The three T's of projects – a White Paper

You've all heard of the 3 R's that form the basics of education Reading, wRiting, and aRithmetic, and let's be serious, the concept is so important that they even willing to stretch the English language to fit it in. So what about Business Systems projects, what are the three pillars on which they are founded? My view is that a successful project will be founded on strength in the 3 T's:

- Training
- Testing
- Data Transformation

Get these three correct and, assuming your design is correct, you should have a successful project. Get any of these wrong, and you should expect trouble in the final solution.

So let us discuss each of these in detail, what do they mean? how can we make sure we can get them right?

Training

It seems fairly self-evident that with a new system, people need to be trained on how to use it. This has never been in doubt in any project I have been involved in, so why do I say this is an issue that needs close attention?

Well the answer is that it's not just the system that is important, training through a project is not restricted to the end where you train the end-users in how to use the system, it's part of an overall change management and knowledge transfer program that takes you from one state to another, and the education starts from day one. However, when budgets are constrained by negotiation, it's often the easiest of areas for your implementation partner to cut. So attention to the knowledge transfer education through the project is important.

What do I mean by training? Well there are a number of elements, not just class-room based training and not all of it should necessarily come from the implementation partner. Here are just a few elements to consider when designing the training program:

1. **Strategic Objectives** – I know you may not necessarily want the business long term strategy to be widely broadcast, but the project team need to know how this fits into your general IT strategy. Do they have to get all of their requirements satisfied in one phase, or is this a rapid implementation to get the foundations in place that you will then build upon? Is there going to be future projects covering elements of functionality, or is this an all-or nothing complete system? Whatever the answer, you need to make sure the project team is fully aware of this.
2. **Project overview** – From the kick-off of the project, the team that you have assembled need to understand their part in the activity. What role does each of them play? This is not a one-off training session either, this needs to be a continuous process of communication through the project.

3. **Project skills** – Do the team know how to deliver their part in the project, do they know how to clean and prepare data, deliver test scripts, carry out testing? Making sure they know how to do the tasks they are assigned will eliminate delays in the project.
4. **Business skills** – some of the concepts you may be introducing to the business may be new, so do your business users who will be involved in the design and delivery of the functionality understand the concepts? It may be necessary to deliver some more generic concepts to help them understand the capabilities they could be implementing.
5. **System Overview** – High level overview training of the system functionality can also be useful, as this may generate some ideas of how to improve processes.
6. **Super User Training** – Specific detailed training of the functionality in the areas you are implementing for the project team enables them to understand the context of the system they are implementing.
7. **End User Training** – Specific focussed training to the end users on how they system you have designed and had built for them

Ensuring that the training is right for the team is important, and the first thing to do would be to form a training assessment and identify what is required.

Testing

Business Systems are complicated things – they are a combination of technical software, configured to meet your specific requirements, and data that you provide. So every time a system is put together it's the first time, even if the system implementer has done it hundreds of times before. That aside, do you procure business critical components into your products without some degree of testing?

So testing is important, but like any service or production process, you don't leave it to the end. When you manufacture a product testing is important throughout the process, and you also need to have a specification to test against, and test procedures to follow.

Testing a business system is no different to testing your product or service, at every stage you need to check and test everything is on track:

1) **Design Qualification** – Maybe not what most people would consider "testing" of the application, but this is an important step in the process. The design specifications provided by your implementation partner are a statement expressing their understanding of how you want the system to operate. They will be configuring the application to meet this, and you need to review and "test" your understanding of the solution with them. Doing this right can save a lot of lost time later in the project, so your implementation partner should be more than willing to invest time at this stage with you.
2) **Conference Room Pilot/Prototyping** – This is the first time you will have your hands on the system, and your opportunity to test the design. You need to make sure that you test out your processes well, to identify any changes you need to make and ensure the system performs the way you need it to in order to support your business.
3) **Data Migration** – Often overlooked, testing to make sure the data being transferred into the system is going in correctly.

4) **Hardware Testing** – It is no good spending all your time and effort on the applications you are going to use if they do not fit in the box!
5) **Module Testing** – Do not forget that there are many individual components to your new system, each one needs to be tested and that it works with the bigger picture.
6) **User Acceptance testing** – This is going to be the first time that users other than those from the project team really get to see what is coming. The clue is in the title, users will need to run tests to then make the decision, 'does this do what we need and what we expect?'. You really need to think long and hard about the business scenarios that you will encounter and create test scripts to test these out. Your implementer can help you here, but it is your test not theirs remember that!
7) **Load/Performance testing** - What is the expected demand? Will my hardware and network infrastructure be able to cope?

I'm sure no one reading this will disagree with the need for testing, and no project would ever go live without it, but as a support company we so often find issues post go-live that were never found in testing, so why is it so rare to get perfect execution of testing? What needs to be considered to test a system?

Firstly, business scenarios. Many people will test that the system will cope with the day to day operation, the general situation, and overlook the unusual scenarios that occur more infrequently. Making sure you do a complete and full analysis of possible scenarios before designing your test scripts is critical.

Secondly, test script design. When creating test scripts for a product, you define how to carry out the test (method) and the expected results (specification) and testing a system is no different. Yet often reviewing test scripts for applications they often do not meet the same criteria. It is important that when you create test scripts it defines the approach, which should also include test data, and expected results; this then makes the process of testing the system easier.

Data Transformation

"And of course you'll transfer the data?" such simple words, but so dangerous in the assumption that all it takes is to move data from one system to another. Yet it's a fact that every project I have worked on that has gone late, availability of data has been a major contributing factor, and every project with major post go-live issues has been cause by poor data. Why is that?

For a start, the chances are that the data in one system will not directly map to another, processes may be different, user defined fields certainly will be. Secondly, the chances are that one of the major factors in moving to a new system is the quality of data in your legacy system. Over time "Data Accuracy Creep" occurs, as the accuracy of data is effected by continual use and updating, and unless some form of data quality system is being used to manage this, the chances are your data accuracy has led to issues in performance in the business that has caused you to review systems. Finally, there will be a lot of legacy data that you don't want to load into your pristine new system, as it is no longer used, old customer, supplier or product data.

So we can't just move the data, we have to put some thought into it. So this brings in the next thing to consider – it's your data! It's relevant to the running of your business, and you know what it does. All an implementation partner can do is facilitate the process, but you need to be heavily involved in the process.

So what are the key steps to data migration?:

1) **Identification** – Where is the data stored? Some data is held on paper, some in off-line stores such as spreadsheet's and some in online systems. Some data is even held between people's ears! Clear identification of all of these data sources is an important step to making sure nothing is missing.
2) **Policy Definition** – What data will you transfer? I mentioned above the need to leave unused older data behind, but how do you define that? Definition of the policy with regards to the data transferred decides what data you will/won't transfer and defines the size of the data migration project.
3) **Data Extract** – Extraction of data from your legacy systems, getting it into a format on which it can be reviewed and if necessary modified to suit the new system.
4) **Data Cleansing** – Once you have an extract of data, adjustments may have to be made to suit the new system, or even just to align data correctly. A typical example is in data addresses
5) **Data Loading** – The implementation partner will provide templates for you to define the structure of the data for loading into your new system, and there may well be much iteration. Accuracy and an eye for detail will be essential here.
6) **Archiving** – What do you do with the data you don't transfer? Legal requirements for financial records usually require retention for at least 7 years, and other statutory requirements may exist. It's not good practice to try and transfer this historical data into your new system – the time taken to transfer and reconcile records to get a current state would be much longer than making a decision on how you want to archive it and access.

It is likely that the process of extraction, cleaning and loading will have to be done multiple times, as you go through the project, starting with an initial extract for early prototyping, and continuing as you go through system testing and final transition, even if you get the process perfect on the first attempt it is likely to have 3 iterations. However, it's more likely that as you go through the process requirements change, mistakes get made and you need to go through the process again.

Conclusion

So are you guaranteed success if you concentrate on the three T's? Well no, there are other elements to consider, from the product you implement to the system design. However, once these have been set in the early stages of the project, the remainder of the project is about these elements. So to reduce risk of failure on your project, don't forget to put considerable effort into **T**raining, **T**esting and **T**ransforming your data.