# Managing Offshore Software Projects
# The Agile way

**Thavaranjan Thavendran**

## Abstract

Many companies in the western world have begun to move their software development activities offshore to countries like India, China and Sri Lanka. This has occurred in order to achieve cost savings, and thereby to increase competitiveness and optimization of their processes.

Proper management of offshore projects is vital to obtain the expected outcome. Unlike management of traditional domestic projects, management of offshore projects is quite different, due to various challenges such as culture, communication, multiple locations and time zones, visibility and processes. Therefore an effective and efficient management methodology is required for a better outcome.

This document is formulated with experience gained from recent offshore software project work based on agile methodology. It describes observations, which could be used by anyone considering exploring offshore development. It also provides information on what to look for when selecting an offshore vendor.

Agile methodology is a relatively new lightweight methodology and is looked upon by many companies as the way forward in software development. When compared to traditional methodologies, agile provides a practical approach to software development which takes the changing nature of the business environment into account. The main aspects of agile development are about increased speed of development, less time-to-market, quality, efficiency, and management of the impact of change.

## Introduction

Customer satisfaction is a great concern in all industries, to survive and be ahead of others in this current trend of globalization. Companies are competing at different levels to achieve a bigger market share by improving customer satisfaction, and strongly focusing on the deciding factors such as competitive pricing and quality, via the use of technology.

In order to be price competitive many companies from the western world are exploring the options of building their software products offshore to cut down on cost. This has made offshore software development a hot and attractive topic these days. Although, offshore projects have been quite attractive, many of them have failed to yield the expected benefits mainly due to poor or lack of project management rather than technical know-how.

Offshore project management is quite different from managing domestic projects, due to the many challenges that are posed when attempting to go offshore. Some of these are:

- Trust
- Internal resistance with the notion of losing jobs
- ROI (Return on Investment) not as expected
- Culture and language barrier
- Communication
- Multiple Time zone
- Multiple locations
- Lack of visibility

The sections below describe each of the above factors in detail.

## Understanding the Offshore Challenges

The main challenges faced when off shoring are the distance, culture and time, which create hardships for a smooth offshore operation. A good understanding of the above challenges will help to alleviate the risks completely or reduce them drastically.

### Culture and Language Barrier

Culture is a complex and an integral part of our lives, which we have learnt over the years and practice on a daily basis. It is the values, beliefs, behavior, morals, and law of a society, which makes up this culture. Culture has a significant impact on countries, companies, employee ethics and intra employee relationships, which could lead to several issues in offshore development.

It is easy to find technical experts in the outsourcing world, but the question is do they understand the cultural importance of your product? To what extent do they expect you to describe the requirements, even if they have experience in your industry?

Some of the cultural issues I have generally come across are:

- **Misunderstandings**

  Unlike in the west, developers from countries like India and Sri Lanka tend to be polite and hesitate to be forthright about problems. I have come across several situations where a task is assigned to a developer and a deadline is also set. When asked if they could meet that deadline, the answer would often be 'yes'. They will say 'yes' to almost anything, just to be polite, no matter how far fetched the issue is. This could be really frustrating for an offshore client, since when the module is delivered, it could be totally different from what they expected.

  Some would also say 'yes' when they don't understand. Most of them feel embarrassing to ask questions thinking whether the client would feel the question being asked is a silly question.

  There are cultural differences, even when it comes to body language. A simple example is in countries like India and Sri Lanka, people nod their heads top-down to mean 'yes' and left to right to mean 'No'. I have seen many of the clients (from

US, Europe, Australia) I have worked with getting confused with these body signs.

- **Hierarchies**

  From country to country the relationship people have with their superiors and subordinates differ. In countries like China, Japan and Russia rank and classes are very important, therefore there is a gap between managers and subordinates. In the US, Denmark and some other European countries these do not seem to matter much.

- **Trust and Responsibility**

  Trust is usually developed based on repeated face to face human interactions, but in an offshore development situation, most interactions happen via email, phone and online chats. Building trust in this situation takes time.

  Some companies have clearly understood the existence of cultural problems, and are taking measures in order to overcome them. They are organizing social events; educational programs on company background, and cross cultural training programs to bridge the cultural gap, thereby nurturing the organization to adapt to foreign cultures.

  One of the methods that has worked for me is to have my team present in the client's country for a certain period, mainly for understanding the requirements. Although this is the main goal, I am also of the belief that when people mingle with the natives of that country they will begin to understand the culture, work ethics, trust, etc. In fact this approach has worked, for me with several projects I have done.

**Communication**

Communication plays a vital role when dealing with offshore projects. English is mostly the common language for business dealings. Although English is spoken in many countries, English is not the same English everywhere, due to different pronunciations, and slang which makes understanding really difficult. To overcome this problem I have noticed many companies formalize communication with detail documentation and formal processes; some take this way too far. Although this works, I find this too costly, time consuming and it makes maintenance really difficult.

**Multiple Time Zones**

Imagine when you come into the office; your colleagues at your offshore center are packing up to go home; or imagine having offshore centers in several countries from different time zones, rather than in one. There is no solution to resolving this time issue, and the only way is to work with it. The impact of this issue should be born by both parties, and it should not be expected that it will be absorbed by one side only. Of course the situation is not as bad if the time difference is less. This is the case between European countries and Asian countries like Sri Lanka and India.

Several measures can be taken to reduce the impact caused by time zones; some companies have mutually agreed on an acceptable time with their offshore partners for online meetings and phone conferences. Others work according to the client's time during critical deliveries

Although time zones has its negative sides, some companies are found to make use of it as well, by round the clock development. This is achieved by working with the offshore partner on a shift basis.

### Lack of Visibility and Multiple Locations

Due to dispersed or multiple locations, keeping track of projects become a real challenge, in terms of keeping up-to-date with the progress and risk factors. Imagine a situation where the vendor, based on their evaluation mentions 80% of work is complete for a certain module. This, from a client's point of view, would have no meaning and there would be no means of verifying the above figure, due to the intangible nature of software. This often creates uneasiness among the clients.

## Need for a Methodology that Works

### Introduction to Agile Software Development Methods

Software development has been around for many years, and has evolved significantly, to keep pace with:

- The advancement of software development languages
- More sophisticated hardware
- Complicated market/business requirements and expectations.

In order to cope up with these changes, software development methodologies have been under constant scrutiny and research. They have striven to identify effective and efficient processes for delivering projects successfully within specified and tolerable resource limits.

Most of the current (traditional) software development methodologies have been in existence for many years, and have been difficult to put into practice in real life projects. According to Pekka Abrahamsson in their book on Agile software development methods, the traditional methodologies are not used in practice and that they are too mechanistic to be used in detail. With the shift towards offshore development and the requirement for faster development cycles, the traditional methodologies don't seem to deliver results to satisfy the modern day stakeholders.

From observation, the agile concept seems to work well for current day requirements and challenges, and is an effective and lightweight software development model. It is based on a collection of values, principles and practices. As described in the agile manifesto, the values of agile software development are expressed as;

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value
>
> - **Individuals and interactions** over processes and tools
> - **Working software** over comprehensive documentation
> - **Customer collaboration** over contract negotiation
> - **Responding to change** over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more."

The principles followed by the agile manifesto are;

- "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals.
- Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development.
- The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly."

Therefore in summary, the agile manifesto emphasizes people, interaction, working software, customer collaboration and change rather than processes, tools, contracts, and plans.

**What's Lagging in Traditional Methodologies?**

Over the years many methodologies have been created to tackle the challenges of the ever-changing software development process, however all these methodologies share a few common elements. According to Martin Fowler's article on "The New Methodology" (http://www.martinfowler.com/articles/newMethodology.html), he identifies the following common elements;

- **Based on engineering disciplines**

  In engineering disciplines, plenty of emphasis is put on getting an accurate design before construction. With an accurate design, the construction team can easily go ahead and start building. Design usually takes around 10 – 15% of the entire project time, and the rest goes for construction. Due to the difficulty in predicting designs, expensive and creative people are required at the design stage, where as for construction, low skilled people are quite adequate, but more time and cost is spent on construction due to 85 – 90% of the work involves construction.

  The same concept has been adopted in traditional software development methodologies as well, where emphasis is put into having an accurate design before coding starts. According to our observation, it is quite impossible to make an accurate and complete design, especially for business applications. Trying to make an accurate and complete design would only increase the design time and delay project deliveries. On top of this, the construction phase is much lesser than in engineering disciplines.

- **Predictive**

  With the above concept of separation of design from construction, traditional methodologies have a predictive nature, in the construction phase, due to the emphasis on an accurate and complete design. In practical terms as mentioned above it is impossible to make the design accurate. No matter how much time and expensive resources are put into the design phase, errors are uncovered during coding and testing.
- **Not geared for business requirement changes**

  Changes have been a norm when building business applications, particularly because of the volatile nature of the environment in which current businesses are operating. Due to this issue, it is almost impossible to finalize requirements upfront.
- **People are replaceable resources**

  In traditional methodologies people have been treated as replaceable components.

**Benefits from Agile software development methods**

There are numerous benefits that can be achieved from using agile development, which according to Forrester research are, improvements in time-to-benefits, overall quality and efficiency, team morale and relationship between IT and business staff, responsiveness to change.

| Benefit | Description |
|---|---|
| Time-to-benefits | Instead of focusing on improved time-to-market, teams now strive to deliver quantitative benefits to the organization much more quickly than in the past. |
| Overall quality and efficiency | Focus on testing is predominant. Test-driven development ensures that all requirements are sufficiently tested; short iterations allow for frequent checkpoints and user feedback to validate code against requirements. |
| Team morale | Agile teams are often self-organizing – they are told what is needed but not how they must deliver those requirements. They are required to follow any existing organizational standards guidelines and conventions, so what they build fits with the organizations other systems. Team member's collectively own the code and collaborate on each step of the development process |
| Improved relationship between IT and business staff | Business staff play a full time role on the team and are responsible for defining requirements and controlling the order in which requirements are delivered. They work alongside IT developers who estimate and produce the required software functions. Customers have a greater likelihood of getting what they want as they control the scope of each release. |

| Benefit | Description |
|---|---|
| Responsiveness to Change | Agile processes assume that all requirements cannot and will not be available at the start of the project. They therefore include activities to address a high rate of change and to respond to change in each short iteration. |
| Source: Forrester Research Inc. | |

## Agile for Offshore Software Development

Much as been talked about in earlier sections with regards to challenges faced when carrying out offshore projects. This section focuses on how agile methodologies can be adopted to overcome those challenges. As mentioned before changes are the norm in businesses, therefore a process that's flexible with changes and provides solutions to the above challenges, is a must if software development is to thrive in the current business world. Agile methodology is an answer to these issues and is a successful way of meeting expected business values.

We all know that it's a massive challenge to work in an offshore context where the team is distributed, however using agile development, makes the project more manageable. The following sections illustrate how agile method was used in different aspects of the project life cycle.

**Requirements analysis and documentation**

Traditionally, the requirements gathering were done as one big effort, at the beginning of the project. This often takes a lot of time and is still considered incomplete at the end. In the early stages of the project, it is very often impossible to identify all the requirements. Trying to dig into details will only be a waste of time, since some requirements are not known at that stage.

Given the above problem it is best to do the least analysis and documentation as possible at the beginning. A scope document is sufficient at this point, this should describe all the things that the system should do, and should be written in a format that the customer can understand. Also it should only provide enough details to make a reasonably low risk estimate. This method not only saves a lot of time, but also provides a starting point for the project. Detail documentations for each function can be made as the project continues.

It is important that developers meet face to face with the client to understand and document each functional requirement very clearly. Either the client should make a visit to the vendor's location or vice versa.

Functional documentation should be detailed enough to explain the functionality, but should avoid paragraphs and paragraphs of text. Paragraphs of text can lead to confusion and misunderstanding, instead the format shown in appendix 'A' which I used in the recent projects, worked really well for me. This method not only provides a nice template and reduces the time taken to write, but also makes life easier for the client to review.

Due to changing requirements, functional specifications will have to be changed often. When documentation has been written with paragraphs of text, it becomes a tedious job to keep it up-to-date, and therefore the ones prepared at the beginning of

the project become mostly obsolete at the end of the project. Updating the documents finally ends up almost as a new project.

With a format such as that in appendix 'A', I found it very easy and less time consuming to keep them updated during the coding cycle itself. We were able to get the developers themselves to keep these documents updated, which usually would be tough. Hard core developers don't like anything other than coding.

One of the other important section, that has to be included in the functional documentation is the user acceptance test plan. Having a test plan attached to your documentation provides the developer with a more sensible goal to make each test case to pass. This method is quite different from the traditional methods, but provides maximum productivity to take control of risks in managing offshore projects.

Apart from this, we also encouraged people to use video conferencing, phone, and on-line chats a lot. This not only reduces the problem with slang but also enhances team spirit among onshore and offshore team, as well as makes both teams see each others slang as a fun thing, and learn from it.

**Project Planning**

A project plan should exist at any given point in time. How detailed this plan would be, varies with the level of knowledge that we posses of the project. Most project plans change during the project cycle as we gain more understanding of the project. There are many tools out there to help you do project planning, but the most widely used one is the Microsoft Project. When you select a tool, make sure that your client too understands it.

The next step after selecting a tool would be to do the actual planning; which involves doing a task breakdown. Task breakdown should be done on a functional basis, where each task should have a meaningful output and should be testable by an end-user.

Tasks can be estimated using the three point estimation system. You can use the developers themselves to do this, since who else would be more qualified than those who will be directly involved in building the software. In order to do this estimation, I gather these 3 different figures for each task. The final estimate is calculated based on the following formula.

$$\text{Final Estimate} = \frac{\text{Optimistic Estimate} + \text{Pessimistic Estimate} + (3 * \text{Realistic Estimate})}{5}$$

*Optimistic estimate - minimum time a specified task will take to be completed (the best case scenario)*

*Realistic estimate - normal time the activity would take (the most likely estimate).*

*Pessimistic estimate - maximum time the activity will consume (the worst case estimate)*

The above method provides valuable information on the amount of risk associated with the task. This is determined by the difference between the realistic and the pessimistic figures. The higher the difference - the greater the risk.

As the project moves along, it is important the project plan is kept up-to-date; therefore it is important that we capture the actual time spent. These figures can be

easily obtained from the developers, by giving them a simple timesheet to fill on a daily basis. It is best if a project policy is made with regards to percentage completed, this should be communicated to the client as well. Telling a customer that a certain percentage, like 50%, is complete makes no sense for them.

Instead if a policy is made to denote that when a module is completed the percentage will be 60% (if its half complete then its 0%), and when it has been tested it will be 80% and when it's accepted by the client it will be 100%. In order to be successful at this, the project should be broken down into small units as possible, but each unit should be testable and measurable by an end-user.

**Time Boxes**

Managing deliveries is critical in any software development effort. Deliveries have a direct impact on expectations therefore how often releases are made, has to be determined very carefully. If the plan is to deliver once in every six months, then there is a very high risk that expectations will not align with what is delivered. Therefore the shorter the delivery period the better it is, but how short should it be, depends on your team, type of project and stakeholders.

Periods which are very short can cause stress for the team, overheads in release management and planning. For long term projects (such as one calendar year) one month periods are ideal. The selected period has to be used as fixed periods for delivery. This is referred to as time boxes, where delivery date or period is not changeable.

What can only be change is the content or the scope. Scope defined for a time box has to be delivered fully completed at the end of each time box as agreed. So what is delivered at the end of each time box are modules which are testable and can be clearly measured with respect to progress. If the contents have to be altered in a particular time box, then it has to be notified and agreed among the concerned parties.

Contents, or scope, for a particular time box should be fully achievable and realistic, within the set time frame of a time box. No activities should be incomplete at the end of the time box. The whole idea behind this is to have a clearly measurable and testable unit.

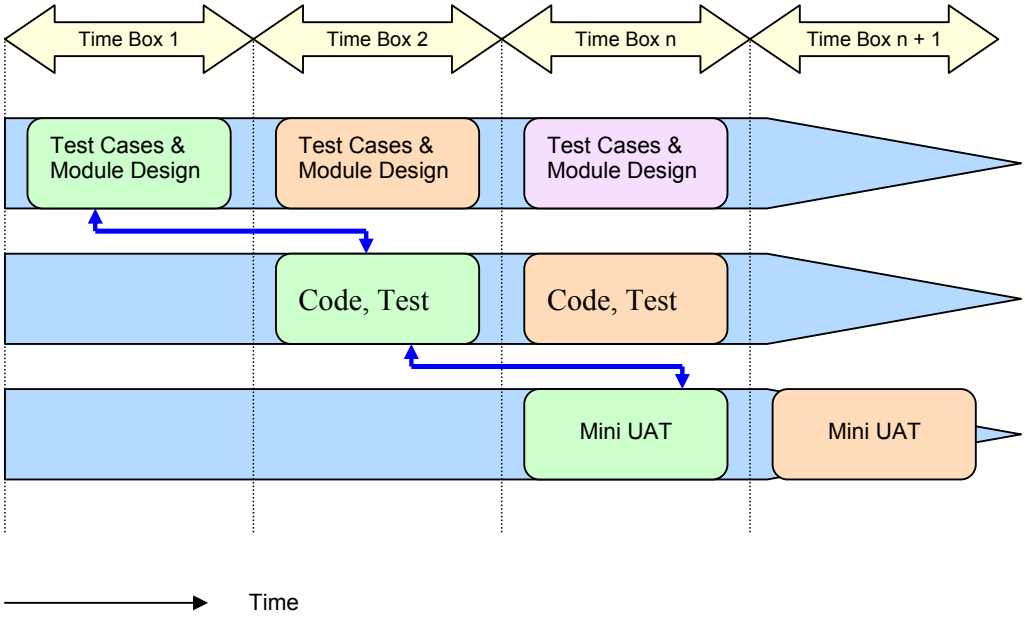
If for any instance, a particular module cannot be completed in one time box and has to span through multiple time boxes, then this module will have to be broken down into clearly testable sub modules. These can fit into a time box and can be completed in full, by end of that time box. Based on this a project plan has to be made and agreed among the project stakeholders.

The thinking pattern behind project planning using the time boxed approach is slightly different to the traditional approach. In order to get the maximum efficiency, divide the activities of a project into the following three;

- Requirements analysis (i.e. test cases and module specifications)
- Development (i.e. code, test and integrate)
- Mini user acceptance test on the delivery

Each of the above levels of tasks are carried out on successive time boxes. A requirements analysis carried out on Time Box 1, will have its requirements coded, tested and integrated in Time Box 2, and delivered to the client for acceptance testing at the end of Time Box 2. The client can do the mini user acceptance in Time box 3. Therefore this approach not only solves the problem of getting on with development

faster, but also provides the flexibility for changes to happen in, on-coming time boxes – see diagram below.




**Continuous Integration**

Integration is a real nightmare and a painful process even with a lot of care and dedication. Imagine several components being developed which have to be integrated almost at the end of the project. This would be a very difficult and time consuming process. On top of this, clients wouldn't have a meaningful software to test, and would always be in the dark, with regards to what they are ultimately going to get.

Continuous integration is an ideal approach to overcome the above situation. Each module is developed and integrated to the main software, before releasing it for testing. So this type of integration enables clients to see all components working together. It also allows them to come up with feedback early rather than having to visualize how it would look later when integrated.

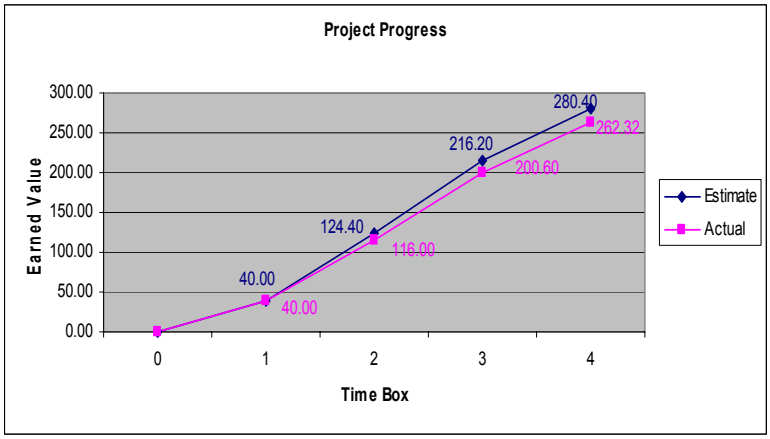
**Transparency and Progress Monitoring**

Stakeholders are always concerned about the progress of their projects, which makes project transparency a critical issue in managing offshore development.

Periodic progress reporting is necessary to keep the stakeholders aware of the project progress. Agile methods track progress by means of completed testable features as opposed to percentage of work completed. Percentage complete means components are neither testable nor tangible. This method of tracking prevents features from being under developed to meet time and budget constraints. Cutting scope usually means a surprise at the end when it is discovered that more work is needed to be done, when that solution is to be deployed.

The most meaningful way to represent progress is via a simple graph, based on actual work performed. Imagine showing a progress report with a graph as in the figure below to your client who is technically not competent in reading your technical

projects plans? It would easily make sense by comparing the estimated and the actual curve. The graph below clearly shows that the project was on schedule at the end of the first time box, but later shows deviation from the estimated performance. The deviation seems to increase at the end of each time box. A graph such as the one below can tell a lot of stories regarding how the project progressed from beginning to end.




The above graph is based on earned value versus the time boxes, and is drawn for a project with four time boxes (each time box is one week). Earned value is an objective measurement of how much work has been really accomplished.

The following example illustrates how a graph like this is drawn. Assume a project with the following tasks and time boxes;

| Task | Estimate (hours) | Actual (hours) | Percentage Completed | BCWP |
|---|---|---|---|---|
| **Time Box 1** (1 week) | **40** | | | **40** |
| Task 1 | 10 | 15 | 100% | 10 |
| Task 2 | 30 | 25 | 100% | 30 |
| **Time Box 2** (1 week) | **35** | | | **32** |
| Task 3 | 20 | 20 | 100% | 20 |
| Task 4 | 15 | | 60% | 12 |
| **Time Box 3** (1 week) | **45** | | | **36.25** |
| Task 5 | 25 | 32 | 100% | 25 |
| Task 6 | 20 | | 75% | 11.25 |

Budgeted cost of work performed (BCWP) is the actual amount of work completed at a given moment from the original scheduled work, and is calculated as follows;

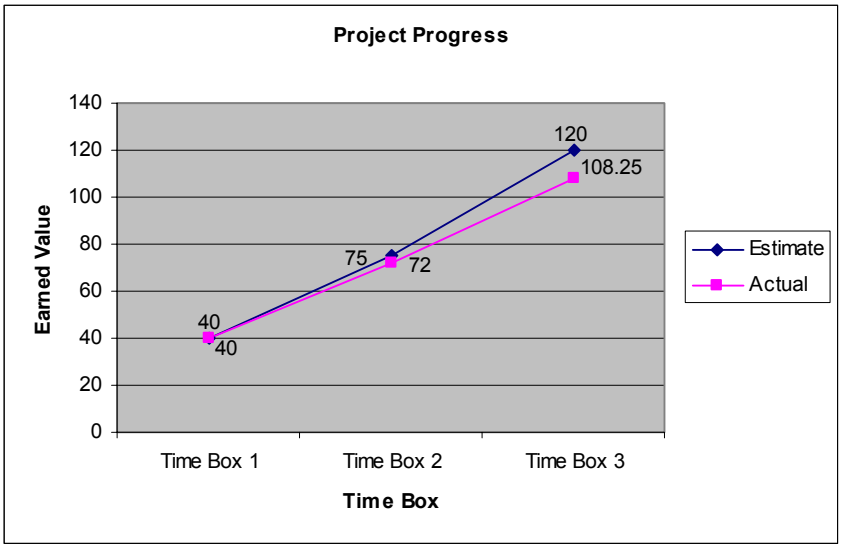
BCWP = Estimate * Percentage Completed

Based on the above data the project progress will appear as follows;

| Time Box | BCWS | BCWP |
|---|---|---|
| Time Box 1 | 40 | 40 |
| Time Box 2 | 75 | 72 |
| Time Box 3 | 120 | 108.25 |

BCWS – Budgeted cost of work scheduled.

Both BCWS and BCWP are accumulated




Some of the other details that I normally include in a progress/status report are

- Planned accomplishments in the current time box
- Evaluation of risks
- Activities requiring further information or approval from client
- Updated activity plan for the ongoing time box
- History of modules completed from prior time boxes

The above is information that I normally include, however it will vary with each client's expectations and needs.

**Active Client participation**

It is beneficial to get the user acceptance plan prepared along with the module specification (at the requirements analysis stage). In fact it's even better if the client himself prepared it, since this communicates the requirements even more clearly to the development team.

Secondly at the end of each time box, the client should carefully test each deliverable against the user acceptance plan made by them, and provide feedback to the software vendor.

**Document and Source Code Management**

Proper document and source code repositories and version control systems are very important. In fact the tool you select for the above purpose should allow for transparency and distributed development.

There are many web based open source tools such as Trac and Bugzilla to help you achieve this. Web based tools (such as "Trac", which is an open source Wiki), can be easily accessible by all parties irrespective of where they are located. When selecting a tool, careful attention has to be paid for the fact, that not all stakeholders in a project are technically competent. Ease of use is a critical factor when selecting a tool. Wikis are excellent tools to use in an offshore-distributed team environment, mainly because of their simplicity, ease of use, browser compatibility and simple to set up features. Some of the common features provided by most Wikis are

- Document management,
- Access to source code (when integrated with Source control software such as Sub version),
- Error/bug reporting (there are other open source tools out there, such as Bugzilla as well) and tracking.

## Part Two

This article will be continued in part two of the white paper.

## About the Author

Thavaranjan Thavendran has engaged in management of several offshore and onshore projects as a Project Manager. He has managed projects for many clients around the world, including Norway, Denmark, Australia, US, Jordan, Sri Lanka and Dubai. He has experience managing projects from a wide range of industries, with specialization in automation of stock exchanges and offshore project management. The types of projects he has managed are client-server application, web application, database applications and systems integration.

Thavaranjan has over 9 years experience in the IT field, and holds a BSc.(Hons.) in Information systems (from the Manchester Metropolitan University, UK) and MBA from the University of Lincoln, UK, and is also an ISO certified internal quality auditor. Author's Email: ranjanthavendran@yahoo.com

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