Agility, Uncertainty, and Software Project Estimation

Todd Little, Landmark Graphics

Summary

Prior studies in software development project estimation have demonstrated large variations in the estimated versus actual result. This paper examines these variations by analyzing three years of historical project data representing 120 "for market" projects developed by a market-leading oil and gas software company. The study reveals the following findings, some of which go against common industry beliefs:

- The ratio of actual over estimate follows a log-normal distribution.
- The initial project estimate tends to be a target with only about a 10-20% chance of being met.
- The median project comes in at 75% over the target, and the average (mean) project comes in at 100% over the target.
- To have 90% confidence, the project estimate would have needed to be 3-4 times greater than the target.
- This behavior and uncertainty range is nearly identical at all stages in the project lifecycle, in conflict with the "cone of uncertainty."
- EQF (Estimation Quality Factor, a project estimation metric proposed by DeMarco) also follows a log-normal distribution.

Successful commercial software companies developing software "for market" seem to follow inherently several key principles of Agile Software Development; one element in particular is "responding to change over following a plan" in order to ship working software to meet the most current needs of customers. These guiding principles are both boon and bane; following them can help make commercial software companies successful, yet they also accentuate the uncertain nature of software development.

Agility, Uncertainty, and Software Project Estimation

Todd Little, Landmark Graphics

Abstract

Prior studies in software development project estimation have demonstrated large variations in the estimated versus actual result. This paper examines these variations by analyzing three years of historical project data representing 120 "for market" projects developed by a market-leading oil and gas software company. These projects follow several key elements of the Agile Software manifesto. One element in particular is "responding to change over following a plan" in order to ship working software to meet the most current needs of customers. The study reveals the following findings:
(1) estimation accuracy follows a log-normal distribution, (2) our initial estimates are targets with only a small chance of being met, (3) the range between the target and an estimate with 90% confidence is about four times greater, and (4) this behavior and uncertainty range is nearly identical at all stages in the project lifecycle, in conflict with the "cone of uncertainty" presented by Boehm.

Introduction

Software development project estimation has long been a difficult problem for our industry. Many prior studies have shown that on the aggregate, software projects are invariably late and/or over budget or fail to deliver altogether[1,2,3,4,5]. Most companies do not cope well with uncertainty of this nature and as a result conclude that something must be wrong and should therefore be "fixed." While it may be possible to improve on software project estimation and to control the project to meet the estimation, an alternative perspective suggests that uncertainty is a natural property of the software development process. This may imply that control alone is futile and that a more productive solution is to embrace change, acknowledge uncertainty, and realize that value optimization is the objective, not project control. Agile methodologies recognize the value of plans and project controls, but emphasize the delivery of working software to meet the needs of customers at the time that they receive it.

Background of the Project Data

Landmark Graphics (www.lgc.com , a subsidiary of Halliburton www.halliburton.com) is the leading vendor of commercial software solutions for the oil and gas exploration and production market. Landmark has grown largely via acquisition, and our current software portfolio includes over 60 products consisting of over 100 million lines of source code. Over the past three years Landmark has been collecting data about all of our software development projects on a weekly basis. We did not have any specific process improvement plan in mind during this time, thus the data collected is relatively unbiased.

There were 570 total projects in the portfolio. Of these 570 projects, 120 projects were commercial releases for the general oil and gas market. The remainder included currently active projects, internal projects, and non-commercial releases. For the purpose of this study, only the 120 commercial releases were considered.

For each active project, the Project Manager recorded on a weekly basis a number of aspects of the project, including the status of the project, the current estimated delivery date, and the nominal phase of the project. Landmark did not follow any formal software development methodology, although several projects followed some form of iterative development, and many project teams followed guidelines from the Microsoft Solution Framework (MSF)[6]. For recording purposes, the four phases used were the four MSF phases: Envisioning, Planning, Developing, and Stabilizing. On projects that utilized iterative development, the Envisioning and Planning phases were usually quite short, and the iterations were all considered to be the Development phase regardless of whether they were planning, developing, or stabilizing that iteration. Typically this was followed by a final Stabilization phase. While Landmark did not follow any formal methodology, nearly all of the 120 projects under consideration followed most of the principles of Agile Software Development.

Agile Software Development

Agile Software Development follows four value principles as set out in the Agile Manifesto[7]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

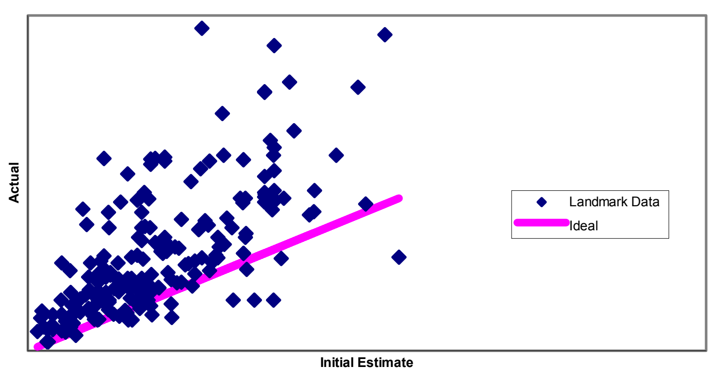
While the Agile Software Development movement is new, many of these principles are not. In the case of software developed "for market," the market is only interested in working software that meets its needs at the time of delivery. Organizations that try to control "for market" software projects by clamping down on change risk missing an evolving market. It is not really possible to negotiate a contract with the "market," but it is possible to engage customers in collaboration. And, the market is certainly far more interested in working software than in comprehensive documentation. As for the first element of the manifesto, the market doesn't really care one way or the other as long as its needs get met.

It is the author's belief that most successful software developed "for market" follows most if not all of the principles of the Agile Manifesto. Even many of our non-commercial projects followed these principles. What separates the commercial releases is that there is often limited degrees of freedom. When delivering software for a single customer, it may be possible to take liberties with scope or quality in order to meet a deadline; but when delivering for the mass market, taking many liberties could quickly result in loss of market share.

Actual versus Estimate

Figure 1 shows data extracted from the Landmark project database history. The x-axis shows the Initial Estimate of project duration, and the y-axis shows the Actual duration that the project required. The solid line shows the ideal case where the actual equals the estimate. It is easy to see that there is quite a scatter to the data, and that by and large the Actual duration was longer than the initial estimate, in some cases significantly longer.

Figure 1: Initial Estimate vs. Actual Project Duration



The plot in Figure 2 shows published data from DeMarco[3]. There is a slight difference from our data in that DeMarco is plotting Estimated Effort versus Actual Effort, but the scatter is quite similar. The blue line is at a slope of 2.0, which seems to validate the old adage "take the initial estimate and double it."

Figure 2: DeMarco Project Estimation Data

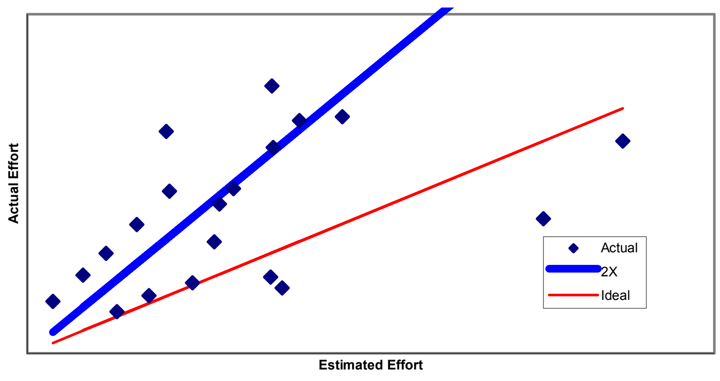


Figure 3 shows a plot of cumulative distribution of DeMarco's data and Landmark's data, plotting the Ratio of Actual/Estimate on a log scale. The magenta squares represent DeMarco's raw data and the blue plusses represent Landmark's raw data, while the red curve and the blue curve represent a log-normal distribution curve fit through the respective data points.

Figure 3: Cumulative Distribution Function of Actual/Estimate Ratio

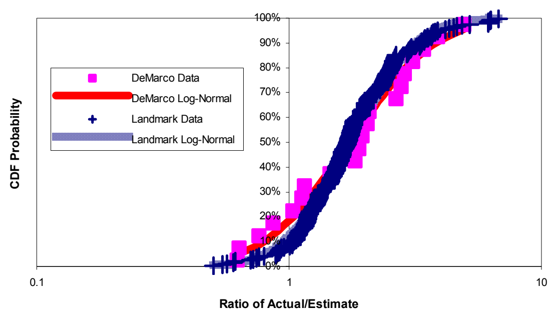
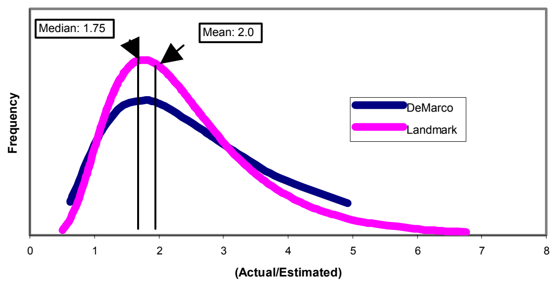


Figure 4: Probability Distribution Curve of Actual/Estimated



A cumulative distribution plot shows on the y-axis the percentage of data samples that have a value less than the value of the x-axis, e.g. 20% of the projects from DeMarco's data came in under the initial estimate (ratio=1.0). A normal distribution has a frequency or probability distribution in the shape of the classic bell curve. When plotted as a cumulative distribution function it takes on the integral of the bell curve and shows up as an S-curve. A log-normal distribution is similar to a normal distribution in that the frequency distribution of the log of the data is normal. In the world of uncertainty, it is common to report values at various probability confidences, particularly p10, p50, and p90 to represent 10%, 50%, and 90% confidence respectively.

Figure 4 shows the same data displayed as a probability distribution curve. Since this is plotted on a Cartesian axis, the log-normal curve shape is observed as a skew to the right. It is quickly observed that there is more area to the right of the median (p50), which implies that the average, mean, or expected value is greater than the p50. In our case the mean is approximately 2.0, while the median is about 1.75.

Landmark's data is remarkably similar to the data collected by DeMarco, and both demonstrate quite clearly that the data follows a log-normal distribution. There seems to be validity to DeMarco's observed definition: "An estimate is the most optimistic prediction that has a non-zero probability of coming true…or... what's-the-earliest-date-by-which-you-can't-prove-you-won't-be-finished?"[3] A variation on this definition is to say that a development target is an estimate with about a 10% chance of being met. Also of interest is the range between the p10 and the p90. For our data, the p90 is up at 3.25, meaning that if we really needed to be conservative, it's not just good enough to double the initial estimate; we may need to nearly quadruple it!

One of the advantages of having 120 data points is that it offers the ability to sub-sample the data and still have enough data to view a statistical trend. Landmark has grown through a number of acquisitions in five development centers serving four nominally separate business areas. Would the data for just one city, just one former company, or just one business unit show a different trend which might lead to potential process improvements that could be utilized by the rest of the organization? The data did not demonstrate this. We have sliced and diced the data in numerous ways, but as long as sufficient samples remain, the curve looks essentially identical to the full sample. Furthermore, we looked at those projects that did "well" at estimating to see if they used "better practices" than the other projects. Since we have three years of data to analyze, most of these project teams released multiple versions of the software product using essentially the same teams and processes. What we found is that many of the project teams whose estimate came close for one project ended up on the opposite end of the spectrum on subsequent projects.

Estimate as a function of project phase

The conventional wisdom is that estimation gets better as the project progresses. This was first stated by Boehm[1], and subsequently stated by many authors, most notably McConnell[4]. Figure 5 shows the cone of uncertainty reported by Boehm. At the Feasibility state the uncertainty band is 16X (from 0.25 to 4.0), while at Concept it has narrowed to 4X, and by Requirements it has reduced to 2.25X. This seems very intuitive.
.

Figure 5: Cone of Uncertainty from Boehm

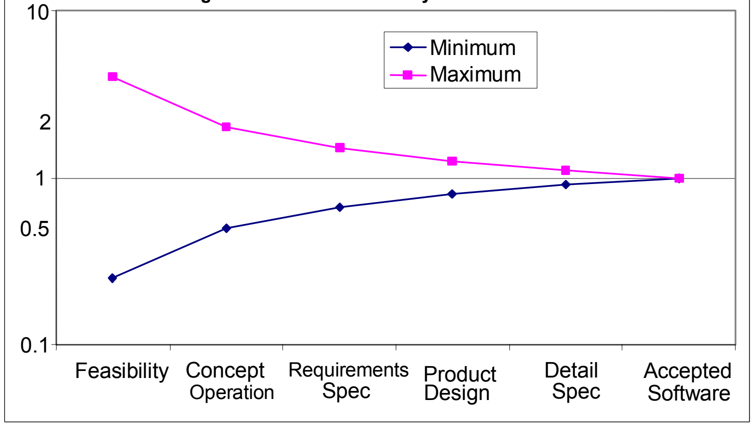
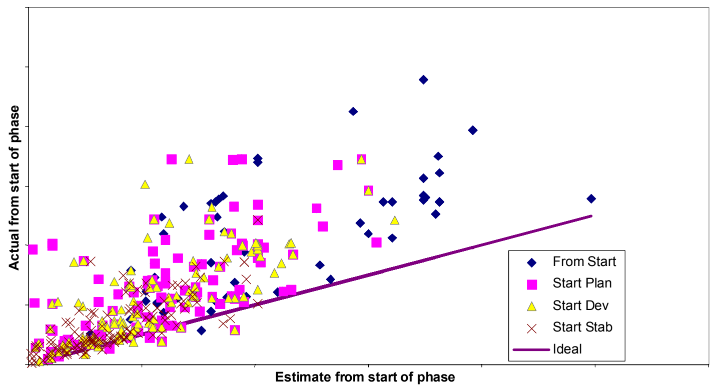


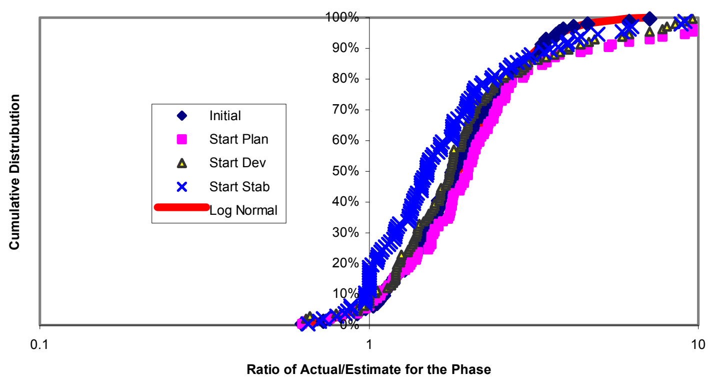
Figure 6 shows Landmark's observed data plotted with Actual remaining duration against Estimated remaining duration at the start of each phase. While the data points get closer to the origin, the scatter from the ideal does not seem to improve. What happened to the increased accuracy as we got further into the project?

Figure 6: Actual versus Estimate by Project Phase



This same data is plotted as cumulative distributions in Figure 7. These CDF curves are nearly identical for each phase. Interestingly, we again see that our most up-to-date target is about a p10. But perhaps more interesting is that we do not see a narrowing of the bands as predicted by the cone of uncertainty. Instead, the uncertainty bands remain constant, and at all stages of the project the range of uncertainty is about a factor of four between the p10 and the p90. Every CDF curve that we have extracted—be it from DeMarco's data, Landmark's data, or some subsample of Landmark's data—exhibits this same factor of four between the p10 and the p90; perhaps we are seeing an innate property of software project estimation.

Figure 7: Cumulative Distribution by Project Phase



Analysis of the findings

The data clearly indicates that Landmark's software project estimation follows a log-normal distribution with an uncertainty range between p10 and p90 of roughly a factor of four. This pattern is nearly identical to that found in DeMarco's data. Additionally, this uncertainty range remains constant over the life of the project, counter to the cone of uncertainty.

It is not a new finding that software estimation follows a log-normal distribution, yet it is still common to see estimates of espoused high confidence of +/-2 months, or +/-20%. Ranges that are given as plus or minus a constant time or constant percent are missing the exponential nature of the problem. Furthermore, the ranges rarely cover the real uncertainty. If typical project estimation follows the log-normal pattern the we observed, we should be estimating projects at +100%/-50%.

There may be nothing wrong with establishing targets by the "what's-the-earliest-date-by-which-you-can't-prove-you-won't-be-finished" method. That is probably a good starting point for a p10 estimate. It would be foolish to plan a business around a p10 estimate, but if the pattern that we observed is typical of most software development, then the full range of uncertainty could be defined by a p50 estimate of roughly twice the p10, and a p90 estimate of roughly four times the p10.

Managing the Uncertainty

Certainly it must be possible to reduce this unacceptable range of uncertainty? Traditional project management approaches, several of which are based on a strong belief in the cone of uncertainty, advocate stronger project control and greater planning. I believe that this frequently attempts to solve the wrong goal. It may be meaningless to ship on time, to spec, within budget, if a competitor is shipping software which has a greater value to the market. In that case the competitor will win nearly every time, and the prize for "good" project management might be going out of business.

Landmark's measure of success over these three years has much more to do with customer satisfaction and market share than with meeting knowingly aggressive targets. During these three years customer satisfaction has consistently been rated very high and has steadily increased each year. Market share continues to grow as well.

Nonetheless, there are things we can do to try to reduce and to manage our uncertainty. We can look to the Blackjack table for some ideas of coping with uncertainty. The Blackjack player is constantly faced with uncertainty. Various players cope with this uncertainty in differing ways. The novice plays with a superficial understanding of the game, and unwittingly believes he has played well if he gets lucky and comes out ahead. The professional understands the nuances of the game and uses uncertainty to his advantage. By counting cards he is able to understand when the odds are in his favor and thus increase his odds of winning. Yet despite using skill to turn the uncertainty to his favor, there is still uncertainty, and even the best Blackjack players will have bad days.

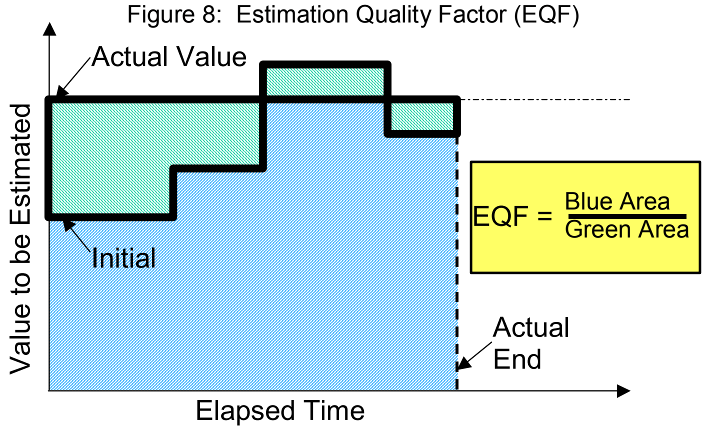
Agile methods take an approach similar to the Blackjack player. They acknowledge the presence of uncertainty and adapt to the situation rather than by trying to control the situation to meet the plan. Similar to counting cards, most Agile methodologies rely heavily on continuous feedback, particularly from customers or customer representatives. Just as with Blackjack, the uncertainty is not eliminated, but even a slight improvement provides a competitive edge.

References:

1. Boehm, Barry W, *Software Engineering Economics*, Prentice-Hall, 1981
2. Boehm, Barry W, *Software Cost Estimation with COCOMO II*, Prentice-Hall, 2000
3. DeMarco, Tom, *Controlling Software Projects*, Prentice-Hall, 1982
4. McConnell, Steve, *Rapid Development – Taming Wild Software Schedules*, Microsoft Press, 1996
5. McConnell, Steve, *Software Project Survival Guide*, Microsoft Press, 1998
6. Microsoft Solutions Framework, http://www.microsoft.com/msf/
7. Cusumano, Michael A. and Selby, Richard W., Microsoft Secrets,
8. The Agile Manifesto, http://agilemanifesto.org

Appendix A
Estimation Quality Factor

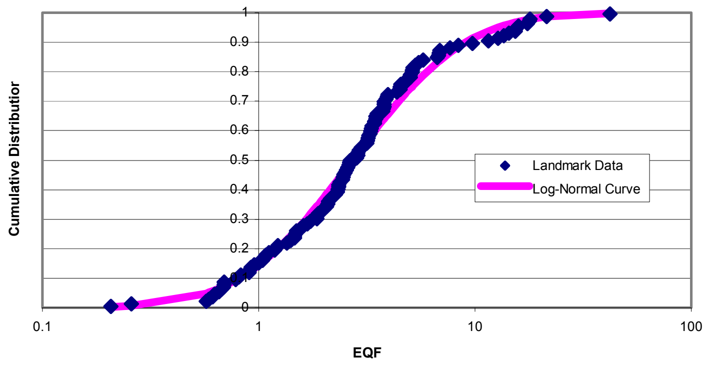
Figure 8: Estimation Quality Factor (EQF)



The Estimation Quality Factor (EQF) is a tool that DeMarco[3] proposed for measuring an organization's ability to adjust their project estimate over the project history. Figure 8 shows a graphical explanation of EQF. At the beginning of a project, there is an initial estimate. Over time that estimate may be revised up or down (black line between the blue area and the green area). At project completion, we know the actual value. The variation of the estimate from the actual integrated over time is the area shown in green. The blue area is the total under the Actual curve, or Actual value * End Duration. The EQF is the Blue Area / Green Area. The reciprocal of this, or Green/Blue, is the relative error of the estimate.

When the estimated quantity is schedule/duration, then the worst one should do would be to guess that it will ship by the end of the day, and make that assumption each day until the product ships. The green area would then be a triangle, the blue area a square, and the EQF = 2.0. Since 2.0 is the lower limit for EQF, it makes sense to plot the value of EQF-2.0. This is exactly what is shown in Figure 9, a cumulative distribution function of EQF for all of our projects. The data fits well with a log-normal distribution curve through EQF-2.0. Approximately 10% of Landmark's projects had EQF's lower than 2.8, half the projects had EQF's less than the median of 4.8, and 90% of the projects had EQF's lower than 11.7. In this case the log-normal distribution works in our favor, as the mean is higher than the median. These results compare to DeMarco's reports of a median value of 3.8 to 4.0.

Figure 9: Estimation Quality Factor Cumulative Distribution


Cumulative Distributior
EQF
Landmark Data
Log-Normal Curve