# Comparing Building and IT Projects

**Neville Turbit**

## Overview

It is often hard to understand why IT projects can be so difficult. People in the Engineering/Construction project management area are shocked at how over time, and over budget IT projects can become. Perhaps it is time to compare a construction project and an IT project to understand why they are different.

## Example Projects

For our example, let us choose two projects. The construction project is to build a new house. The IT project is to build a customer management system to replace a manual record keeping function.

## Approach to a House Building Project

If we were to build ourselves a house, we would probably start with a number of things in place.

- A budget. We would know how much we intended to spend, or more likely, how much the bank would lend us.
- The type of house we want. Typically before starting the project we will have looked around at different display homes and know what we like and don't like. We will know for example that we want a brick two story house with a particular style or a timber single story house.
- The general parameters for the house. Perhaps we want three bedrooms and two bathrooms. We might want a single garage and a pool.
- A location. Normally we would have a piece of land to put the house on and an understanding of the geographical limitations. For example it may be on a hill and the surface could be sandstone.
- Building constraints. Most councils place limitations on what can, and cannot be built. We would know if two story houses are allowed in that street, or the ratio of house to land in the area.
- Availability of previous experience. Even if we have not built a home before, we have access to friends and family who have been down that path. There is also a considerable body of knowledge available from builders who have done the same sort of job thousands of times before.

## Approach to a Customer Information Project

A typical IT project starts with a different set of information:

- Budget unclear. The budget is not usually set before the project kicks off. Whereas a house is constrained by budget, an IT project typically focuses on 'what do we want' before 'how much will it cost'. There may be some horse

trading after the scoping but the usual point is to work out what you want first and worry about budget later.

- Requirements vague. With a house, we usually know the sort of house and the parameters but for an IT project, the sky is the limit. Perceptions of requirements for IT probably range from straight automation of the manual records, to an all singing, all dancing web based system that will predict when customers are about to order, and call them to take the order. The potential scope may vary by factors of 5 or 10.
- Environment unclear. Whilst a house is built on a bit of land, an application needs to run on a bit of LAN (excuse the pun). If the application scope is unknown, it might run on a PC or it might require expensive infrastructure.
- Building constraints not so constraining. Most organisations have some architectural compliance factors however they are usually not a serious constraint. There are many options open to you in building a new system. It might be Windows, or Linux based. It might use any number of databases. If skills are an issue, you can hire resources. All this leads to more uncertainty.
- Previous experience is not necessarily a good indicator. A builder can come to you and say "I built that house for $X. I can build the same one for you for $Y. The difference is because your site is less accessible, and the price for widgets has gone up 10% since I built the other house. The time will be the same. You can see and touch the house. It has a definitive number of bricks and tiles, and timber. You cannot do the same with a piece of software – at least not in the early days.

## The Initial Differences

So where are the differences? I suggest the following:

- A house is a highly repeatable process. Software is not.
- A house starts with a budget. Scope is adjusted to fit the budget. A software development has vague scope definition and budget comes later.
- Perceptions of the final deliverables are highly variable for software. They are more concrete for a house. For a house, it will look like that one over there.
- The environmental issues are probably well understood for a house. They are less well understood for a software project because we don't actually know what we will build so we cannot know the environmental issues.
- Software is more creatively sensitive in that all sorts of clever ideas can be incorporated that will impact time and cost. There may be clever ideas in houses but unless the budget is extremely flexible, most of the ideas will have to be accommodated within the budget.
- Constraints in building are well known and, in some cases, quite limiting in what can be accomplished. It is less so in software development.

## Decision Makers

In a software development, there may be a Sponsor, a Steering Committee, a key Business User, other Users, an IT department, an Architect etc. They are all involved in the decision making process. It is unlikely they will all sit down for the duration of

the project and make all the decisions as a team. More likely there will be scores of decisions made by one or two people.

It is inevitable that many of these decisions will have a degree of incompatibility. For example the decision to allow all users to update client credit details made by the Sales Manager may be incompatible with the decision about who can update certain information made by the Security Manager, and different again to the decision by the Accounts Manager that only Supervisors can change credit details. The point is that the complexity of decisions can easily lead to contradictory requirements.

I have actually seen a system built where at the screen end of the application, users could change credit details, but the database which had been configured to meet security requirements would not accept the changes. To make it worse, there was no warning that the changes had not been applied. Fortunately it came out in testing before the system went live.

In a house building project, there is typically one decision maker – the wife. Sorry guys but that's real life. The Project Manager or Builder is responsible for implementing the decisions. There is less diversified decision making.

Progress is easier to see with a house. You can walk around and see where the pipes for the hot water are going rather than have to inspect code to see what edits are being built into order entry screens.

In summary in a house building project, there is more central decision making, more visibility of progress so that errors are identified earlier, and less coordination of different points of view.

## How far to go

As the project progresses, on a building, it is much easier to see how much work is left. The outstanding work is highly visible. The rooms may need to be painted, landscaping done, and a fence finished. With software, it is difficult to see the work outstanding. Often completion of one area identifies unforeseen work in another area. We complete a screen only to find performance issues with the database.

With a house, it seems the further you go the more visible the outstanding work becomes. With software, the reverse is often true. The further you go the more obscure the remaining work becomes – up to a point at least.

Once again visibility is a major factor. You can see what is physically missing from a house. You cannot see physically what is missing from software. A screen may be there, but does it do everything you expect? Only testing will answer the question, and final testing cannot take place until the application is complete.

## Interdependence

This is one of the key differences. It leads on from the previous topic. In a house, there is an electrical system, a water system, a sewage system, etc. Each is relatively standalone, and interfaces are clearly defined. The hot water system may connect to the electrical system if it is electric hot water, but the interface is a simple power point. The guttering interfaces with the storm water system through a simple downpipe.

Consider an IT system. The interfaces can be extremely complex. There is no such thing as a standard interface in many cases. Linking to a database may seem relatively simple in IT interface terms but it can still take many days to get it right,

much less get it performing to expectations. When faced with interfaces to legacy systems, or just getting the links to your corporate email in place, there is rarely such a thing as plug and play.

On one project I heard about, several days were lost because a driver was upgraded by a supplier and the testing team were not aware of the new driver. They were using an old driver that did work before a service pack was applied to XP. It took several days to work out that the cause might not be the application but the service pack that had been rolled out to their operating system.

A home consists of a number of almost self contained components that have clearly defined ways of connecting together. An IT system consists of a number of integrated components that may or may not have a way to connect together. If they do have a way of integrating, there are likely to be a number of options in how it can happen, and various configurations within those options.

## Ability to Visualise

If I am describing a house, I can see it in my mind, and know what it will be like. It is hard to do the same in an IT system. The level of complexity is many orders of magnitude more in IT. A decision to make a field mandatory may have implications far beyond the IT system. For example, the decision to make phone numbers mandatory in an order entry system may cause all sorts of problems to cash sale staff. The customer just wants to pay their money and take their goods. Why do you need a phone number? This is another real life experience I have seen.

Whilst a wrong decision in a house may result in insufficient storage space, or inability to get the piano up to the second floor, the implications are usually more visible. If you have a piano and you want to put it on the second floor, you will likely think about how you will get it up there before you build the staircase. If you run out of storage space, you can always buy a cupboard and have it as a free standing storage unit.

This ability to visualise is another reason IT projects are more prone to cost and time overruns.

## Contractual

I recently was a member of a committee to develop standard project management terminology. It was interesting to understand the background to the need for an Australian standard. The key driver was that in the construction industry, interpretation of contracts was dependant on common meanings for words and phrases. It was effectively a legally driven need to ensure consistency in contracts. If a dispute arose, it should be clear that certain words mean certain things to everyone.

Construction contracts tend to be very specific – hence the need for consistent terminology – and quite detailed. There is often tens or hundreds of millions tied up in contracts and they have to be well thought out and involve a common, shared understanding.

Contrast this with IT contracts. It is usually well into a project that a statement of work is developed and a contract signed (if at all). It is unusual for the initial stage of requirements gathering to be covered by a contract, and many organisations would laugh at the idea. How can you quote on a piece of work when you don't know what

you don't know? IT starts with less of an understanding of what is required than does building and construction.

I did some work for a home builder many years ago and was amazed that from someone saying they wanted to purchase a particular style of house, within 2 weeks, they could develop a final costing. That included deciding options, modifications and additions to the standard plan, doing a site inspection to determine any environmental costs and producing a final plan.

As an aside, I was more amazed that they could get the purchasers into a room and in 4 hours determine all the furnishings, fittings and colours for the house. My partner would take around 4 years to do that.

Contrast that with an IT project. You cannot produce a bill of materials and labour estimate with any confidence in 2 weeks for a significant project. You will probably put a caveat on the estimate that it may be out by a factor of 100% or more. The IT components are not that easy to estimate.

## Summary

If you want to take two extremes in terms of achieving goals, at one end you could have building a house. At the other extreme you might have finding a cure for the common cold. People would expect that, with a little initial investigation, you could estimate with a high level of accuracy the cost of a house. It is not in the realms of reality to estimate how much to cure the common cold. There might be an early breakthrough or it may go on for decades. Whilst IT is not at the research end of the spectrum, it is somewhere around the middle.

The biggest risk with IT is that it is treated like building a house. Hopefully this white paper has illustrated where the differences lie. With an IT project, complexity will become evident all through the project. If you want to fix the budget and time early in the project, and are not prepared to adjust it, you have two options:

- As complexity becomes evident you abandon that functionality as it is too complex
- You trade off some existing functionality to cater for the complexity

Working harder will only take you so far. Working smarter can only improve productivity a certain amount. Sometimes the only solution to meet your objectives is to throw more money, resources and time at the project.

Finally, don't treat an IT project like building a house. The approach required is different for all the reasons above. If you set expectations that we can cater for what we know today but we also need to cater for what we don't know, you are more likely to deliver a satisfactory result.

Neville Turbit has had over 15 years experience as an IT consultant and almost an equal time working in Business. He is the principal of Project Perfect. Neville can be contacted at turbit@projectperfect.com.au

Project Perfect is a project management software consulting and training organisation based in Sydney Australia. Their focus is to provide creative yet pragmatic solutions to Project Management issues.

Project Perfect sell "Project Administrator" software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. They also created a technique for gathering requirements called "Method H"™, and sell software to support the technique. For more information on Project tools or Project Management visit www.projectperfect.com.au