

ERP System Acquisition Project Planning

ERP system acquisition projects are rarely deemed to fail: usually buyers end up buying, and then it's up to the implementers to make it work (or not). All too often ERP acquisition means little more than system selection and contract negotiation, perhaps going so far as to include the contract for implementation. In fact the acquisition phase is the only real opportunity to put in place the mechanisms that - *if* they are implemented - can later ensure implementation success. This is the opportunity for the buyer to control the meaning of implementation success in concrete terms that buyer, system provider and implementation partner can accept as being in their own best interests. This is, in short, the only time when overall project success can be conclusively defined in enforceable, concrete terms that all parties can willingly agree to be held to. That this rarely happens should not obscure the fact that the techniques for achieving it are well understood and have been used successfully by some companies for many years.

The amount of time and effort that go into system selection varies radically from company to company. Some companies spend years at it; others (usually subsidiaries) are told which system they will buy before they begin their acquisition efforts. Companies in the first group aren't necessarily more successful than those in the second. As important as system selection can be, a well done system selection in and of itself is never enough to guarantee implementation success.

Let's take a look behind the statistics about ERP implementation failures and overruns. Almost every ERP vendor has a list of multiple successes – their systems have been proven to work many times over. Unfortunately, they also have many troubled implementations. The fact is that, in and of itself, the choice of an ERP product guarantees nothing. Success happens to them all, but so does failure.

Likewise, most ERP implementation companies also have long lists of happy clients; they also have other clients they'd rather not talk about. That is, their methods have been proven to work, but only some of the time.

Why do the same products sometimes work and sometimes fail? Why are the same people sometimes able to implement them and sometimes not?

The short, unpleasant answer is that ERP implementation success requires that the company acquiring the system do its homework, because failure follows from not doing it. But why wouldn't anyone do their homework in matters as critical as this? Usually because they try to start some of it long after it should have been completed, and often because they try to outsource work that only they can do for themselves.

What critical work ought to be done prior to ERP system acquisition that is often deferred to the implementation phase, or simply never done at all? *Why* should it be done prior to acquisition? *Why can't* it be outsourced to the implementation provider?

Consider an obviously false choice: two hypothetical attitudes toward buying an ERP system, phrased in excessively simpleminded terms.

- Go ahead and buy one without an agreed understanding of what you need or how it will be implemented or who is responsible for making it happen, or
- Before you buy anything, agree with your system vendor and implementation partner(s) about what you need, who will provide it, when you'll get it and how much it will cost.

Who would not agree that only the second approach makes sense? *Nobody*. Who would ever choose the first way? *Nobody*.

So *everybody* thinks that they acquire ERP software and implementation services based on a shared understanding between all parties, a clear business understanding that covers all significant requirements and can serve as a roadmap for implementation. That this *doesn't* happen much of the time is manifestly obvious – otherwise, why does any project ever fail, to any extent? – but how can so many people be so deluded about a process that seems so clear, so fundamental?

The short answer is that ERP buyers and sellers *do* come to agreement, but the agreements they come to are way too general and abstract. They don't describe expected use and behavior of the new system in nearly enough detail (or in the *right kind* of detail), and for this reason they can't (and *don't*) govern the day to day tasks of the implementation. When a project is not being governed by an explicit, shared understanding of what is to be done (and why it is to be done), it is being governed by something else – and no matter what that something else may be, whatever shared understanding the parties thought they had achieved is not driving the effort.

Pre-sale agreements are usually reached through a combination of a proof of concept demonstration and gap analysis, which identifies required changes to both software and procedures. Implementation tends to be viewed as simply an elaboration of this pre-sale process, but in fact implementers work in far different terms – wherein lies the problem.

ERP packages are monstrously complex. Even the simplest are complex enough that no one person – *no one*, anywhere, including their (multiple) architects and authors – can even begin to fully understand all the ways they might behave. Their behavior is controlled by switches set by the implementation team, more than 10,000 at the most complex end of the scale but never less than a few hundred. Even 100 two-position (either/or) switches can be set in a huge number of combinations (roughly, 1 followed by thirty zeros). Of this immense set of possible ways to configure any modern ERP system, only a tiny fraction has ever been implemented. Even if a product has been configured in thousands of ways for thousands of companies, this still represents only a tiny fraction of the possible configurations.



No implementer is equally expert in all parts of the same system (which is why they come in teams). The system itself, however, is not so compartmentalized. Any one system function has potential implications for every other system function. Of course, for any one configuration of any one function, not every potential system-wide effect is realized, but some are.

Ideally, the implementation team would somehow notice, evaluate and manage all of these actual effects. In practice, this is horrifically difficult; still, it can be done, and sometimes it is. *But only until the target moves.* And the target usually *does* move.

The "target" in this sense means *the implementation team's working understanding of their task*, and nothing else. It is not necessarily the same as any written spec or contractual obligation, but it is the only thing that counts. Before the fact, the consequences of any change to the implementation team's working understanding of their task are utterly unforeseeable, ranging from inconsequential to catastrophic; but even inconsequential changes have an impact on the project schedule because they must be evaluated and proven to be inconsequential. All too often changes are somewhat more than inconsequential. Impacts throughout the system have to be discovered, assessed and dealt with. Often changes come in waves, one change requiring many more. Work previously thought complete has to be redone and retested, and more work added to the schedule as the full impact of each change sinks in.

Why and how does the target move? Implementers tend to view any change to their understanding of their job as *scope creep*. Scope creep – a change to the user's requirements while the project is underway – does happen. It can be controlled but only if the user's requirements have been rigorously analyzed prior to acquisition; even then, controlling it requires deep project management discipline coupled with strong, informed executive support.

But most instances of what implementers call scope creep – most of the changes to their working understanding of their task – do *not* represent changes in requirements from the users' perspective. Rather, they are just unpleasant flashes of enlightenment occurring as implementers suddenly come face to face with some unperceived aspect of the users' expectations. They are *surprises*, nothing more, and they invariably represent hidden disconnects between user and implementer. They happen because the implementers, with the best will in the world, miss important aspects and implications of the user requirements; and *that* happens because these requirements are not fully presented in concrete terms that the implementation team can work to (and can be *required* to work to, and can be *measured by*) on a day to day basis.

It is possible to state the user requirements in ways that vendors and implementers can understand and conform to. It isn't always done but it is possible, and it has been done many times by many companies. It certainly isn't something we claim to have invented. Not only did we not invent it, we can't even begin to do it for anybody. We don't think anyone can. It is something that only the company purchasing the system can do. No one else can come in and explain the details and nuances of the company's business to itself, let alone to anyone else. The most that outsiders can do is show a company *how* it can explain itself to the world (and to itself!) in ways that translate into ERP implementation success. ***This is what we do.***



Even a perfect description of requirements and expectations is not enough by itself. It needs to be stated in terms that the implementation team can work to, that their own management can hold them to and that the implementation vendor as a whole is contractually bound to respect. In other words, all real requirements and the methods employed to fulfill them must be gathered into a full and complete Statement of Work (SOW) around which the implementation contract is written and the implementation project is managed.

Requirements cannot be imposed after the contract is let. Implementers must work to something, and they will set their own targets if none are set for them, targets that will not be based on their client's understanding of its own business. After the contract is signed, implementers will rightly view attempts to hold them to unstated specifications as scope creep at best, interference at worst. And implementers will not usually agree to be bound by a sufficiently detailed SOW that they had no part in negotiating. They have every right to insist that their own implementation methods and management practices be written into it.

Once the acquisition is over, the opportunity to control the implementation has passed.