub-core team, rather than the core team, and will be aimed at testing that the gaps are all fixed. Customizations and go live critical reports, including customer reports, should be ready. Ideally, security profiles will now have been established. All data that needs to be migrated at go live ideally will be migrated here. "All data" means all master data, as well as open and remaining partial orders if there are too many to be keyed.
5. **Go Live Dry Run, including testing the cutover process.** By now end-users have been trained and should be engaged in this test cycle. GL balances, including inventory, will be validated against legacy item costs, also. The client may choose to take one day of actual transactions and repeat them in this test, even if it takes several days to accomplish that. Ending balances would then be reconciled.
6. **Actual Go Live.** The ultimate test!

Depending on results, one or more steps may be repeated. The Proof of Concept is optional. If the second pilot goes really well, the third might be dropped.

Client Ownership

I consider this to be the most significant way to reduce risk. I've found that people really know their own processes. They live them. But they may not be so good at articulating them, with all their nuances, dangers and tricks, to a third-party consultant. Performing their processes day to day can actually trick a person to take a certain aspect for

granted, so that they are surprised it wasn't obvious to the consultant without it being told. People can find it hard to imagine things being done differently.

Relying on even experienced consultants, instead of client staff, to catch all the requirements is optimistic. If consultants are given the lead, checkpoints can be built in when the consultants present THEIR solution and the client team challenges them. But this typically leads to expensive cycles of redesign where we are essentially training the consultants to satisfy their counterpart client. Since the client is not savvy about the full system capability and is really only armed with how they do things today, this can lead to increased pressure to replicate the old system in the new one.

This process also automatically creates an US-versus-THEM situation in which the client team is trained to be critical of the system and the vendor is incented to try to replicate the system being replaced. In my opinion, the "us vs. them" aspect is much more dangerous than the risk of missing requirements.

Contrast this to a world where the consultant is trained to bring the client up to speed fast on the tool, and it's clear the client is responsible for the design. The consultant is now essentially SELLING the client on new ways of doing things and the client has a specific role of designer rather than critic. This is MUCH more conducive to a better design, one that solves old problems and is more effective. Clearly the whole process is more collaborative and knowledge transfer driven.

The client owns the design, script writing and all testing. Naturally, by the time we go live, they will really know the product, and, more importantly, how to use it. Even more valuable, it's the client's design. By the time the consultants leave, the client team will be dramatically more capable of supporting the system and sustaining its performance. Performing the subsequent phases, with all their deferred benefit, is much more likely to happen and happen sooner and can be achieved with less, or maybe even no consulting support.

Fixed Bid

A fixed bid means the consulting vendor commits to a maximum charge for the implementation rather than an hourly charge for the consulting. I deliberately left this one until last. This approach, to me, is dangerously seductive. Indeed, if I had been braver, I would have put fixed bid in the project risks section instead of one of the ways to mitigate risk. The thinking goes something like this:

I'm buying a system from you. You are the experts. It's risky. If you want my business, you are going to have to take some of the risk. You tell me up front what it's going to cost me and you guarantee it. If we go over budget you eat it instead of me.

Seems reasonable, right? But there is an assumption tied up in here. The assumption is that most of the risk factors are in the hands of the consulting company rather than the client. While the consulting company starts the process knowing much more about the tool than the client, it's not generally the tool that determines success or failure, it's how it's used. Think table saw. Think tennis racket. At the end of the day, the consulting company won't be using it, the client company will be. A good consulting company will see the implementation as primarily knowledge transfer. By go live, you will know the system almost as well as the consultants do, and you'll know how to use it for your business BETTER than the consultants will. But knowledge transfer is a two-way process. I would argue that there is much more risk of a poor, disinterested learner than a poor teacher.

We just got through talking about the risks associated with a lack of client commitment. We saw how important promoting a collaborative effort was. But now we're even making the contract somewhat adversarial. Why? It's because scope suddenly becomes the bone of contention. The vendor is a business. If they are being held to deliver something by a certain date within a certain budget, they better know exactly what they are delivering

– and it will be a whole lot more specific than a successful go live. It will be a lengthy list of what is in scope and what is not. A list prepared before the project starts, so likely wrong.

Now the client is contractually guaranteed timeframe and budget. The team can focus on getting the most value for their money - in other words, expanded scope. "Of course we need to be able to do that, we can't run the business without it...It goes without saying it has to work this way, we'd have to hire another person to do it that way…It should have been listed in scope and you know it." And so on. This is a recipe for wasteful cost simply over-defining scope and repeated fights over the list and whether costly change orders are needed. Under fixed bid the vendor may be tempted to cut corners, thereby increasing risk.

To me, though, it's the built-in adversarial aspect that makes me resist this option. It completely sets the whole project off on the wrong foot. You can picture the client, standing there arms on hips, saying "OK. I've paid. Now deliver." And we've seen that, useful and expert as the consultants are, they don't deliver the solution. They help deliver a client team that is capable of delivering and sustaining the solution.

Also, the vendor isn't stupid. They know that most of the variables, most of the risks, are in the hands of the client. If they have to bid fixed, it's going to be bid high - an implementation estimate plus an expensive insurance policy. And they will be forced to specify scope so tightly that even reasonable discoveries become change orders. The administrative costs alone will be painful. Such waste!

Ask yourself what starting with a fixed-bid contract will do for client ownership. They are in opposition. Lack of client ownership leads to "US" vs. "THEM." Fixed bid enshrines that in a contract. Let's review the list of risks:

- **Executive Buy-in.** Having a fixed bid in their hand does nothing to incent the executive to invest their time and energy committing to the project - quite the reverse.
- **Unrealistic Expectations.** Fixed bid certainly does nothing to reduce this risk and likely makes this worse. In the client's mind, the vendor has committed to "meet their expectations" at a fixed price. A scope war and change order cycles are even more likely if expectations were unrealistic.
- **Core Team Buy-in and Bandwidth.** Again, a fixed bid in their hand does nothing to incent the core team to invest their time and energy committing to the project - quite the reverse.
- **Inadequate Testing.** Having a fixed bid does little to promote more testing. There's every chance the client will think "You're delivering it. You test it first."
- **Client Subject Matter Experts Buy-in and Bandwidth.** Again, a fixed bid in their hand does nothing to incent the team to invest their time and energy committing to the project - quite the reverse.
- **Funding.** Having a fixed bid does nothing about this risk.
- **Team Training.** Having a fixed bid does nothing about this risk. It can make it harder to force team members to attend training.
- **Ill-Defined Scope and Missed Requirements.** Having a fixed bid does nothing to incent the client to be more helpful identifying missed requirements early. In their minds they are buying a solution and the vendor is paid to come up with a solution they like.
- **End User Buy-in and bandwidth.** Again, a fixed bid does nothing to mitigate this risk.
- **Turnover.** A fixed bid does nothing to mitigate this risk.
- **Consulting Quality.** You could argue that the vendor is on the spot to deliver and will put their best consultants on the job. But that's not necessarily the case. This is a fixed bid. They may just as easily place their best consultants with clients who are paying by the hour.

- **Software Fit.** A fixed bid does nothing to mitigate this risk.
- **Software Quality.** A fixed bid does nothing to mitigate this risk.

So, how did fixed bid reduce risk? Can you see why I was tempted to list fixed bid as a source of risk, instead?

Conclusion

My submission is that there is nothing more important in reducing risk than securing client ownership. I suggest that the executives, core team, sub-core team, software vendor and consultants should all obsess on winning early client ownership. It is the most effective way of mitigating risk, with testing coming in second. And it is also highly beneficial for other reasons:

- The solution is far more likely to sustain itself after the consultants leave.
- The client will save on consulting costs.
- The solution is far more likely to accommodate any client uniqueness.
- The client will have a ready pool of internal resources for future phases.
- The consulting company can register another reference account and more quickly move on to the next success-hungry client.

Client ownership doesn't happen immediately. Since the consultants come in with training, experience and standardized methods of implementation, they naturally have an early leadership role, especially since early stages focus on training and configuring. But as time passes the client should take more and more ownership of:

- Design
- Testing
- Planning – especially cutover planning
- Documenting
- Support

. . . And all parties should facilitate this transition process.

Again, let's go through the list of risks and ask ourselves if increased client ownership of the design and implementation reduces these risks:

- **Executive Buy-in.** This is the first and most important target for securing buy-in.
- **Unrealistic Expectations.** Securing ownership won't change the initial expectations, but the gaps will emerge much quicker, and with a team committed to the project, if the expectations were unreasonable, they should more quickly reconcile to this.
- **Core Team Buy-in and Bandwidth.** This is an important target for securing client ownership.
- **Inadequate Testing.** Client ownership should mean much higher testing quality as well as a much more capable client.
- **Client Subject Matter Experts Buy-in and Bandwidth.** This is an important target for securing client ownership.

- **Funding.** Client ownership likely has no impact here.
- **Team Training.** In my book, client ownership dramatically incents learning.
- **Ill-Defined Scope and Missed Requirements.** Client ownership helps here by the collaboration of consultants who know the product and the clients who know the requirements.
- **End User Buy-in and Bandwidth.** Like we said - if the executives, core team and SMEs are all bought in, the end users will typically follow.
- **Turnover.** I'd argue that while the client owning the project won't make turnover more likely, it just might reduce it.
- **Consulting Quality.** Client ownership certainly doesn't threaten this. I'd argue it permits the consultant to focus on what they should be good at – knowledge transfer.
- **Software Fit.** Unaffected.
- **Software Quality.** Unaffected.

So, not only does stronger client ownership mitigate project risk, it promotes scope quality, speeds up implementation, and reduces consulting costs. Seeking client ownership deserves to be a cornerstone of client implementation, as well as a cornerstone of consulting methodology.

This article is authored by Nigel Cox, a lean inventory and manufacturing expert at enVista. For more information, please contact us at 877-684-7700 or info@envistacorp.com.

Nigel Cox is Managing Partner and VP of Services for Enterprise Solutions at enVista. He is an established expert in Lean Principles, Quality, Supply Chain Management, Manufacturing and Demand Planning. His experience spans a variety of industries and specialties that includes Engineer-to-order, Assemble-to-order, Semiconductor, Construction, Pulp and Paper, Textile and Apparel, Light Assembly and Fabrication. Cox is also Master Certified for Microsoft Dynamics AX and is a respected expert, especially in Product Builder and Lean Manufacturing.

Nigel is a certified fellow in Production and Inventory Management, certified in Integrated Resource Management and has been involved for many years with APICS, delivering certification workshops and as a speaker. He also teaches bachelor level business administration classes.