"How can I establish a software vendor management system?"

July 2016

Scope of Report

This month's report will focus on two key areas of vendor management. The first is vendor price evaluation which involves projecting the expected price for delivery on the requirements. The second is vendor governance. This is the process of monitoring and measuring vendor output through the use of service level measures.

Vendor Price Evaluation

"Vendor Price Evaluation" seeks to enable pricing based on an industry standard unit of measure for functionality that puts the buyer and seller on an even playing field for pricing, bid evaluation and negotiation.

Organizations leverage third party vendors for the development of many of their software initiatives. As such, they are continuously evaluating competing bids and looking for the best value proposition.

Being able to properly size and estimate the work effort is critical to evaluating the incoming vendor bids. Furthermore, an internally developed estimate provides a stronger position for negotiating terms and conditions. The effective delivery of an outsourced project is in part dependent on an open and transparent relationship with the vendor. A collaborative estimating effort provides for greater transparency, an understanding of potential risks, and a collective accountability for the outcomes.

To better control the process, an economic metric is recommended to provide the ability to perform true value analysis. This metric is based on historical vendor spending over a diverse sampling of outsourced projects, thus creating an experiential cost-per-unit "baseline". Knowing the cost-per-unit price gives you leverage in negotiation. Instead of using hours billed as a fixed price measurement, you know the functional value of deliverables which allows billing on a per unit delivered basis.

To achieve this, we recommend the use of function points as a measure of the functional size of the project. Function Points (FPs) provide an accurate, consistent measure of the functionality delivered to the end user, independent of technology, with the ability to execute the measurement at any stage of the project, beginning at completion of requirements. Abundant function point-based industry benchmark data is available for comparison.

By comparing historical cost-per-FP to industry benchmark data, organizations can quickly determine whether or not they have been over- (or under-) spending. Under-spending may not seem like a problem but under-bidding by vendors is an established tactic to win business that may not represent a sustainable price. If forced to sustain an unrealistic low price, vendors may respond by populating project teams with progressively cheaper (and weaker) staff to the point where quality drops and/or delivery dates are not met. At this point, having great lawyers to enforce the original contract doesn't help much.

Implementing this approach provides an organizational methodology for bid evaluation and a metric for determination of future financial performance.

270 West Lancaster Ave., Liberty Square B-2, Malvern, PA 19355 • 1-610-644-2856
info@softwarevalue.com • www.softwarevalue.com

Vendor Governance

The key to a successful vendor governance program is an effective set of Service Level Agreements (SLAs) backed up with historical or industry benchmarked data and agreement with the vendor on the SLAs.

The measures, data collection, and reporting will depend on the SLAs and/or the specific contract requirements with the software vendor. Contracts may be based strictly on cost-per-FP or they may be based on the achievement of productivity or quality measures. A combination can also be used with cost-per-FP as the billable component and productivity and quality levels used for incentives and/or penalties.

There are a number of key metrics that must be recorded from which we can derive other measures. The key metrics commonly used are: Size, Duration, Effort, Staff, Defects, Cost & Computer resources. For each key metric, a decision must be made as to the most appropriate unit of measurement. See the appendix for a list of key metrics and associated descriptions.

The service level measures must be defined in line with business needs and, as each business is different, the SLAs will be different for each business. The SLAs may be the typical quality, cost, productivity SLAs or more focused operational requirements like performance, maintainability, reliability or security needs within the business. All SLAs should be based on either benchmarked or historical data and agreed with the vendor. Most SLAs are a derivative of the base metrics and thus quantifiable.

One output measure to consider adding is business value; fundamentally, the reason we are developing any change should be to add business value. Typically, business value isn't an SLA but it can add real focus on why the work is being undertaken and so we are now recommending it. The business value metric can be particularly helpful in the client-vendor relationship because it helps to align the business priorities of the client and the vendor (or to highlight any differences!).

The key is to define the measures and the data components of the measures prior to the start of the contract to avoid disputes during the contract period.

Reporting

Measurement reports for vendor management are typically provided during the due diligence phase of vendor selection and during the execution of the contract. During due diligence, the reports should provide the vendor with the client expectations regarding the chosen measures (e.g. Cost-per-FP, hours-per-FP, etc.). During the life of the contract, reports should be produced to show compliance to contract measures and to aid in identifying process improvement opportunities for all parties.

The typical reporting for vendor management consists of balanced scorecards for senior level management, project reports for project managers, and maintenance reports for support areas.

Balanced scorecard

These reports provide a complete picture of all measures required for managing the contract. These are typically summary reports that include data from multiple projects. The Balanced Scorecard Institute states that,

> *"the balanced scorecard was originated by Robert Kaplan and David Norton as a performance measurement framework that added strategic non-financial performance measures to traditional financial metrics to give managers and executives a more 'balanced' view of organizational performance".*

In the case of software vendor management, the scorecard should have multiple measures that show contract results. For example, even though productivity may be a key 'payment' metric, quality should also be included to ensure that in efforts to improve productivity, quality does not suffer. The report should also include a short analysis that explains the results reported to ensure appropriate interpretation of the data.

Project reporting

These reports focus on individual projects and are provided to project teams. The reports should contain measures that support the contract and provide insight into the project itself. Analysis should always be provided to assist teams with assessing their project and identifying process improvement opportunities to better meet the contract requirements.

Maintenance reporting

These reports are at an application level and would be provided to support staff. This data would provide insight into the maintenance/support work being conducted. Again, this would be in support of specific contract measures, but it can also be used to identify process improvement opportunities and/or identify which applications may be candidates for redesign or redevelopment.

Data Definition and Collection

Data definition and collection processes need to be developed to support the reporting. As stated in the book, "IT Measurement – Practical Advice from the Experts", this step should,

> *"focus on data definition, data collection points, data collection responsibilities, and data collection vehicles".*

Who is going to collect the data? When it will be collected? How it will be collected? Where it will be stored? These are important questions to drive the implementation of the contract measurements but these all depend on the most difficult step, data definition.

Data definition involves looking at all of the data elements required to support a measure and ensuring that both the client and the vendor have the same understanding of the definition. Since most software vendor contracts utilize productivity (FP/effort), this report will focus on defining the data elements of FPs and effort by way of example.

Function Point Data Definition

Function point guidelines should be developed for all parties to follow. This should include which industry standard will be used (e.g. International Function Point User Group Counting Practices Manual 4.x) as well as any company specific guidelines. Company specific guidelines should not change any industry standard rules, but provide guidance on how to handle specific, potentially ambiguous situations. For example, how will purchased packages be counted -- Will all functions be counted? Or will just the 'customized' functions be counted? Another consideration is how changes to requirements throughout the lifecycle will be counted. For example, some organizations count functions one time for a project unless a changed requirement is introduced late in the life cycle (e.g. system testing). Then a function may be counted more than once. Guidelines need to be established up front for as many situations as possible, but may need to be updated throughout the life of the contract as new situations arise.

Effort Data Definition

Effort can be one of the more contentious data elements to define in software vendor management systems. It is important to determine what life cycle activities are included in each aspect of the software vendor contract. For instance, if productivity is an SLA or a payment incentive, then vendors will want to exclude certain activities that clients may want to include. One example is setting up a test environment for a project. A vendor may want to exclude this from the productivity calculations while a client may think it should be included. A 'rule of thumb' is that if an activity is required for the project specifically the effort should be included. If the activity is to set up something that is for all projects to use, then it should be excluded. So in the test environment example if the vendor is setting up scenarios or simulators to test specific project functionality the effort should be included as

part of the project productivity calculation. If the vendor is installing servers to host test data and tools, the effort should be excluded. There are more effort categories to examine than can be included in this report. A non-category issue decision with effort is the inclusion or not of "overtime" hours. The recording of overtime hours in time management systems tends to vary widely even within organizations because many software development employees are not paid for overtime hours. The important thing is for vendors and clients to work together to define and document the guidelines.

Code Quality Analytics

In addition to the standard SLAs and beyond functional testing, code analytics and an application analytics dashboard can provide an IT organization with key insights into code quality, reliability and stability of the code being delivered by the vendor.

Code analytics tools, such as those provided by CAST Software, analyze the delivered code to detect structural defects, security vulnerabilities and technical debt. The metrics generated by these tools can be used as SLAs.

There is value in understanding what is being developed throughout the lifecycle. In this way security, performance and reliability issues can be understood and addressed earlier while still in development.

In a waterfall development environment, code analytics can be executed at defined intervals throughout the lifecycle and after deployment to production. In an Agile framework, code analytics can be run as part of each code build, at least once per sprint, and code quality issues can be resolved real time.

Having this information early in the lifecycle enables fact-based vendor management. Code analytics, along with traditional measurements provides the buyer with the information needed to manage their vendor relationships and ensure value from their IT vendor spend.

Conclusion:

A robust vendor management system includes:

- Pricing evaluation using industry standard measures to promote meaningful negotiations,
- Service level metrics backed up with historical or industry benchmarked data and
- Code analytics to ensure quality, reliability and stability are built into the systems being developed.

With these components in place an organization can efficiently manage vendor risk, monitor and evaluate vendor performance and ensure value is derived from every vendor relationship.

Sources:

- Balanced Scorecard Institute Website – www.balancedscorecard.org/resources/about-the-balanced-scorecard
- "IT Measurement – Practical Advice from the Experts." International Function Point Users Group. Addison Wesley (Pearson Education, Inc.).2002 – Chapter 6 Measurement Program Implementation Approaches.
- CAST Software Website – Application Analytics Software - http://www.castsoftware.com/

Appendix - Key Metrics

Size

Project size can be described in several ways, with software lines of code (SLOC) and function points being the most common.

Function Points

The industry standard approach to functional size is Function Points (FPs). It is a technology agnostic approach and can be performed at any point of the lifecycle.

FP analysis provides real value as a sizing tool. Even in software developed using the latest innovations in technology, the five components of function point analysis still exist so function point counting remains a valuable tool for measuring software size. Because a FP count can be done based on a requirements document or user stories, and the expected variance in FP counts between two certified function point analysts is between 5% and 10%, an accurate and consistent measure of the project size can be derived. And because FP analysis is based on the users' view and independent of technology it works just as well as technology evolves.

SLOC

Source lines of code is a physical view of the size but can only be derived at the end of a project.

It has some inherent problems, one being that inefficient coding produces more lines of code, another being the fact that determining the SLOC size of a project before it is coded is itself an estimate.

However, it can be used retrospectively to review a projects performance and you need to consider ESLOC or effective Source lines of code to remove the expert/novice factor of more lines of codes highlighted above.

Code analysis tools like CAST can provide excellent diagnostics and even FP counts based on the code.

Story Points

Projects in an Agile framework typically use Story Points to describe their relative size. They work well within a team but are ineffective at an organization level to consider relative size.

For example, a team can double their velocity simply by doubling the number of story points they assign to each story. They can also vary from one team to another as they are only relevant to the team, and sometimes, the sprint in question.

Time (Duration)

Simply the time measure for completing the project and/or supporting the application. This is calendar time, not effort

Effort

Effort is the amount of time to complete a project and/or support an application. Typically, hours is the metric used as it is standard across organizations. Work days or months may have different definitions across organizations.

Effort is one of the more challenging pieces of data to collect and the granularity at which you can analyze your measures is determined by how you record and capture the effort.

In agile teams, the effort is relatively fixed but flexible in the work performed, so if you want to analyze testing performance you need to know the split of testing work and so on.

Quality

Quality is a key measure in a vendor management situation as the quality of the code coming into testing and into production determines how well the project performs. We are all aware of the throw it over the wall mentality when deadlines start to hit and the resultant cost is defects being delivered to production.

A common request is how many defects are expected for a project of particular size.

The truth is that the answer is not straightforward as many organizations have a different view of what a defect is and how to grade them. Set your criteria with the vendor framework first and then record going forward. A view of historical performance is extremely useful here as well.

The defects should be measured during User acceptance test as well as go-live during the warranty period and used to predict future volumes and identify releases where further investigation or discussion is warranted.

Staff – FTEs

This is the people metric, it is usually measured in FTE or Full time equivalents so we have a comparable metric, you might have had 20 different people work on a project with a peak staff of 8FTEs or 10 people with the same effort and staffing profile, it's the FTEs that is consistent and comparable.

There is also person resource type that can be relevant here so consideration to things like onshore/offshore, contractor/permanent/consultant or designer/manager/tester may need to be included.

Cost

This may be actual cost or a blended rate per hour. Where multiple currencies are involved, assumptions may be need to be fixed about appropriate exchange rates.

Computer Resources

Computer resources covers the parameters of the technology environment such as platform, programming language etc. The final metric captures the "what?" and "how?" to allow to compare against similar project types by language and technical infrastructure