Get The Most From Your Software Project

Sometimes business people feel a little lost when they embark on a software project. They don't know what to expect, nor how to contribute.

Follow these tips and you'll be well placed to make a solid start on your project, work effectively with your developer, and learn the rest as you go.

East Software is a small company located in Sydney, where the sun always shines and everything seems possible.

East Software
156/6-14 Oxford Street
Darlinghurst NSW 2010
Australia

http://www.EastSoftware.com.au
info@EastSoftware.com.au
Telephone +61 (2) 9361 3536

Do Nothing

The first thing you should do when you consider having a software system built is to ask yourself “Do I really need this?”

You can’t really answer this question at this very early stage because you don’t know yet what it will cost and what benefits it will bring. But it’s important to ask this question at several key stages.

Before you even start talking to a software developer, figure out exactly what problem you’re trying to solve. Estimate what the problem costs you in terms of time spent executing a manual process, the likelihood and cost of mistakes, the missed benefits of other things you could be doing instead, and so on. This is the cost of doing nothing.

After you’ve got a price quotation from one or more software developers, compare those prices with your cost of doing nothing. If it’s cheaper to do nothing, then cancel the project right now.

If you decide to proceed with the project then consider splitting it into several smaller phases rather than doing it all in one go. At the end of each phase, ask yourself whether it’s worthwhile continuing to the next phase.

If it’s cheaper to do nothing, then cancel the project right now.

Think First

When you engage a software developer the first thing they'll do is start asking lots of questions. This is called requirements gathering or requirements analysis, and its purpose is to work out what the software should do.

You might say "I want to track sales versus targets," and it probably seems straightforward: add up the sales in each month, divide by the target, and you get a percentage performance figure. If that really is all you need your software system to do, then the project will be finished in no time and everyone will be happy.

But the devil is in the details. Where will the sales figures come from? What format will they be in? Will you want to alter individual targets from time to time? How will you want the calculated results presented? Even for a small project there will be many questions.

Of course it's not necessary to work out every detail in advance. Indeed, some questions won't even arise until later in the project, as the developer learns more about your business and gets deeper into the nuts and bolts of the new system.

The important thing is that the developer should know enough about your business to make good progress, and to avoid taking wrong turns along the way.

Poorly understood requirements can have serious consequences for the project cost. If an important feature is identified only after the project is underway, the developer may have to redo significant design work. Generally, the later an error is discovered the more expensive it is to fix.

Research indicates that 68% of companies have poor requirements analysis capability[1]. In these companies, 50% of their projects have budget overruns of more than 60%. Clearly it's important to think carefully about your requirements before you commit to the project.

Generally, the later an error is discovered the more expensive it is to fix.

[1] Keith Ellis, "*The Impact of Business Requirements on the Success of Technology Projects*", IAG Consulting, 2008

Break The Project Into Phases

In all but the simplest of projects the requirements usually can be divided into three groups:

- must have
- would be useful
- icing on the cake

You may find it helpful to break the project into phases, implementing only the must-have requirements in the first phase. If there are many must-haves then pick just the top five or six and move the rest into the second phase. This way you can start realising the business benefits as soon as the most important features are ready, rather than waiting for everything to be finished.

After you've used the system for a short while you'll develop a better idea of which features are truly useful. You might find that some features you had planned for later phases don't seem so useful any more, and you can save time and money by dropping them. Conversely, you'll probably come up with new ideas as you use the new system. With the insights gained from practical use, as you discover what's possible and what's useful, you'll be better placed to plan the next phase.

Industry research shows that smaller projects tend to have smaller budget overruns[2]. Other studies have concluded that the majority of features that are built are never used, and that attempting to do too much is now the leading cause of project failure[3].

If there are many must-haves then pick just the top five or six.

[2] Keith Ellis, "*The Impact of Business Requirements on the Success of Technology Projects*", IAG Consulting, 2008

[3] Standish Group, *Chaos Report v3*, 2008

Involve Your IT Department Early

If you have an IT department then it's a good idea to talk to them early, while you're planning your project. If you don't have a dedicated in-house IT department then maybe you use an external consultancy firm, or you might have an IT-savvy employee who keeps your systems working. Whichever model you use, your project will proceed more smoothly if you involve your IT folks early.

Your IT people may be able to solve your business problem more easily in some other way, saving you the trouble and expense of having custom software built. Or they may see problems that you hadn't thought of. They might see opportunities to take advantage of existing IT systems and infrastructure, and can cooperate with your software developer to integrate your new system more seamlessly into your organisation.

And after the system is built you'll probably need your IT department's help to install it on your computers. Installation often requires administrator privileges, and connections to file servers and databases and so on. If your IT folks are already familiar with the project then they'll know what to do, so installation can go ahead without delay.

Most IT departments are happy to work with external software development companies. The earlier you introduce your developer to your IT folks, the better their relationship will be when it comes time to deploy your new system.

Expect Delays

Every project runs into problems from time to time. Industry research estimates that somewhere between 60% and 80% of projects take longer than planned, or exceed the budget[4].

Common reasons for project overruns include the original estimate being over-optimistic and unanticipated difficulties arising.

Some studies estimate the average overrun to be about 30%[5] or 40%, but overruns of 100% are not uncommon. So it seems reasonable to expect an overrun in your project, and if it costs only 50% more than expected then you're lucky.

Some developers will commit to a fixed price. This is good for you because it means you know exactly what it will cost. But in order to commit to the fixed price the developer will have factored this risk of overrun into the price.

It seems reasonable to expect an overrun in your project, and if it costs only 50% more than expected then you're lucky.

[4] K. Moløkken and M. Jørgensen, "*A Review of Surveys on Software Effort Estimation*", IEEE International Symposium on Empirical Software Engineering (ISESE 2003), pp223-230, 2003.

[5] Dexter Whitfield, "*Research Report No. 3*", European Services Strategy Unit, 2007

Do Your Own Testing

Software systems are always complex and always have bugs. Your developer will use his skill, and modern techniques, to eliminate most bugs before he hands the system over to you. But inevitably there will be bugs that just didn't surface in the development laboratory.

You'll almost certainly use the software in slightly different ways to how the developer intended, and your computer environment and other software systems will interact with your new system in ways that the developer couldn't anticipate.

So you'll need to do your own testing. This means that you'll need to set aside some time, and you'll need to plan for it. Ask the developer for advice on how to conduct your testing, and how to report bugs when you find them.

You'll almost certainly use the software in different ways to how the developer intended.

Train Your Users

The success of your project will be measured by the benefits it delivers to your business. And no matter how good the software is, if your users don't use it then it can't deliver any benefit.

You and your developer will have been thinking about the new system for a while, and so you'll know how it works, which buttons to push, what goes where, and so on. But when you introduce the system to your users it will be totally new to most of them, and things that seem obvious to you won't be obvious to them. Most users will try their best to figure it out, but if they can't get it to work then they'll give up and simply stop using it.

The most effective way to train users is to let them use the software in a direct face-to-face meeting, while you explain it. If possible, do not simply show them – they'll remember better if they do things themselves rather than just watching you do them.

If you have a large number of users, or if they're in distant locations, then a face-to-face meeting might not be feasible. Maybe you could train a small number of key people directly, and then let them go out and train others. Or maybe all you need is to send everyone an email describing the major features of the system.

The success of your project will be measured by the benefits it delivers to your business.

Establish A Support Facility

All software systems need some kind of support service. Support requests range from simple “How do I ...?” questions to reports of crashes and other serious problems. You can probably handle some of these issues yourself, but for others you’ll need help from your IT department or from the software developer.

It’s important that the people who use your system can get support when they need it, otherwise they’ll just stop using it. For some businesses this means 24/7 support from a helpdesk, while others are happy with a 9am-5pm Monday-Friday arrangement. Some need an instant response, and some can wait an hour or two. Think about what’s appropriate for your business.

Software development organisations generally aren’t set up to offer 24/7 helpdesk support, and anyway if you need that level of support then you probably have an IT department or external consultancy who can do it. But still your software developer should be available to offer advice and to handle critical issues within a reasonable time. Nobody understands the internal workings of the system better than the people who built it.

In some environments support is delivered within the framework of a Service Level Agreement (SLA). This defines who handles the different types of support request, and a timeframe within which to do so. For example, your own helpdesk might handle first-level issues such as simple questions about how to perform a task. Issues that need more consideration might be passed to a more technical team in your IT department, and if they can’t solve them then they may refer them to the software developer for deeper analysis and possible bug fixing.

However you handle support, the purpose is always the same: To keep your business users working with the minimum of disruption.

The purpose of support is to keep your business users working with minimum disruption.