A Lead Role in Software Success

Successful software projects have four characteristics in common: they deliver **on time**; they meet **budget constraints**; they fulfill their **business objectives**; and they achieve the **desired quality**. Schedule and budget constraints are usually known in advance. These two goals are straightforward — it's easy to determine whether they've been met. Business and quality objectives, on the other hand, are rarely well defined. Without clear, measureable standards and benchmarks, it can be difficult to determine whether or not the software meets them.

So what can executives do to promote successful software initiatives without becoming mired in the details of directly managing them? The following four actions — when properly implemented and adhered to — will improve the predictability of your software development and enhancement projects while reducing unpleasant surprises and budget overruns.

1. Get a good chief technology officer.

This is almost self-evident. He or she is the one who should be the focal point for software development; not you.

2. Determine your capabilities and plan within them.

There is an old myth about King Canute who was so impressed with his own abilities that he believed that everybody and everything would do as he commanded. He had his throne set on the seashore at low tide and commanded the sea to stay out. It didn't work. Organizations that do not understand their capabilities are very much like the good king: they base their decisions on their desires. Often, this backfires.

So, how do you determine what these capabilities are? The answer is simple yet it forms the crux of how to promote successful software projects: **measure them, collect and analyze the data, and base future decision-making on the results.** Organizations have patterns in their software development; how they staff projects, whether they emphasize cost or schedule, and the overall quality of their work. These patterns exert a strong inertial pull and define an organization's capability to produce software. They can be changed over time, but only if the underlying reasons for their existence are altered. They cannot be changed by executive fiat.

3. Measure for Success.

Software projects leave artifacts that can be measured and used to determine your current capabilities and plan for ways to improve them. What you as a senior executive must do is assure that this information is collected and used as the basis for decision-making. So, what is this forensic evidence that every software project creates? Every software project:

- Takes place over time. This is **project schedule.**
- Requires staff that needs to be paid. These are **effort** and **cost.**
- Creates an end product. The measurement of this is **project size.**
- Experiences glitches both in development and the end project. These are **defects.**

This seems like a simple list, and it is. However, analysis of this data from your organization's projects provides the basis for determining minimum development time, optimal staffing, and schedule. This data allows you to benchmark your capabilities against other organizations. It will allow you to distinguish project plans that are

feasible from those which are unrealistic. Accurate measurement does not just happen because it is a good idea. Three things are required. Two of them are your responsibility.

- A small measurement group within your company that collects, stores, and analyzes the data
- Your commitment to establish and enforce processes to collect and analyze project data
- Your commitment to base project decisions that involve staffing, cost, and schedule on what the data tell you; not on wishful thinking

4. Establish a Measurement Group.

It is a dirty secret, but true nonetheless: many software measurement programs fail. For yours to succeed, you must identify potential pitfalls and avoid or ameliorate them. Here are some of the more common reasons for failure and suggestions for addressing them.

> **Get support from senior management.** Without management commitment, a measurement program is doomed from the outset. If management does not adequately fund the effort, establish the necessary processes and procedures, provide justification to those tasked to follow them, train employees, and establish and follow through on consequences if they are not adhered to, the program will thrash around for a while and ultimately end up being cancelled as an unnecessary expense that provided little value.

> **Don’t underestimate resistance.** Resistance is very much like the tide in the King Canute analogy: people and organizations are accustomed to the way they do things and feel threatened by change. Most resistance will not be overt; but it is there, nonetheless, and must be dealt with. A common source of resistance is that employees believe that a measurement program will be used to evaluate them. When they believe this, project data will be manipulated to paint a good picture of their performance, which, of course, invalidates the data. There is a compelling reason not to use software project data for performance evaluation: it doesn’t provide accurate insight into individual performance and any decisions regarding performance based on it are inaccurate and invalid. What it does is provide a basis for evaluating project performance and whether the processes and procedures in place are helping or harming your organization. Performance management is a valid organizational concern: one that should always be separated from a software measurement program.

> **Too much, too soon.** It is leadership’s role to address the two concerns listed above. When instituting a software measurement program, start by making it as unobtrusive as possible. When leaders see the possibilities inherent in software measurement for improvement and profitability there is often a desire to jump in headfirst. The software measurement team, desiring to demonstrate its value, may be all too willing to oblige. But transforming an organization from one that bases decisions on desires and hunches into one that grounds them in proven capabilities requires time and effort. Here are some ways to avoid overwhelming your employees.

> **Leverage processes you already have in place.** Your current time tracking tools are an excellent place to obtain cost and effort data without requiring additional work from employees. They may also provide information on when projects begin and end (schedule). Some work may be necessary to massage existing data into a form the metrics group can use, but this is a whole lot simpler than implementing a completely new process. Likewise, if you have a defect tracking system and/or a ticket system for recording problems and change requests, take advantage of it. Most of the information required for data driven management already exists; it just hasn’t been organized, analyzed, or used.

> **Start small and build on success**. With cost/effort, project duration, defects, and project size an amazing amount of project analysis can be performed. Rather than attempting to do everything at once, select a particular pain point or problem of your organization and focus on it. You will learn a lot from this that will help make solving the next issue proceed much more smoothly and help resolve a particularly important issue while not

overwhelming your organization in the process.

A Word about Outsourcing

Most large enterprises now contract with third parties to do some or all of their software development and maintenance. Often, the actual work is done in other countries. The reason for this is very simple: software is labor intensive and labor rates for IT personnel in developing countries are frequently a fraction of what they are in the United States and Western Europe. Even after taking into account the inefficiencies the practice presents, business has overwhelmingly decided that it is economically worthwhile.

Outsourcing presents difficulties that are not present when the work is done by an enterprise's own employees. First, the IT staff are not employees and are not directly managed by the enterprise so there is a loss of direct control over IT development and maintenance. Second, another party, the vendor of the IT services, is now involved and the level of support and how that support is provided are contractual issues.

Organizations that outsource software services need the same measures from vendors that they would need if the work were being done internally. Why?

- To evaluate the vendor's performance. Are productivity and quality improving over the life of the contract, or not?
- To validate vendor estimates. Are the vendor's cost and schedule estimates accurate or not? Are the estimates improving over time?
- For cost benefit analysis. Are you getting what you paid for? Would another vendor do a better job or cost less? Would it be better to do the work in-house?

The difference between collecting basic measures needed to monitor vendor performance and not doing so can amount to millions of dollars. As an example, several years ago QSM was brought in to benchmark a vendor's performance. Analysis indicated that the program was severely overstaffed and that the same level of work could be done with half the staff with minimal impact on schedule or level of support. These recommendations were implemented at a savings of around $10 million over the three-year life of the contract with no negative side effects.

Nobody likes to be measured, and companies to whom IT support and development are outsourced are especially sensitive to it. If they are not required to supply the raw metrics required to evaluate their performance, these measures will not be provided. Therefore, it is extremely important that there be a contractual obligation that defines what will be collected and its frequency. Careful thought should be dedicated to oversight during contract negotiation, and a measurement expert with experience in contract negotiation should be involved in defining the measures for service level agreements.