A Business Case for Improved Project Practices

Clarrus Consulting Group Inc. 7770 Elford Avenue Burnaby, BC Canada V3N 4B7 **T** +1 (604) 540-6718 www.clarrus.com

Abstract

Though most IT organizations could benefit from changes to their approach to building software, few are able to attempt change. Even fewer manage to sustain the effort to realize the benefits. One of the reasons for maintaining the status quo is that the true costs of poor practices are seldom quantified, let alone exposed to most decision makers.

This paper describes an approach for quantifying the costs associated with inefficiencies that most development groups consider normal. It shows that with very modest improvement goals, significant savings can be achieved. This business case affirms that most groups should start on the improvement path today.

The Business Case for Improved Project Practices

All software organizations suffer to some extent from inefficiencies and imprecision in their development processes. Indeed, the majority of organizations in the world (70% or more) would fall into the maturity rating of Level 1 according to the Software Engineering Institute's Capability Maturity Model for Software – an indicator of an 'ad hoc' software organization. Strong dependence on heroes to pull projects through, projects repeatedly being broad sided by major issues that prevent their timely completion, and poor delivered quality are typical of such organizations. If some of these symptoms describe your situation, you're not alone.

The degree to which this inefficiency and lack of precision is felt by an organization varies tremendously. The software industry is unique in that many companies have poor insight into their practices, but relatively high margins still allow them to remain profitable (sometimes very much so). In addition, few developers have been trained in reasonable software development practices, and few enjoy the opportunity to even learn about good development practices, as poor ones are so widespread. There is the common view that missed deadlines and poor quality are the nature of the software industry and there really "isn't much that can be done about it". Compounding this is the confusion between the intended approach to development (the process) and the activities that actually occur over the life of the project (the practice). Regardless of the published process for the organization, it is the practices that will determine whether a project succeeds.

Even if an organization is aware of its inefficiencies and understands that there are better ways to develop software, most never seem to find the time to actually embark on a software process improvement path – they just can't find the time to 'stop for gas'. Without an explicit understanding of the costs incurred, it is easy to rationalize the path of least resistance: continuing to do business in the old, inefficient but familiar way. Comments such as "There is too much work on our plate right now for a process improvement effort", "It doesn't contribute to our bottom line", "We tried it before and it didn't work for us", and "It's way too expensive" are common.

This paper suggests the decision to embark on a process improvement activity should be based on a traditional business case – it looks at the relative costs and benefits, and calculates a payback time and profitability index for the activity. Conservative estimates should be used as much as possible and industry averages can be cited to demonstrate that for most organizations there is clearly a very strong case for process improvement activities. You should start as soon as possible, as the payback time for relatively immature organizations (the majority, based on the CMM scale) is typically in the order of several months. It is our experience that this is indeed the case, and that companies can feel the benefit of process improvement activities immediately, as opposed to the 'next project cycle' that is commonly expected for initiatives such as the incorporation of CASE technology, or new development languages or methodologies (where there is a learning curve to be overcome before benefits are reaped). We have found that it's usually the appropriate application of well-understood practices, such as fundamental change management, that will provide early returns in a practical improvement initiative.

The Many Costs of Poor Process

There are several areas where costs incurred due to poor practices may not immediately be clear in an organization. Few companies have an adequate cost accounting or metrics system in place to determine the root causes, and these high costs associated with software development activities are considered to be the way things are. While new tools or methodologies are often prescribed to alleviate these cost areas, the potential return is marginal and the underlying practice deficiencies, such as poor project management or scope control, often remain untouched.

The clearest evidence of excess cost is the traditional project overrun. Industry reports indicate that software projects experience a cost overrun of 120% on average[1]. Normally, this cost can be attributed to applying more effort to tasks than was planned or performing new tasks that were not originally planned. In organizations where large amounts of overtime to get a project completed on time are not accounted for, this cost takes an additional, more human form that cannot easily be measured, namely low morale and poor staff retention. In addition, many projects experience schedule overrun, which generates a cost of a different form – an opportunity cost of lost sales as the product is delayed to market.

Core Steps for an Improvement Business Case

Here's a brief summary of the steps you could take in using a business case approach for driving improvement in your organization:

1. **Identify the need to get the message across** – the fact that current practices are not necessarily optimal. Derive the case in a language that is understood by the decision makers – this is usually a monetary one.
2. **Identify the symptoms of poor practice** – what is happening that is less than optimal. Common symptoms are schedule slip, scope creep, and poor quality.
3. **Quantify the costs of poor practice** – back to that common language, dollars, identify the real pain the company is in by quantifying the symptoms identified. This can normally be quantified into opportunity and inefficiency costs.
4. **Identify the potential gains** – based on reasonable assumptions, where could the company be? Don't expect perfection as a result, and be sure to include the additional cost of refined practices.
5. **Perform a reality check** - to ensure that the information in the model makes sense. Build measures into the improvement project that will allow you to validate your model for the next iteration.

In an ongoing improvement program, you would distill the symptoms to the root causes – those current practices that contribute to the most troublesome symptoms. A focused approach where you improve the few practices that are generating the greatest pain will usually provide quick, clear returns for your efforts.

For organizations that share software development resources across several projects, or that have a cross-dependency between different software products, the schedule overrun costs are compounded. As one project is delayed, those resources that were expected to be free to work on other projects are held back, thus delaying these secondary and tertiary projects as well.

Many software companies will consider a schedule milestone to be a firm date, and will drive the project to completion 'on time' by dropping functionality as necessary. For those companies that have agreed to this functionality up front with their client, there is a very real cost of failure to meet expectations with delivered functionality. This can be realized as costs through reduced customer satisfaction, or through additional effort to deliver the complete set of functionality (similar to the project overruns described above).

Most organizations also have ongoing rework for their products, sometimes disguised as "maintenance effort". Industry averages show almost 30% of all development effort[2] is rework to accommodate poor quality or functionality change.

---

[1] Chaos Report, The Standish Group, 1995

[2] Howard Rubin, 1997

An Overview of the Business Case

The model suggested here attempts to quantify these many costs, and the benefit to be gained through the reduction of these costs. It's based on some simple assumptions and basic data regarding the organization's size, sales figures, and 'symptoms of trouble'. Two cases are considered. As in traditional business cases, the 'status quo' approach of making no changes to the business is considered as the baseline. The second case quantifies the costs and benefits associated with reasonable process improvement activities.

Note that the numbers provided in a model such as this would not be intended to map directly to numbers that you would see in your corporate financials. There are a number of additional costs in an organization than just the burdened labor rate for an R&D group, for example, and you are unlikely to see the net revenue indicated by such a model in your bank accounts. These numbers do, however, quantify the relative costs that can be attributed to different categories. Remember that University of Wisconsin-Madison professor George Box once noted, "All models are wrong, some models are useful". The goal here is not to predict next year's balance sheet, but to quantify an argument for improving your practices.

Initial data for the model should include some quantification of the current state of the organization, to form the basis for the business case. Given simple metrics such as the size of the development group and an average burn rate, along with corporate sales figures, you can easily calculate the overall current costs and revenue to use as a baseline.

Basic assumptions can be made regarding the internal cost of the process improvement activities and the expected value of these improvement activities. From these figures, a quantification of the costs and potential benefits of process improvement activities can be modeled, and each case can be considered from the perspective of a Profitability Index and Payback Time. For those interested in playing with the inputs to help understand the sensitivity of the model, contact us for a softcopy version of this model: info@clarrus.com

The Symptoms of Trouble

The following symptoms of trouble are useful as drivers for determining the capacity for improvement. While organizations face many other symptoms of trouble in software development projects, we have found that the following are usually the ones that cause the greatest grief. In the table below, industry averages are provided as a guideline to illustrate the state of the industry in general. If it is not clear where your organization stands (which in itself is a good indication that process improvement activities may be warranted) you may wish to view these industry averages as a starting point. We have found that symptoms that are a fraction of these industry averages will provide a compelling argument for improvement.

Symptoms of Trouble	Industry Averages
Average project slip	120%
Functionality dropped to meet deadlines	20%
Customer satisfaction	80%
Average effort devoted to rework	30%

Average project slip is the average duration beyond the originally planned duration for the project. The industry average of 120% indicates that the average project actually takes 220% of the originally planned duration to complete.

Rather than allowing projects to slip, many organizations choose to 'time-box' their software projects, and come to closure by reducing functionality to meet the desired end date. The functionality dropped to meet deadlines is this percentage, as a function of the original scope that was specified.

We consider customer satisfaction as a percentage of customers that are happy with the product, and hence do not request a refund or somehow attempt to recover their costs. Remember that for each customer that provides negative feedback, there are many more that don't bother to let you know their feelings. In addition, there is a cost of negative 'word-of-mouth' marketing that is inevitable, and not considered in this model.

Rework effort is the effort that is outside the primary, single path through the development cycle – the ideal goal should be to elicit requirements once, design once, code once, test once, and ship once. Any effort towards items previously assumed to be completed should be considered rework.

Pros and Cons of an Externally-Driven Improvement Activity

There are several compelling reasons to take advantage of an external agency to drive a process improvement effort for most organizations. These fall into two main categories: experience and objectivity. You may have the resources in your organization that have the required experience as outlined below and they may be able to devote the time to driving this sort of activity. If so, then the reasons based on experience provided below become moot and there is no real motivation to enlist the services of an external agency. Few organizations have this luxury.

Experience comes in several flavours. For a process improvement activity to be successful, it must be driven by a strong understanding of the breadth of best industry practices, methodologies, and process standards. As no single standard is a good fit for all, and those standards that exist require fine tuning for each organization, depth of knowledge is key in understanding the process areas with the highest return on investment for a given organization.

In addition, experience in rolling out process refinements backed by a proven methodology is a definite asset. Knowing what works and what doesn't work in the practical application of improvements is critical to the success of the effort. Using an organization with a proven track record in this activity reduces a great deal of the risk associated with the improvement project. For organizations that attempt a 'home-grown' approach to process improvement, this risk is realized in the inefficiencies of sub-optimal approaches, including repeated false starts and long learning curves.

Objective assessment of where the organization currently stands (a critical first step in any improvement activity) is often difficult for people within the group. Indeed, if the goal is formal certification for standards such as ISO or a rating at a specified CMM Maturity level, an external, objective opinion is mandatory.

An external resource is not at risk of having priorities changed mid-stream and moved onto another activity, such as the completion of a project in trouble. As any internal resource used for driving this type of activity must, as described above, have strong depth of knowledge, this risk can be very high for groups with significant process issues – as a result, the improvement activity would be delayed or cancelled.

On the negative side, using an external agency can make it relatively easy to defer the improvement activities as issues arise in other parts of the organization – there is no internal clout to ensure that the improvement project will stay on track. Ideally, the expertise and objectivity of an external agency should be combined with real buy-in to the improvement project high enough in the management levels. This will ensure that there will be follow-through, even when the temptations are there to fall back into your old, comfortable, inefficient ways of developing software.

The Cost of Poor Quality

These symptoms can be translated, given the original corporate metrics described above, into a Cost of Poor Quality for the organization, in terms of the costs of inefficiency (schedule slip, functionality drop, and rework) and the opportunity costs (through loss of customers as well as costs associated with delay to market).

Note that while the cost of rework is independent from the costs associated with schedule slips or functionality drops, it is a useful approach to only consider the largest of the individual costs, rather than to sum them together. This makes the model much more conservative and defendable, and you are likely to find that the resulting numbers are still very compelling. From the costs of inefficiency typically derived in a model like this, it is quite often the case that 50% or more of the overall development labor costs can be associated with inefficiencies in the process!

Even if you deliver all functionality on time, it is a rare project that has not had inefficiencies due to rework. While rework can be a major contributor to the costs from schedule or functionality slip, poor planning and estimation of scope (including the failure to take into account expected rework effort) is probably a more accurate description of the cause.

Add to these the opportunity costs, which, for product-based organizations, can be significantly larger than the inefficiency costs. There is the cost is due to poor customer satisfaction, which may be a function of quality, service, or some other factor. This differs from the second cost, which is the loss of new sales due to the lack of product availability. This opportunity cost can be realized even if the product ships “on time” if the expected functionality has been cut to meet the target date.

Reasonable Improvement Expectations

We turn now to what could result from an externally driven improvement approach, where internal resources are typically used for the bulk of the effort, and an external resource is brought in to guide the project. Refer to the accompanying side bar for our impression of the value of an externally driven improvement effort.

With this approach, we have found improvement in terms of the reduction of the pains specified above to easily be 25-50% or more for initial efforts. This would be offset by an internal investment of 5-10% of the development force’s effort, and some smaller external investment as well. Again, the assumption is that you will not gain perfection in a single iteration of improvement activities, but that this should be part of an ongoing improvement program.

These reduced pains will typically result in reduced costs in inefficiency that will more than make up for the investment in improvement, both internal and external investments. We usually see the overall development costs change slightly as the reduced inefficiencies (a reduction of costs) and the costs of the process improvement activities are taken into account. Note that the external process improvement cost is typically a small fraction of the internal investment based on our experience with a large number of software development organizations.

We have found that while the opportunity cost improvements can appear extreme for many organizations, the numbers can lead to interesting dialog between the development group and the sales and marketing force, centering around questions such as “If the product was this much better, or had this much functionality, do you think you would be able to sell this much more?” Our experience is that you will usually get an affirmative response.

The results we’ve seen for typical organizations using a model such as this show a significant revenue increase and corresponding increase in productivity that suggests a ROI of 200% or more, and a Payback Time that is within a single project cycle for most organizations (well under a year).

The model should assume the costs and benefits over a specific time period, usually between 6 months to 1 year. In reality, the costs of the improvement effort are loaded early in the cycle, where stronger planning practices are introduced and training sessions are held. The benefits, on the other hand, will tend to gradually increase to a new steady state for the organization, and more importantly, continue beyond the period in question. In some cases, significant benefits can be experienced immediately.

But we're not a product-driven company…

Some of the most compelling arguments in this business case come from the multiplying effect of revenue generation associated with product driven companies. Where is the value in this argument for traditional IT shops that provide software and services to internal clients?

All of the internal development efficiencies hold true, and this in itself should be a strong enough argument for many groups. In addition to this, there are a number of client benefits, even if they cannot easily be translated to monetary gain:

- Predictability – the organization is likely to be more reasonable in setting and meeting expectations, with fewer significant issues arising from a schedule and budget perspective. Note, however, that it is often the case that these more reasonable expectations will result in a predicted schedule or cost that is greater than those predicted before the improvements – a prediction that is closer to reality. Though it will be a tougher pill to swallow, it will allow for more realistic assessment of options.
- Quality – the quality of the product can be expected to increase, but there is also the quality of service provided. You can expect improvements in the overall satisfaction of dealing with the organization.
- Throughput – as a direct result of the improvement in productivity (the reduction of inefficiencies), you can expect to get more output from the development group, or the group may be able to satisfy the needs of a larger client base.

Eventually there is an external client, even if it is not the direct recipient of the product of the software development group. The product this client receives may be a service that the organization provides, and there is a strong likelihood that this service can be dramatically improved through refinement of the IT development practices.

Reality Check

It's been said that the difference between theory and reality is minimal in theory, and huge in reality – so how do the results of this model reflect what we have experienced in the real world?

While we have not measured these values in most of our improvement projects (as one of the traits of immature organizations is the lack of baseline data), anecdotal evidence fits well with the model described here. Development teams experience almost immediate benefit from improved practices and the quality and predictability of the product increases significantly (much more than the model data here would suggest). We have measured greater than 100% improvement in raw productivity on several projects.

Sales variability is such for most organizations that it is usually difficult to attribute sales increases to specific changes within the development organization, but opportunity costs and customer returns can be significant motivators for an organization.

Key points

Even with modest assumptions in a model like this, it usually shows significant payback in a short period. The most significant cost justification for improvement work does not come from cost savings, as the development group is likely to continue working at its present rate.

The primary savings come from reduction in opportunity costs which translates into higher sales and higher customer satisfaction, not to mention better staff morale and lower turn over. This is reflected in reality, though not as clearly measurable, in most organizations.

In addition, the non-financial improvements can be just as compelling. The sanity introduced into a workplace where projects can be repeatably predicted to complete in a reasonable timeframe can add a great deal of credibility to an organization. Quality is another key target of any improvement initiative, and we have experienced significant results in this area as well.

Remember that this is a model, and that your mileage may vary.

All attempts should been made to ensure that there are no ‘magic numbers’ in the model, and to be generous with the costs, while critical of the benefits. We have found that initial initiatives for process-immature organizations can be spectacularly better than represented in this model.

Once this initial improvement has been made, the ongoing incremental improvements can still be of the orders shown by the model. In addition, with an initial successful foray in improvement of development practices, the interest and momentum generated will be such that the need to sell the business case for an ongoing effort will be eliminated.

About Jim Brosseau

Jim has been in the software industry since 1980, in a variety of roles and responsibilities. He has worked in the QA role, and has acted as team lead, project manager, and director. Jim has wide experience project management, software estimation, quality assurance and testing, peer reviews, and requirements management. He has provided training and mentoring in all these areas, both publicly and onsite, with clients on three continents.

He has published numerous technical articles, and has presented at major conferences and local professional associations. His first book, Software Teamwork: Taking Ownership for Success, was published in 2007 by Addison-Wesley Professional. Jim lives with his wife and two children in Vancouver.

About Clarrus

It is often the same class of challenges that are the root cause of most business inefficiencies. A key aspect of these challenges is that they manifest themselves differently in every organization. That's where we come in.

Clarrus is a down-to-earth, value-driven company with a focus on improving project teams and the projects they work on, across all industries.

Clarrus is unique in that it understands the interplay of human dynamics and project mechanics. Our consulting services and workshops bring a practical and proven approach to increasing the effectiveness and satisfaction of your project teams and delivering results.

Clarrus believes in the principle of teaching people to fish. Our approach nets its richest results in complex environments where multi-disciplinary teams bring diverse perspectives to the table. But, we also help small teams thrive, too. In a nutshell:

- We harness the best of best project practices and apply only what's needed for tangible short- and long-term results.
- We create an environment of trust so our sometimes tough questions can open the doors to peak performance.
- We provide inventive and practical ways for your teams to approach challenges.
- We secure the engagement of your project teams by focusing on issues where the impact is greatest.

Contact us today.

- On the web at http://www.clarrus.com,
- By e-mail at info@clarrus.com, or
- By phone at +1 (604) 540-6718.

January 1, 2010 - Version 1.0