

“THE PROBLEM WITH REQUIREMENTS”

AKA

THE REQUIREMENTS OF THE PROBLEM

PART 1 OF 2

The VP of Marketing calls the CIO into his office and says “Our sales force is complaining that our customer database isn’t working for them. I want them to have a state-of-the-art Customer Relationship Management system which will help them develop stronger relationships with our customers.” The CIO tells his staff to put together a project plan with a budget and timeframe for implementation. The staff follow their usual process of comparing vendor packages, performing due diligence with vendor-supplied reference accounts, and selecting a vendor who’s software fits within the overall Technology Strategy and will interface with the existing systems. The budget and timeframe are presented to the VP of Marketing for approval, she signs off and the implementation project is launched. The first major activity is to “Gather Requirements”.

Wait! How did we select a solution much less specify a timeframe and budget if we don’t already know what we need to accomplish? Let’s call the information to be produced by the “Gather Requirements” activity the “***Requirements of the Solution***”. These are often spelled out in tedious detail by asking the operational staff what they need to do their job better. Spreadsheets are produced with hundreds if not thousands of individual bits of detail. Super-users who love to play with technology throw out suggestions for database design (“we need a table to hold contact information”), UI design (“we need a link to the sales leads system”), and functional details (“the system shall notify the user if there is an existing customer record already in the system”).

There is an implicit, usually unstated and always unverified assumption that the sum of all those little “Requirements” will actually produce business value, which in my opinion is the reason we do projects in the first place. Let’s call these higher level business issues the ***Requirements of the Problem*** (or “Opportunity” if you are an optimist). Looking back to the example above, let’s see if we can identify the requirements of the *problem*. Is it to install a state-of-the-art Customer Relationship Management system? Actually no, that is a candidate solution. Is it to make the sales force happy? Well, that may be a desired by-product but there is no actual business value generated. How about “help them develop stronger relationships with our customers”? YES, that’s it! Or is it? What business value will be generated by stronger customer relationships? Has the value been quantified (really, not just a swag) and all have the steps been laid out to achieve that value? Have metrics with dates been clearly stated? In my experience the answers to these types of requirements questions range from non-existent to a half-page “business case” supported by a few spreadsheets and a PowerPoint presentation with lots of nice iconography (all related, by the way, to the Solution). How many times have you seen a project implement a solution and be declared a success however if anyone looks the actual business value generated it is very much less than what was used in the original business case? Or even negative? Why does this happen so often? Why are so many IT shops really good at implementing *a really great solution to the wrong problem*?

Why does this happen? Consider how much effort went into defining the problem / opportunity? Note in the example above the “requirement” already presumes a solution (a Customer Relationship Management system). What if the real problem is high turnover in the sales force? Or perhaps it is duplicated and/or conflicting data in the existing system? Or perhaps lack of training? Or, could it possibly be a broken business process?.

None of these will be improved by virtue of a system implementation. Presumed solutions such as these often are the result of a solution vendor selling directly to the business, or perhaps the business executive was golfing with a peer from another company who told him a great story about how package “X” improved their business results. Even if this is actually true, how would we know if the fundamental problems are congruous?

Perhaps we should assume that the business executive will already have analyzed these possibilities. I suppose there are times where that is true, but in my experience they are rare. It can also be politically sensitive to ask such questions and great tact is needed. The biggest problem, however, is that if the project is already launched and the budget and timeframe set then it is too late to ask these questions. In order to have any effect on requirements specification at this level we must strive to get involved in the governance processes which precedes project approvals.

Some egregious examples from my personal experience include a customer database stored in a Mainframe IMS system. The business users were sold on an idea that IMS was the problem and a new open systems solution would make things better. Many millions of dollars were spent acquiring and customizing a vendor package and then all of the erroneous data was transferred from the IMS database into the new system. The project was declared a success however the business value was negligible because the root problem had been carried forward.

Another classic example involved a Finance department using a mainframe based batch accounting package. A vendor came in and said they could improve the efficiency of their processes by switching to a new real-time system. The \$7mm project went smoothly right up to UAT testing at which time the users realized the new real-time system did not support management approval of a batch of transactions prior to posting. They demanded last-minute changes to layer batch controls on top of the real-time system which of course destroyed any business benefit the new system might have delivered.

The best case is for us to get in front of the governance process and help drive adequate description of the business problem/opportunity, but until then we can at least try to point out the issues when we see them at the beginning of a project. Some warning signs include the project manager saying “now that we know the cost of the solution, we need to find enough business benefits to make the ROI come out positive”. Or the current system is used by 3,500 people and there is no money or time in the implementation plan for training? Or perhaps there is no implementation plan at all (the development of a good implementation plan will sometimes address some of these concerns).

It is commonly said that 75% of IT projects fail to deliver results on time and on budget. It is interesting to note that when this statement is made there has been a subtle and often un-conscious shift in perspective. The question being asked is “Was the committed business value (or ROI) generated? In my terms this is asking “Were the Requirements of the Problem met?” IT may have delivered on the Requirements of the Solution, but now the project is being held to the higher standard.

Consider the following uber-simplified scenarios:

<i>Outcome</i>	<i>Adequate Definition of the Solution Requirements</i>	<i>Adequate Definition of the Problem Requirements</i>
Success	Yes	Yes
Failure	No	Yes
Failure	Yes	No
Failure	No	No

In words, the IT Project succeeds according to the above definition when they develop adequate Solution Requirements and those are based on adequate Problem Requirements (and good execution, of course). On the other hand the IT Project will fail if either of those two conditions are not met. All other conditions being equal, the chance of IT Project Success can be no greater than 1 out of 4. In reality all other conditions are not equal so the chance of success is actually somewhat less.

In this forum and in countless others there have been endless discussions of how to improve the art and science of eliciting and analyzing the “Requirements of the Solution”. While extremely valuable these are directed at improving only one of the four scenarios, leaving two of the four to chance or assumptions. Is this all there can be? Is there no hope for the 50% of failed projects? I believe there is, and in the next installment I will propose a method for leveraging the success of those efforts.

To be continued...
