



A Predictive Approach to Test Planning

A White Paper

by

Brian Yahne

and

Lonnie Franks

April 2005

Version 1.1

**Property of
Lighthouse
Technologies, Inc.**

Do Not Remove
from Booth

Introduction

Testing is a complex, challenging phase of software development. Because of this, many organizations struggle to plan their testing needs and often underestimate the effort. This paper discusses our predictive approach to test planning that helps to ensure high quality, lower cost, and improved schedule, while reducing risk. The predictive approach estimates the actual number of defects that should be discovered in a particular test phase. Based upon this information, an organization can plan the number of test cases and testers that are needed to successfully accomplish the test objectives. In addition to assisting in test planning, the predictive approach can help manage the entire testing process by comparing actuals to the prediction at any point in time.

The Software Test Challenge

Software test planning is not simple. Typical questions that organizations find themselves asking are:

- How many test cases will need created?
- What are my staffing requirements?
- How long will it take?
- What will it cost?
- How do I know when I'm done?

Most organizations perform test planning using historical data from other engagements of roughly the same magnitude. This is somewhat effective until any system characteristic, such as project size, team composition, or business objectives, differ from the historical data.

For systems of approximately 2000 function points (roughly 100,000 source lines of code in C#), a Capability Maturity Model® Integration (CMMI) level 1 organization typically budgets 20% of the project cost for testing. For most CMMI 1 organizations, this is significantly too low. Industry data indicates that 40% of project cost for these organizations should be allocated to testing (10% to integration test, 10% to systems test, and 20% to acceptance test).

Without good knowledge of the testing domain, organizations typically underestimate testing costs causing:

- A low number of discovered defects
- Significant defect leakage to later phases of development
- The number of defects released to production is higher than industry standards

This leads to:

- Increased development costs
- Increased deployment costs
- Increased maintenance costs
- Missed schedules
- Poor acceptance by users
- Future business opportunities missed

The Predictive Approach

The predictive approach to test planning is based on predicting various aspects of the test phase, allowing organizations to better plan and manage their test activities. Effectively predicting the outcome of the test phase is far more useful in planning and managing test activities than simply analyzing historic, inconsistent

metrics. Typically, historical metrics are only useful when the project size, team size, team composition, technical understanding, and business objectives are consistent between projects.

For the predictive approach, Lighthouse utilizes industry data gathered from various sources including IEEE, Capers Jones, and the Software Engineering Institute to name a few. In addition, Lighthouse also collects and uses our own data gathered from many projects. We tailor this data specific to each engagement using our personal experience in the industry and the historical performance of the organization we are assisting.

Lighthouse first assesses the size of the software in terms of Function Points. Function points are an industry standard (refer to www.ifpug.org for more information on the International Function Point User's Group) way of quantifying the size of a software system. There is a significant amount of industry data associated with the use of function points. They can be used to help predict the size of the software system, staffing requirements, defect injection and removal rates, etc. In addition to prediction, function points can be used to validate that the delivered system matches the requirements.

Next, Lighthouse analyzes the processes and procedures that are applied to the project to get a thorough understanding of the development methodology used. We take into account several factors such as the size of the team, competency of the team members, quality of requirements and design artifacts, and thoroughness of peer reviews. Using these factors, we adjust the industry average defect injection and removal rates.

Lighthouse applies these rates against the size of the system to estimate the current defect density of the software. This data is then placed into our predictive models to determine:

- How many defects should be discovered and removed during the different phases of testing
- The required number of test cases
- The effort to develop and execute those tests

The inputs to the model include the adjusted defect injection and removal rates, the size of the system, and if fixed resources (testers) or fixed schedule is desired. As the organization becomes more mature, the outcomes from each project are fed back into the model to help more accurately predict future systems.

On our initial engagements, customers drive Lighthouse to make estimates that are very conservative because they have a tendency to be skeptical when the estimates are very large. However on subsequent engagements, the approach is proven and clients ask us to be less conservative.

In addition to using the predictive approach to assist in test planning, it can also be used to manage the entire test process. As metrics are gathered during testing, they can be compared to the predictive model, providing a quantifiable way of assessing the progress of the test phase. The following chart demonstrates this point.

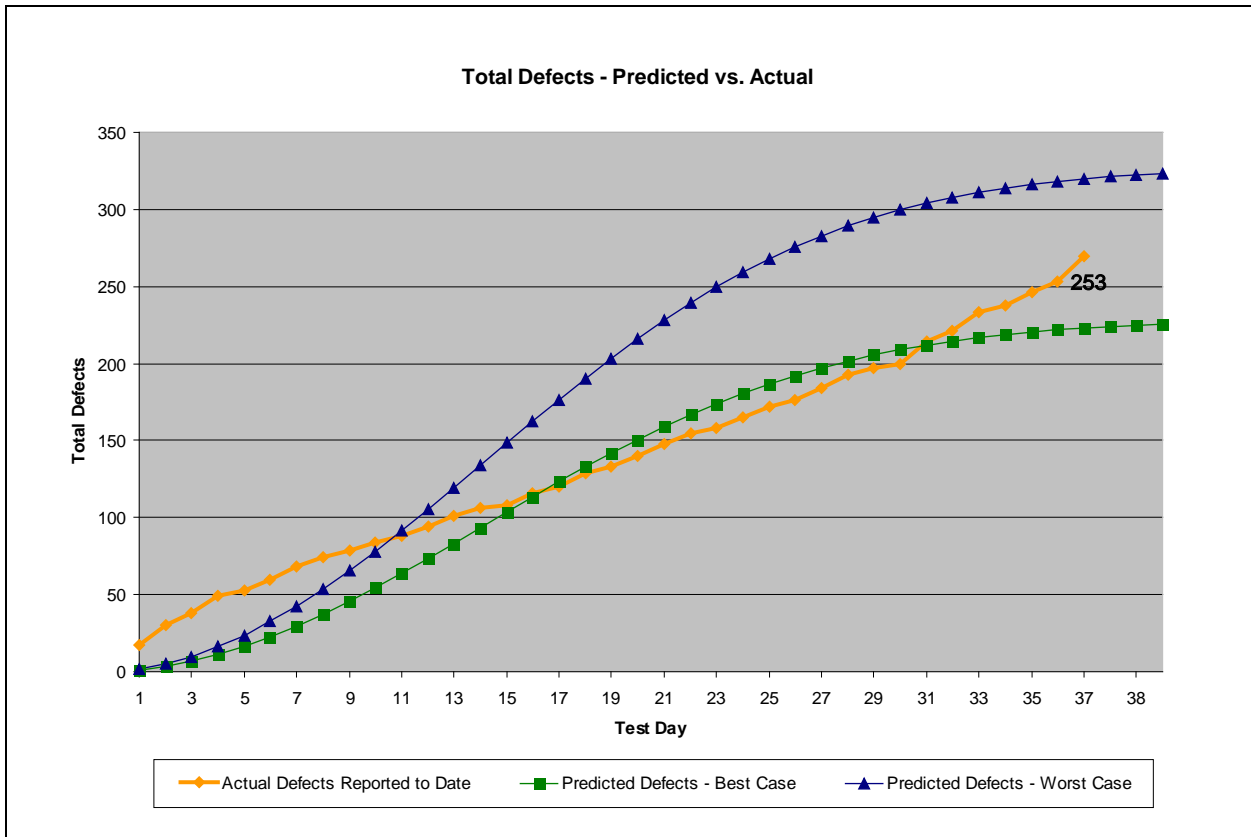


Chart – Planned vs. Actual Defects Detected

This chart shows the best case and worst case predictions for total defects detected during testing. In addition, the orange line represents the actual defects found during testing. We can see from this chart that the testing started out strong, but then began to lag behind the prediction. Because the predictive model was utilized, the testing staff understood that they were falling behind schedule and took corrective action. Without the predictive model, the testing organization would have no means to measure the progress of testing.

The Benefits of the Predictive Approach

The predictive approach to test planning provides the following benefits:

- Better understanding of the test schedule
- Better understanding of the test staffing
- Clear goals for the testing groups
- Lowered risk

As organizations mature in their test planning, the model can be refined to effectively “tighten up” the prediction, specific to the way they conduct business. The model can also be used to isolate areas where the organization may be deficient in some manner and then apply training as needed to maximize their effectiveness.

To better demonstrate this approach, let’s examine the following case study. The case study focuses on integration testing, however the same strategy can be applied to system and acceptance testing.

Case Study

The following case study discusses how Lighthouse applied the Predictive Approach to Test Planning with one of our customers on a system of approximately 636 FP. The customer was in the process of unit testing, quickly approaching their integration test. Lighthouse was engaged to analyze their processes and procedures and make recommendations for test planning including staffing and duration. We focused the prediction on integration testing specifically, but also provided data for system testing, and user acceptance testing. The customer used this information and initiated testing activities early to discover that the defect levels were higher than they had anticipated, allowing them to better plan the remainder of the testing activities.

Lighthouse was asked to assist in planning the integration test of a reasonably small system (32,000 SLOCs). The project team had spent approximately 40 staff months performing development. The system was quickly approaching the integration testing phase and there was little understanding of the effort associated with performing a rigorous integration test. The organization had a goal to complete integration testing in 4 calendar weeks.

We began by assessing the size of the software. We performed a code count on the existing C# code and SQL code base. This resulted in approximately 32,000 lines of code. By using industry average data of 52 SLOCs per function point in C# and 40 SLOCs per function point in SQL, we determined that the size of the system was approximately 636 function points.

Next, we analyzed the processes and procedures that were applied to the project. We found that no formal requirements or design artifacts were generated and no peer reviews of the existing code had been conducted. The project was executed in a small team environment (roughly 7 to 10 members) with requirements analysis and design being done primarily in an informal whiteboard mode. Given these factors, we modified the industry average defect injection and removal rates for requirements, design and code. The negative impact of no peer reviews outweighed the positive impact of having a small team. Therefore, overall, the defect injection rate was slightly higher than average, and the removal rate was lower than average.

Feeding the adjusted rates into our predictive model, we estimated that 678 defects remained in the system. To facilitate user acceptance of the system, we recommended that it was appropriate to achieve a 95% defect free system by the time they released to production. Based on this, we were able to estimate the number of defects that would need to be found in each phase of testing.

Generally speaking, on a system this size in a CMMI level 1 organization, we would see approximately 20% of the remaining defects being removed during integration test. Given the fact that this particular project had no formal QA/QC during the requirements, design, and code phases, we recommended that 42% of the remaining defects (285 defects) would need to be found in order to make integration testing successful. Effectively, integration testing needed to make up for the lack of formal process early on.

Using industry data, we estimated the number of staff hours to develop and execute the test cases required to discover the 285 defects mentioned above. In general, we would expect to have 4 test cases per defect found, 2 hours per test case created, 40 minutes per test case execution, and 4 test executions. This equates to approximately 10.7 staff hours per predicted defect to develop and execute the integration test cases for this project. This totals 17.8 staff months of effort (13.4 staff months to develop test cases, and 4.4 staff months to execute them).

The detection of defects was plotted against a Rayleigh curve to fully understand the daily rates of detection and the staffing requirements associated with performing the integration test. The Rayleigh curve is displayed below. The chart shows defects discovered during each day of integration testing.

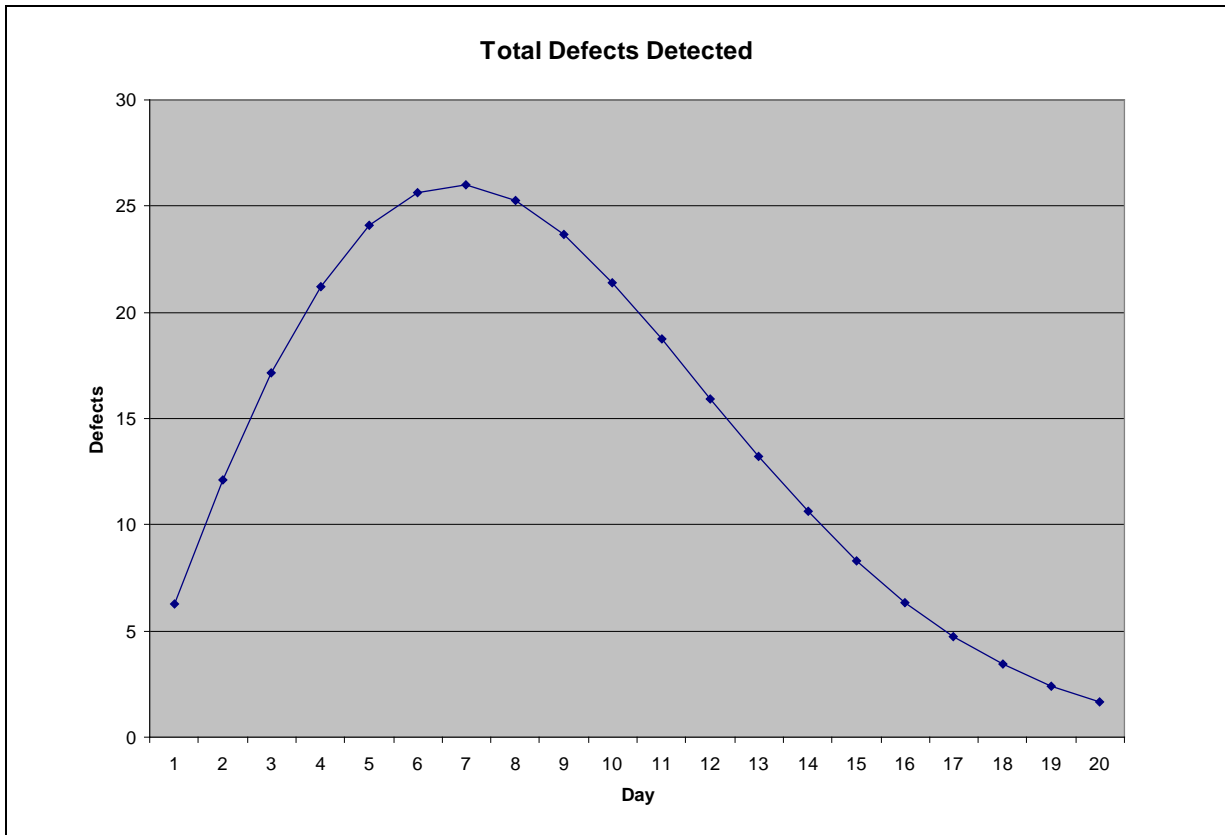


Chart – Total Defects Detected

The Rayleigh curve is used because it effectively models the typical defect detection rates: defect detection starts low, but quickly ramps up. The curve peaks and then, as there are fewer defects to find, falls more slowly until the integration test is concluded.

Our customer indicated that the testing period was fixed at 1 calendar month (20 staff days). Given that high-quality testers can find and document defects at a peak rate of 2 hours per defect, and the chart peaks at 26 defects detected on day 7, our peak staffing requirements are 7, assuming test cases are completed before integration testing starts.

Additionally, Lighthouse estimated the total effort for all testing (integration, systems, and acceptance) to be 31.8 staff months. Because fixing defects late in the life-cycle is many times more expensive than fixing the defects in the phase in which they occurred, we predict that the defects discovered during all phases of testing will require an additional 76.3 staff months of rework through the initial pilot of the system. In this rework estimate, we also included the effort required for retesting to verify the fix, as well as a 10% regression error injection rate for “bad fixes” (i.e., fixes that cause additional defects). The following chart shows the customers original estimate, and our predictions based on the model.

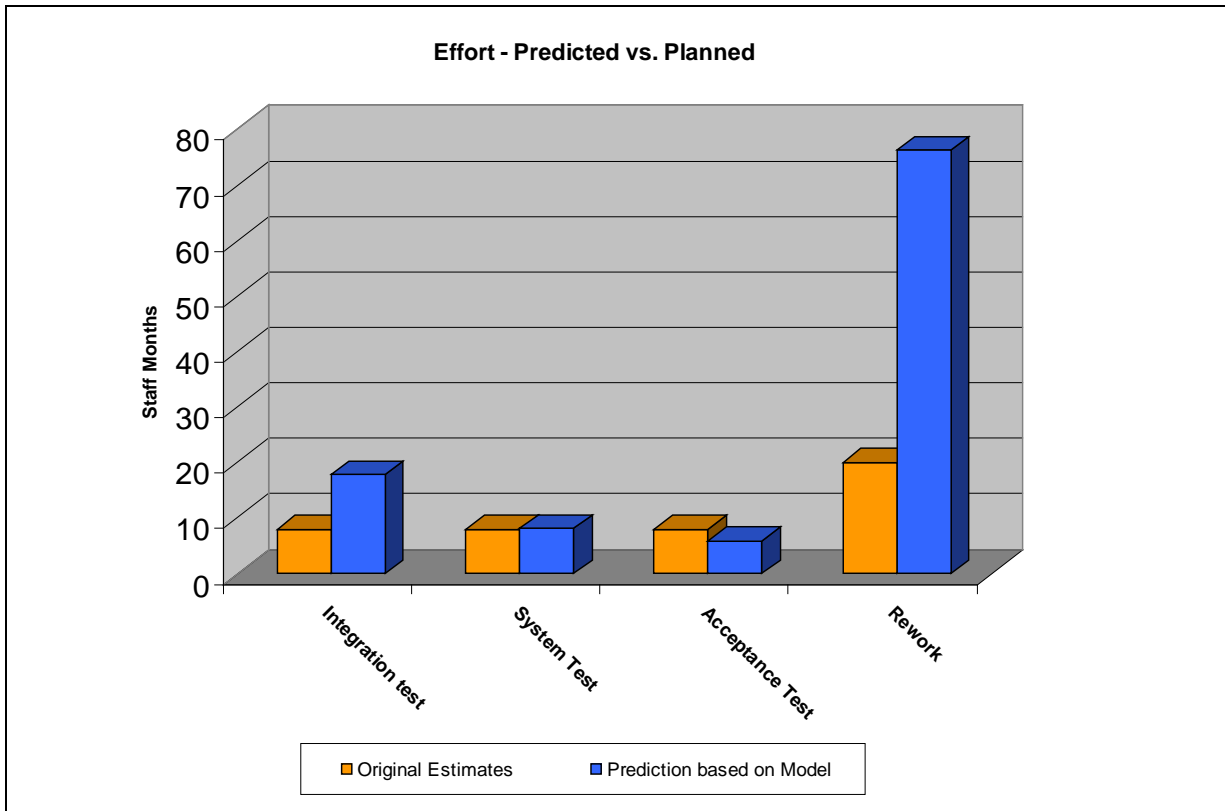


Chart – Planned vs. Actual Schedule

It is difficult to have faith in the numbers presented here, so we recommend organizations pilot the predictive model and validate their actuals against the prediction. Let the model prove itself. The model is simply a tool to help the organization. Applying that tool to the problem in the appropriate manner is critical to achieving the benefits of the predictive approach to test planning. To prove the predictions, we recommended to the customer to perform some sample code inspections and measure the number of defects discovered during a fixed-length inspection effort. This will validate the code-phase defect density. Additionally, we recommended that as specific functionality is completed, have an end-user test the system. The defects found should be recorded and an analysis should be done to identify whether the defects were a result of requirements, design, or code mistakes. This provides more information to validate the predicted defect leakage from these specific development phases.

After these two exercises are performed, the customer can adjust the defect injection and removal rates, and re-run the prediction model. With this information, the customer is now able to effectively plan and manage the integration test of the project and be reasonably certain of the outcome.

It is worth noting that based on our advice, the customer engaged an end-user to begin testing early. The end-user conducted testing during an 80 hour period and identified 150 defects, 80% of which were requirements defects. This was higher than our original estimate because the prediction was very conservative. The customer now understands the value of predicting these defects and acting on them early in the project lifecycle, and has subsequently asked us for less conservative estimates.

Conclusion

The predictive approach to test planning requires a shift in the culture of the organization. It must be embraced and supported to be effective. This cultural shift drives the organization to be more proactive in their planning, not only in testing, but across all development phases. The predictive approach to test planning is a process that focuses on knowing the number of defects that need to be removed during a particular test phase, and then planning the test cases, team size, cost, schedule, and processes to achieve the objectives. The predictive approach is also a useful tool in managing the testing phase. This will result in higher quality software, lowered cost, and improved schedule, while reducing risk.

About the Authors



Brian Yahne worked as a Sr. Software Engineer for Lighthouse Technologies, Inc. He has more than 10 years of experience in the software engineering industry. Brian has acted in many roles on software development engagements including technical lead, architect, and developer. In addition to software development, Brian works with various teams at Lighthouse Technologies to perform assessments of external software organizations, develop predictive models for customers, and develop and maintain the Lighthouse CMMI level 3 processes and procedures



Lonnie Franks is an Executive Consultant for Lighthouse Technologies, Inc. He has more than 30 years of experience in the software engineering industry. Lonnie has worked as a software developer, analyst, architect, Director or Architecture, Director of Engineering and Vice President of Research and Development for a Fortune 100 company. Lonnie regularly consults with Fortune 1000 companies to deliver predictive techniques to manage large-scale software development and maintenance.



Lighthouse Software Testing is committed to software quality from the point of project initiation through the software testing life cycle. Whether you are considering offshore software development or looking to outsource software testing for an upcoming project, Lighthouse's expertise in the software development life cycle (SDLC) can help you achieve maximum efficiencies in all phases of the project. We use industry benchmarks and carefully measured project tracking assessments to lower development costs, shorten time-to-market and provide optimal software performance.

To learn how Lighthouse Software Testing can help with your next software project, please contact Jeff Van Fleet at (937) 458-0055 or visit our [contact page](#) to submit an inquiry.