Technomics
Better Decisions Faster

What Does Agile Software Development Need?

Predictable Cost or Predictable Outcomes?

Christina Kosmakos-ckosmakos@technomics.net

Dave Brown-dbrown@technomics.net

International Cost Estimating & Analysis Association

5/17/2022

Abstract

Despite ongoing laudable attempts to advance the state-of-the-art in agile software estimating processes and techniques, the software development and cost analysis communities are far from ‘cracking the nut’. There is a dire need for a cost estimating approach that accurately predicts agile project costs with the correct granularity to enable budget and execution planning. This paper will detail the exploration of three approaches that utilize extrapolation from actuals to implement a solution.

Keywords: *Agile, Extrapolation, Software, Estimating*

Table of Contents

Introduction

Agile software development originated in the spring of 2000 with the purpose of speeding up development times and bringing new software to market faster. Whether agile has been successful in accomplishing its original goal does not change the fact that it has become increasingly prevalent in the software development community, replacing the previously prominent waterfall approach. This necessarily means that the cost estimating community must follow suit and adapt methodologies that support this transition.

From the perspective of the cost estimator, software development efforts would ideally have the ability to be planned far in advance with minor variability. Waterfall is more in line with this notion than the iterative nature of agile. Despite ongoing attempts to advance the state-of-the-art in agile software estimating processes and techniques, the software development and cost analysis communities are far from solving this problem. To date, a cost estimating approach that accommodates both an estimator's need for a long term view and agile developer's need for short term agility has not been developed.

This paper presents research resulting in three agile software cost estimating approaches intended to address these needs. The results represent an advance in creating a practical solution for predicting agile project costs with the correct granularity to enable budget and execution planning.

Overview of the Problem

Over the past decade, the software development process has transitioned from the prominent waterfall approach to the more prevalent agile approach. Waterfall is a sequential development process that flows through all phases of a project including analysis, design, development, and testing. An important differentiation of the waterfall approach is that each phase completely ends before the next phase begins. On the

other hand, agile is a time boxed, iterative approach that delivers software incrementally from the start of the project vice in total at once near the end.

The Planning Horizon: Short Term versus Long Term

Agile inherently takes a short term planning horizon and then iteratively adjusts the plan. Cost estimators prefer a long term planning horizon, seen by the typical Life Cycle Cost Estimate (LCCE). This ideological discrepancy between the two is the source of the challenge for accurately estimating agile software development costs. The initial steps in resolving the problem are thought to be: 1) assessing and better understanding current agile software development programs; 2) making best use of data that is typically available in an agile environment; and 3) creating methodologies for projecting future costs.

How Important is Cost?

A second aspect of agile development is that cost may not be viewed as a risk that needs to be managed. This viewpoint conflicts with a cost estimator's view that there is always cost risk, and that a risk-adjusted LCCE is one of the best tools to understand and manage cost risk. For example, an agile project manager might view the following parameters as fixed, and therefore known in advance: number of development teams, team size, and development schedule. If all of these variables are known, then it is a simple exercise to estimate cost by multiplying total team size by time and an average labor rate. For an agile project manager, the cost and schedule are fixed, whereas the amount and type of work completed at the end of development is not. The amount and type of work completed is therefore managed based on user input and continuously updated priorities. This view is summarized in the GAO Agile Assessment Guide[1]:

> *Since Agile programs have flexible requirements and fixed budgets for an iteration, some have argued that conventional performance management tools,*

[1] GAO-20-590G, AGILE ASSESSMENT GUIDE: Best Practices for Agile Adoption and Implementation, https://www.gao.gov/assets/gao-20-590g.pdf

such as life cycle cost estimating, are not applicable to Agile programs. Those arguments are made because Agile programs have structures and processes that are dynamic and iterative and spread planning activities throughout the program's duration instead of conducting extensive planning upfront, as in traditional program development.

One way to resolve these differing views of cost is for the cost estimator to project not only cost but also amount of work completed for this cost. That is, if cost is assumed to be fixed, then the relevant question (for both cost estimator and agile practitioner) is: What will be delivered at the end of the development life cycle? Even if the agile philosophy prevents an exact specification of the end product(s), it is still useful to quantify the amount of work completed. Agile metrics, such as story points, are one way to quantify work completed.

Exploration of Three Potential Solutions

Extrapolation from Actuals Background

This paper explores utilizing one of the four primary cost estimating techniques known as Extrapolation from Actuals. This technique uses actual cost of a system to predict the future cost of the same system. As with any cost estimating technique, there are associated strengths and weaknesses that should be considered and understood. One benefit of extrapolating from actuals is it utilizes actual costs (i.e., hard evidence) to predict future costs and therefore provides the *highest credibility and greatest accuracy when properly applied*[2].

Another benefit, which is particularly appealing for an agile environment, is it allows the estimator to continuously update the estimate throughout the life cycle of the program. An extrapolation from actuals can be revised and improved as frequently as actual cost

[2] ICEAA CEBoK, Unit I – Module 2

and technical data (again, hard evidence) is collected. For an agile program, monthly updates are often possible.

There are two primary weaknesses of an extrapolation from actuals approach. The first is that some amount of actual cost history (i.e., hard evidence) is required. This makes the approach unusable for any program in which software development work has not yet begun. In such a situation, the techniques described in ***Dynamic Software Cost Estimation*** (Gellatly, Jones, Wekluk, Brown, & Braxton, 2022) are a good choice when a measure of functional size (such as Function Points) is known. In situations where neither actual cost nor functional size is known, then the techniques described in ***Uncertainty of Expert Judgment in Agile Software Sizing*** (Braxton, 2022) should be considered.

The second weakness of extrapolation from actuals is that the work to date may not be representative of the remaining work to be completed. These weaknesses are characteristic of any extrapolation, and must be acknowledged.

With these benefits and drawbacks in mind, this paper explores three different variations of extrapolating from actuals within an agile environment. Each method is evaluated based on its ability to predict the cost of agile development, as well as its ability to predict the productivity and corresponding outcome of agile development.

Three Extrapolation Methodologies

The first of the three approaches utilizes a cost per month metric. The second employs a cost per story point metric, and the third leverages a three month rolling average cost per story point. These three approaches were assessed for how accurately they can predict future costs on a monthly basis.

Actual cost, collected from the start of development, is needed for all three approaches. Cost is collected on a monthly basis from the performing activity and reflects each sprint team's expenditures for a specific build. For example, if the performing activity was working on Build 5 of the software development effort with two different sprint teams,

then the cost would be collected as Sprint Team 1 Build 5 and Sprint Team 2 Build 5. The ability to differentiate costs by build by sprint team is critical, particularly if software metrics such as story points are collected in the same way.

The story point metric is considered in both the second and third approaches. Story points are a commonly used agile metric that measures the overall size and complexity of a user story, feature, or other piece of work. The number of story points associated with a user story represents the size and complexity of the user story relative to other user stories in the same project. There is no set formula for estimating story points. Rather, estimating the number of story points within a user story is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, and the risk inherent in it.

The preceding discussion highlights the need for cost estimators to have access to cost, story point and other metrics that address project complexity to enable cost estimation. A valuable source of cost and story point actuals is the agile project management tool known as JIRA, which some performing activities use. Specifically, this software is intended for use in tracking team efforts and it enables routine accessibility to useful monthly data. There are other agile project management tools (e.g., ProofHub, Wrike, Smartsheet, etc.) or periodic data calls that could provide the required data.

Two of the three approaches (i.e., 2 and 3) explored in this paper leverage several software several metrics, including Sprint Team, Story Points, Status, and Build. JIRA provides the following key information and context:

- number of story points completed
- whether the story points should be associated with Team 1 or 2
- the build associated with the story points
- status indicator which will inform whether the story points have been closed (i.e. completed)

These data fields should be collected on a monthly basis, similar to the cost data. They form the basis of an important computed metric -- closed story points per build per sprint team. It is important to consider only closed story points and story points in external review each month. We believe that story points closed in the month will better reflect the actual effort completed in that month by offsetting the incomplete story point work at the beginning and end of the month. This ultimately serves to better connect the story points with the cost metric. Once the data has been collected from JIRA and exported to Excel, it is ready to be analyzed.

Approaches 2 and 3 also employ planned story points. Each agile sprint team must identify the number of story points they anticipate closing or submitting for external review in upcoming months. These planned story points are logically viewed as the team's backlog.

Approach 1: Cost per Month

This approach only considers the *cost attributed to the build per sprint team, and generates projections using the actuals from the previous months.* This approach is normalized based on the working days per month. In order to generate projections, two months of cost data are required. The costs for the two months are then added together and divided by the sum of the working days for the two months. This provides an average cost per day metric which can be applied to the working days of the month whose cost is being projected. Projections are generated for each sprint team individually. In other words, this method uses the cumulative average cost per team per day as the basis for extrapolation. As more months of cost data are collected, the average cost per day metric is recalculated and then the build cost per sprint team projections are updated.

Approach 2: Cost per Story Point per Month

This approach considers the *cost attributed to the build per sprint team and the story points attributed to the build per sprint team*. An efficiency metric is then calculated by dividing the cost by the story points per month. Similar to Approach 1, two months of

data are required to develop projections. The costs for those two months are then added together and divided by the sum of the story points for those two months. This provides an average cost per story point metric which can be applied to the planned story points of the month whose cost is being projected. Projections are made for each sprint team individually. This method therefore uses the cumulative average cost per story point per team. As with the first method, when more months of cost and story point data are collected, the metric calculation and associated projection are updated.

Approach 3: Three Month Rolling Average Cost per Story Point per Month

This approach considers the *cost attributed to the build per sprint team and the story points attributed to the build per sprint team*, as done for Approach 2. An efficiency metric is calculated the same way as done for Approach 2. Unlike Approach 1 and 2, three months of data are required to develop projections. The costs for the three months are then added together and divided by the sum of the story points for the three months, as done for Approach 2 with two months of data. This provides an average cost per story point metric which can be applied to the planned story points of the month whose cost is being projected. Projections are made for each sprint team individually. This method differs from Approach 2 as it does not take a cumulative average approach when creating the cost per story point metric for projections, but rather takes a three-month rolling average. This approach smooths the data for the previous three months, but is not impacted by any data that is more than three months old. As with the first two methods, when additional months of cost and story point data are collected, the metric calculation and associated resulting projection are updated.

Comparing the Approaches

The dataset available for this analysis includes actual cost and story point data, by team, for 12 months. The dataset includes the minimal amount of two months of data required for Approaches 1 and 2 and three months of data required for approach 3. Meaning, estimates could be created for each approach starting in month 4. In months

4-12, estimates could be updated based on the new data available in the previous months.

In order to evaluate each approach, estimated costs and story points were compared against actuals. This enables evaluation of how well each approach estimates future monthly costs. The coefficient of variance (CV) and correlation were the two statistical evaluations utilized to understand the answer to this question. CV is the ratio of the standard deviation of the error terms to the expected mean. The higher the CV, the greater the level of dispersion around the mean, which indicates a relatively poor predictor. The lower the CV, the more precise the estimate. Correlation is also used to understand how the actual costs and projected costs move in relation to one another. A positive correlation indicates that the actual and projected costs move either up or down in the same direction, which indicates a strong predictor. In contrast, a low or negative correlation indicates a poor predictor.

The table below shows the CV and correlation for all three approaches for Sprint Teams 1 and 2, and then at the total level.

	Sprint Team 1		Sprint Team 2		Total	
Approach	CV	Correlation	CV	Correlation	CV	Correlation
1	0.37	31%	0.60	78%	0.48	54%
2	0.31	61%	0.59	68%	0.45	65%
3	0.34	57%	0.49	84%	0.42	71%

The results indicate Approach 3 is the best predictor by virtue of having the lowest CV (0.42) and highest correlation (71%). Approach 2 is the second best predictor, and Approach 1 is the worst predictor.

Based on these results, we conclude that a 3 month rolling average of daily cost per story point is the preferred method of extrapolation. The CV and correlation statistics give an indication of how well the predictor performs.

When looking at results by team, we see a slightly different outcome. For Team 1, the best predictor is Approach 2. For Team 2, the best predictor is Approach 3. We also

note that all three approaches are better predictors for Team 2 than the corresponding predictors for Team 1. These results, which give additional insight into this particular project, could be explained if Team 2 is better able to predict future story point requirements. This may be a result of Team 2 having more consistent, or less complex requirements assigned to them, or possibly more expertise in their ability to predict future story points.

Looking Beyond Cost

The full dataset, showing both cost and story point actuals by month and by sprint team, is shown in the table below:

Month	Team 1		Team 2	
	Cost	Story Points	Cost	Story Points
January	$ 116,825	119	$ 151,018	88
February	$ 134,048	61	$ 112,597	171
March	$ 114,612	140	$ 125,180	104
April	$ 165,175	114	$ 151,293	117
May	$ 149,500	118	$ 122,772	179
June	$ 108,191	132	$ 119,113	125
July	$ 133,288	156	$ 139,438	137
August	$ 108,022	83	$ 35,684	91
September	$ 159,247	120	$ 46,688	83
October	$ 94,635	117	$ 18,693	42
November	$ 52,542	98	$ 5,675	131
December	$ 32,408	78	$ 2,289	90

Something that stands out is cost is relatively variable. This is especially true when looking at the efficiency metric for each sprint team calculated as cost per story point, as seen in the table below:

Month	Team 1			Team 2		
	Cost	Story Points	Cost/Story Point	Cost	Story Points	Cost/Story Point
January	$ 116,825	119	$ 980	$ 151,018	88	$ 1,721
February	$ 134,048	61	$ 2,207	$ 112,597	171	$ 658
March	$ 114,612	140	$ 822	$ 125,180	104	$ 1,209
April	$ 165,175	114	$ 1,449	$ 151,293	117	$ 1,293
May	$ 149,500	118	$ 1,270	$ 122,772	179	$ 688
June	$ 108,191	132	$ 820	$ 119,113	125	$ 957
July	$ 133,288	156	$ 854	$ 139,438	137	$ 1,016
August	$ 108,022	83	$ 1,309	$ 35,684	91	$ 393
September	$ 159,247	120	$ 1,327	$ 46,688	83	$ 566
October	$ 94,635	117	$ 809	$ 18,693	42	$ 445
November	$ 52,542	98	$ 539	$ 5,675	131	$ 43
December	$ 32,408	78	$ 415	$ 2,289	90	$ 25

CV can then be calculated for the efficiency metric for each sprint team:

Cost/Story Point	CV
Team 1	0.45
Team 2	0.68

These results indicate that the efficiency metric of cost per story point is variable, as seen by the CV (0.45 and 0.68) for both sprint teams. This suggests that there is significant risk and uncertainty in agile software projects relative to how much work is accomplished. It would be easy for deferred work to proceed unnoticed since the variable efficiency indicates that the amount of work completed per dollar spent is not consistent.

Cost is relatively variable in this data set, however the risk and uncertainty would still exist when managing cost in a fixed environment. Since productivity is variable, it is unclear how much work will be accomplished for a fixed cost amount. Only by also considering an agile metric, such as story points, do we get a full picture of the entire outcome. These results challenge the agile community viewpoint discussed in the Overview of the Problem section, which explained that cost is not a risk that needs to be managed for agile development projects. This observation is supported by the results indicating Approach 3, which considers both cost and story points, is the superior method of extrapolation.

Based on these results, we conclude that both cost and productivity must be considered in agile development project estimation. Extrapolations that take both of these variables into account will ultimately produce better results than extrapolations that consider cost alone.

Application

Application of all the methods described in this paper is highly dependent on data availability. As with any extrapolation, the cost estimator must have access to relevant historical cost data. Our analysis suggests that at least two or three months of data is needed. Additionally, to use the methods described as Approach 2 and Approach 3, measures of completed and remaining (i.e., planned) story points are needed. Depending on the data available, these approaches could be used for budget and execution planning.

There are various advantages and disadvantages to each approach we researched. For Approach 1, the most advantageous feature is that actual cost per month is the only data required. In contrast, Approach 2 and 3 each require actual story points per month and planned story points of future months. The obvious disadvantage of Approach 1 is it does not consider effort accomplished with the expended funds each month. It requires the assumption that future cost is dependent solely on historical costs. Approach 2 and 3 are advantageous in the way that they consider the work accomplished with the expended funds each month. The data shows that cost is variable each month, thus there is a relationship between work accomplished each month and funds expended.

The statistical results, CV and correlation, provide a quantitative measure of how good each extrapolation approach is at predicting total cost. In our dataset, we found a high level of variance, both on a month-to-month basis and between the two sprint teams. These results indicate that more research and data collection is required to fully understand the cost and productivity of agile development. However, given the data

available for analysis, Approach 3 represents the best estimating method. The CV and correlation results should then be used as the basis for risk adjusting any estimate that uses this method.

Next Steps

In order to expand and grow the cost community's software development cost estimating effort further, collecting more data from other programs that utilize agile would be beneficial. This would potentially facilitate generation of various cost estimating relationships between effort, time, and cost. The long term goal is to establish a software cost estimating methodology that accurately predicts the agile effort of a program prior to initiation. In order to obtain a complete picture, we suggest a requirement for activities performing agile development projects to provide historical metrics such as the time spent on each story point and the anticipated story points to be completed in a given timeframe. The latter would be an effort to minimize the challenge of the iterative nature of agile and create a metric that can be used to predict more accurately.

Conclusion

This paper demonstrates that extrapolation from actuals is a viable method for estimating agile software development cost. Our research indicates that the best results are achieved when a combination of cost and productivity is considered. An estimation method that considers cost alone produces less accurate results and potentially masks the true risk and uncertainty of the project.

Which does agile need: predictable cost or predictable software outcomes? Cost estimators tend to focus on cost. However, we believe that within an agile environment, a cost-centric view will obscure a large portion of risk and uncertainty. As a result, the success of any agile software development project will depend on the estimator's ability to predict the final outcome of the development effort. Our research shows that the final outcome is likely to depend on multiple factors, including both cost and work

accomplished. Additionally, we have found that variation between development teams, and the development team's ability to forecast story point requirements are also likely to impact the final outcome of the project.

Ultimately, agile software development projects will be well served by close collaboration of the agile developer and cost estimator. The cost estimator who uses the extrapolation techniques discussed in this paper, will be able to provide the best estimate of project cost and productivity. The agile developer plays a critical role in the collection of data, predicting remaining story points, and understanding the context of the data. A collaborative effort that considers all these variables has the best chance at achieving the ultimate goal: a favorable project outcome.

References

[1] GAO-20-590G, AGILE ASSESSMENT GUIDE: Best Practices for Agile Adoption and Implementation, https://www.gao.gov/assets/gao-20-590g.pdf

[2] ICEAA CEBoK, Unit I – Module 2