Mining Software Effort Data: Preliminary Analysis of Visual Studio Team System Data

Lucas Layman[+]
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
lucas.layman@ncsu.edu

Nachiappan Nagappan, Sam Guckenheimer, Jeff Beehler
Andrew Begel
Microsoft Corporation
Redmond, Washington, USA
nachin, samgu, jeffbe, abegel@microsoft.com

ABSTRACT
In the software development process, scheduling and predictability are important components to delivering a product on time and within budget. Effort estimation artifacts offer a rich data set for improving scheduling accuracy and understanding the development process. Effort estimation data for 55 features in the latest release of Visual Studio Team System (VSTS) were collected and analyzed for trends, patterns, and differences. Statistical analysis shows that actual estimation error was positively correlated with feature size, and that in-process metrics of estimation error were also correlated with the final estimation error. These findings suggest that smaller features can be estimated more accurately, and that in-process estimation error metrics can be provide a quantitative supplement to developer intuition regarding high-risk features during the development process.

Categories and Subject Descriptors
D.2.8 **[Software Engineering]:** Metrics—*Performance measures, Process metrics, Product metrics.*

General Terms
Measurement, Experimentation

Keywords
Effort estimation, scheduling, prediction

1. INTRODUCTION
Accurate estimates of software development effort lead to more reliable schedule prediction. As work progresses on a software feature and the amount of remaining effort changes, inaccurate or varying effort estimates can serve as indicators that the feature may require further risk assessment and mitigation. Estimation inaccuracy can be detrimental to an organization's economics and credibility due to delayed releases, poor quality, and customer dissatisfaction with the product.

One approach to software effort estimation entails the creation of prediction models, including the classical COCOMO [2] and COCOMO II [3] models, and the use of function points [1]. These models focus on an algorithmic approach to build software effort estimation models. Other effort estimation efforts, such as the CoBRA® model [4], supplement expert judgment with quantitative data. There is tremendous potential to mine effort databases and to use machine learning algorithms to understand and build cost-estimation models using effort estimation revision data. Defect prediction, a current focus of the software mining community, can also be combined with effort data to understand the relationship between cost estimations and quality. This paper serves to highlight the need for research in mining software effort databases and discusses our preliminary work in this area.

This paper describes our research in mining software effort estimation data for the Visual Studio 2008 release of Microsoft's Visual Studio Team System (VSTS). The VSTS development group predicted the effort required to complete 55 features for the 2008 release and updated the actual and remaining effort tallies throughout the development process. We collected the effort estimation data from Microsoft's Team Foundation Server and performed statistical analyses to identify estimation error and relationships between error and effort metrics. We also asked team members about the causes of estimation error.

Our paper begins with our motivation and research goals in Section 2. We summarize related work in Section 3. In Section 4, we discuss our data collection, and experimental context. Section 5 presents our experimental findings. We conclude the paper with our contributions and plans for future work in Section 6.

2. MOTIVATION
Effort estimation in software development is still an open problem in research and in practice. Top software engineering conferences and journals including TSE, TOSEM, and ICSE yield few publications on effort estimation when compared to the amount of papers on software quality prediction. A better understanding of effort estimation will help organizations to predict and adhere to development schedules and will provide a more thorough understanding of risk factors affecting software development effort. In this paper, we aim to motivate researchers to mine effort estimation data to help understand various software engineering processes and phenomena and to encourage researchers to build

[+] This work was completed while Lucas Layman was an intern with the Software Reliability Research Group at Microsoft Research in the summer of 2007.

better prediction models to estimate software effort.

In our first experiment, we analyzed effort data extracted from the latest release of the Microsoft Visual Studio Team System (VSTS) project – a large software product developed at Microsoft Corporation. The main goal of this paper is not to provide definitive conclusions or research results, but to open discussion on mining and understanding software effort data. We had the following goals in our preliminary research experiment.

Goal 1	Identify the relationship between feature estimation errors (actual and relative to feature size) and various estimation metrics (e.g., # of updates, effort change)
Goal 2	Visualize overestimation and underestimation of features in VSTS.
Goal 3	Build time-based statistical models for effort data to predict future effort.

3. RELATED WORK

Algorithmic prediction of software development effort estimation began with research on COCOMO [2], COCOMO II [3] and function points [1]. Shepperd and Schofield [8] describe an approach for building effort estimation models by using similar projects according to attributes such as development method and size of the requirements document. Jørgensen et al. [7] report that predictions of an experienced maintainer were better than the predictions of an inexperienced maintainer for corrective and small, simple maintenance tasks only in an empirical study with 109 randomly selected tasks at a large Norwegian company. Jorgensen and Moløkken-Østvold [6] analyzed data from 68 completed projects and identified reasons for estimation error that include the role of the respondents, the data collection approach, and the type of effort analysis. Jørgensen [5] and Briand, et al. [4] present approaches for systematically integrating expert opinion into a more formal effort estimation process.

4. EXPERIMENTAL SETUP

We collected effort estimation data for all 55 features in the Visual Studio 2008 release of Microsoft's Visual Studio Team System (VSTS). VSTS is a suite of Application Lifecycle Management tools for effective team software development. The VSTS product is comprised of over 1 million lines of code. It is written primarily in C#, with some T-SQL, and some C++. The VSTS product group was formed in 2003, and is presently composed of 350+ personnel: 21 managers. The group is distributed over locations in Redmond, WA (USA), Raleigh, NC (USA), and Hyderabad, India. In its first release, Visual Studio 2008, the development team followed a traditional "plan-driven" approach where the entire feature set was planned in detail up-front and then updated periodically.

In the VSTS development group, work is divided into features that are implemented by small teams (4-5 persons) comprised of developers, testers, and one program manager, and are called "feature crews." Feature information, including effort data, is available through the Visual Studio IDE to all members of the team. Having the feature information centralized within the development environment lessens the amount of work required to update the data. Effort data for features implemented the Visual Studio 2008 release are stored and maintained in the VSTS system by program managers. Each feature in VSTS received an initial effort estimate in person-hours. The amount of estimated remaining work and the amount of completed work on each feature is updated periodically by members of the feature crew. The effort estimates and work completed are typically updated on a weekly basis.

For our research, we mined the revision history for each feature in the Visual Studio 2008 release of VSTS. All feature information is stored in the Team Foundation Server (TFS) component of VSTS. TFS serves as a central repository for the information, and its data is accessible through the Visual Studio IDE and a variety of database connections. TFS provides an On-Line Analytical Processing (OLAP) cube that affords easy access and querying of the data stores, including effort information. We collected revision history using TFS's query capabilities. An example effort revision history for a feature is shown in Figure 1.

Revision Date	Cumulative Completed Work	Cumulative Remaining Work	Cumulative NEWLY Completed Work	Estimation Change
8/21/2006	6	510	0	0
8/22/2006	70	558	64	112
8/31/2006	144	484	138	0
9/7/2006	174	454	168	0
9/14/2006	240	388	234	0
9/20/2006	286	342	280	0
9/22/2006	286	294	280	-48
9/28/2006	352	230	346	2
10/5/2006	394	188	388	0
10/11/2006	422	144	416	-16
10/19/2006	443	123	437	0
10/23/2006	451	235	445	120
10/26/2006	470	188	464	-28
11/2/2006	558	100	552	0
11/9/2006	552	61	546	-45
11/16/2006	583	30	577	0
11/30/2006	584	2	578	-27
12/13/2006	554	0	548	-32

Figure 1. Effort revision history for a feature in person-hours

We computed overall estimation accuracy according to the effort completed since the first available estimate. The "Cumulative Completed Work" (CCW) value is entered by a member of the feature crew to represent the number of person-hours thus far completed toward the feature. The "Cumulative Remaining Work" (CRW) value is also entered by a member of the feature crew and serves as an estimate of the amount of remaining work. The effort completed since the first estimate is calculated automatically as "Cumulative Newly Completed Work" (CNCW), where CNCW is the CCW in the revision minus the CCW in the first entry. This computation affords a better assessment of estimation accuracy for the many work items with work completed *before* the first effort revision, but it overcorrects for work items that had no work completed before the first entry. All calculations regarding effort estimation and accuracy use the CNCW (rather than CCW).

The "Estimation Change" (ΔCRW) column displays estimate change between effort revisions while accounting for work completed in the interval. A positive value indicates an increase in the amount of estimated work.

Table 1. Effort metrics

Effort metrics	Description	Formula
Actual Estimation Error (AEE)	The difference of the initial estimate from the cumulative newly completed work.	$AEE = CRW_1 - CNCW_n$
Relative Estimation Error (REE)	The initial estimation error proportionate to the amount of effort in the feature.	$REE = \frac{AEE}{CNCW_n}$
# of Estimation Updates	The number of effort entries for a work item sans those entries where the CRW has remained at zero. The first effort entry is not counted.	
Mean Estimation Change Magnitude ($\overline{EC}$)	The average magnitude of change in a work item's effort entries.	$\overline{EC} = \sum_{i=2}^{n} \frac{\lvert\Delta CRW_i\rvert}{\#\ of\ Estimation\ Updates}$
Mean Update Interval ($\overline{UI}$)	The average number of days that have passed between effort entries.	$\overline{UI} = \sum_{i=2}^{n} \frac{T_i - T_{i-1}}{\#\ of\ Estimation\ Updates}$
Mean Estimation Speed ($\overline{ES}$)	The estimation change magnitude relative to the number of days between estimates.	$\overline{ES} = \sum_{i=2}^{n} \frac{\lvert\Delta CRW_i\rvert}{T_i - T_{i-1}}$

5. FINDINGS

The metrics defined in Table 1 were computed for the collected data. The Actual Estimation Error (AEE) and Relative Estimation Error (REE) provide the overall effort estimation accuracy of a feature. A negative AEE/REE value indicates that the work item's effort was *underestimated* – the amount of work to complete the feature was *greater* than originally estimated. The AEE value is useful for interpreting the impact of effort estimation inaccuracies, while the REE value can be used to normalize estimation accuracy for comparison between features.

5.1. Metric Correlations

We performed a Spearman rank-order correlation on the metrics to identify any relationships between the metrics collected, the amount of new work, and the estimation error. The results are summarized in Table 2.

Table 2. VSTS 2008 metric correlations

	(*CNCW*) (Person Hours)	|*REE*|	|*AEE*|
Absolute Relative Estimation Error (|*REE*|)	Not significant		
Absolute Actual Estimation Error (|*AEE*|)	$\rho = 0.635$ $p < 0.01$		
# of Estimation Updates	$\rho = 0.754$ $p < 0.01$	Not significant	$\rho = 0.573$ $p < 0.01$
Mean Estimation Change Magnitude ($\overline{EC}$)	$\rho = 0.642$ $p < 0.01$	$\rho = 0.573$ $p < 0.01$	$\rho = 0.825$ $p < 0.01$
Mean Update Interval ($\overline{UI}$)	$\rho = -0.522$ $p < 0.01$	Not significant	$\rho = -0.450$ $p < 0.01$
Mean Estimation Speed ($\overline{ES}$)	$\rho = 0.668$ $p < 0.01$	$\rho = 0.507$ $p < 0.01$	$\rho = 0.761$ $p < 0.01$

The correlation between the CNCW and the |AEE| is expected and implies that larger features tend to have larger estimation errors. The correlations between CNCW and $\overline{EC}$ (estimation change magnitude) and $\overline{ES}$ (estimation speed) also suggest that larger work items require more substantial revision in effort estimates during development. Not surprisingly, more estimation updates are correlated with larger features and higher |AEE| values. Related to this, larger features and those with higher |AEE| values are updated more frequently.

The $\overline{EC}$ and $\overline{ES}$ values are strongly correlated with both the |REE| and |AEE| values. This implies that the magnitudes of the individual effort estimation changes are indicative of the overall accuracy of the initial estimate. This relationship is expected, but is also important for two related reasons. First, the relation of these in-process metrics to the overall estimation accuracy suggests that these *in-process* metrics are viable measurements of overall effort estimation accuracy. Second, these in-process metrics may have value to the product teams during development as a means to identify and quantify which features are likely to require large overall changes in their effort estimates.

5.2. Visualizing Effort Estimation History

Visualization of the effort estimation revision history for a feature can provide insights into a feature's history. Estimation evolution charts are "burn-down" charts with a separate line for the completed work. Figure 2 shows an "ideal" estimation evolution chart. The chart displays the "burn-down" of the cumulative remaining work (CRW) along with the "burn-up" of the cumulative newly completed work (CNCW). The Y-axis units are person-hours. The X-axis displays the date of the revision, however the X-axis units are the revisions themselves and *not* the time span between revisions.

The evolution chart in Figure 2 is "ideal" because the CRW and CNCW lines mirror each other. The lines would be an exact reflection if the CRW and CNCW values were changing in concert in each segment (i.e., the amount of estimated remaining work was dropping by the amount of completed work).

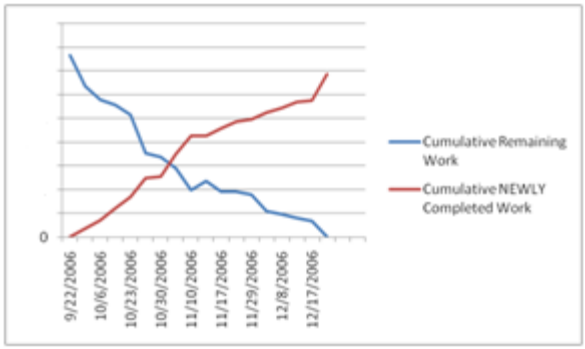

Figure 2. An "ideal" estimation evolution chart from a VSTS 2008 feature

One can gauge the actual estimation error (AEE) from the evolution chart by evaluating the distance on the Y-axis from the beginning of the CRW line and the end of the CNCW line – greater distance implies more actual error. Figure 3 shows an underestimation – indicated by the difference in the magnitude of the CRW and the CNCW.

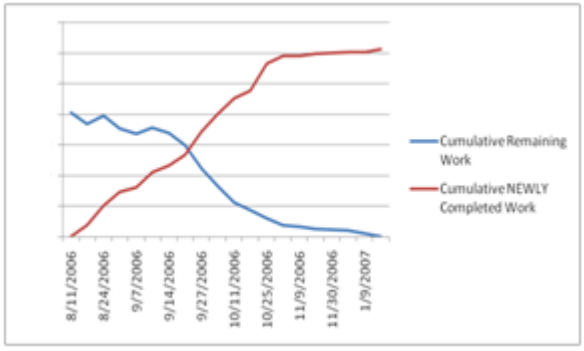

Figure 3. Estimation evolution chart showing a 33% underestimation

Effort evolution charts were created for all features in the two releases of VSTS under study. The feature crews that had estimation errors (over/under estimation) were contacted to ask for reasons for the errors. Conversations with VSTS personnel revealed common causes for estimation inaccuracy: requirements change, changes in feature scope, developer resource constraints and external commitments, newly formed teams, and unfamiliar feature requirements.

6. FUTURE WORK AND CONCLUSIONS

Our main intent with this paper is to motivate further research in the software mining community to investigate effort databases. A wealth of effort and process information, easily gathered, is available for analysis for process improvement when developers are willing and motivated to update effort data. The data gathered for our study was readily available from the TFS system available in VSTS, and the analysis effort enabled by TFS was simple and yet informative for the team.

In the future, we plan to investigate whether the use of statistical prediction models is useful for identifying trends in effort revisions. If we can fit the effort data to statistical growth models, we can understand better the trends in the effort growth/modification and use this information to help plan future effort estimates. We plan to collect additional information regarding causal relationships, expert decisions, and feature attributes to supplement the historical data used in effort prediction models.

In this paper, we described our preliminary research in mining effort estimation data for 55 features in the Visual Studio 2008 release of Microsoft Visual Studio Team System. We observed the following:

- There is a positive correlation between actual estimation error with feature size
- In-process metrics of estimation error were correlated with the final estimation error.
- Visualizing effort estimation revisions allowed us to quickly identify over/under estimation errors
- Conversations with development team members revealed common, yet unsolved, causes for effort estimation inaccuracies

Our findings supplement conventional wisdom and empirical data on the sources of effort estimation inaccuracy and that larger features are harder to accurately estimate than smaller features. The correlation between in-process estimation accuracy metrics and overall accuracy metrics suggests that monitoring estimation accuracy throughout the development process can be useful for identifying risk-prone features.

7. REFERENCES

[1] A. J. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validations," *IEEE Transactions on Software Engineering*, vol. 9, pp. 639-648, November-December 1983.

[2] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

[3] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall, 2000.

[4] L. C. Briand, K. E. Emam, and F. Bomarius, "COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment," in *20th International Conference on Software Engineering*, Kyoto, Japan, 1998, pp. 390-399.

[5] M. Jørgensen, "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation," *IEEE Software*, vol. 22, pp. 57-63, May-June 2005.

[6] M. Jørgensen and K. Moløkken-Østvold, "Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method," *IEEE Transactions on Software Engineering*, vol. 30, pp. 993-1007, December 2004.

[7] M. Jørgensen, D. I. K. Sjøberg, and G. Kirkebøen, "The Prediction Ability of Experienced Software Maintainers," in *4th European Conference on Software Maintenance and Reengineering*, Zurich, Switzerland, 2000, pp. 93-99.

[8] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Transactions on Software Engineering*, vol. 23, pp. 736-743, November 1997.