Effort Estimation in Component-Based Software Development: Identifying Parameters

Randy K. Smith
The University of Alabama
P.O. Box 870290
Tuscaloosa, Al 35487-0290
rsmith@cs.ua.edu

Introduction

Traditional software development is characterized by the structured programming paradigm introduced in the late 60's and early 70's. This paradigm relies on top-down functional decomposition to derive software modules. The structured programming paradigm provides a monolithic view of the software development process. Traditional software effort estimation models capture this monolithic view of software development. In these models, software effort is projected at the large-grained system level [1,4].

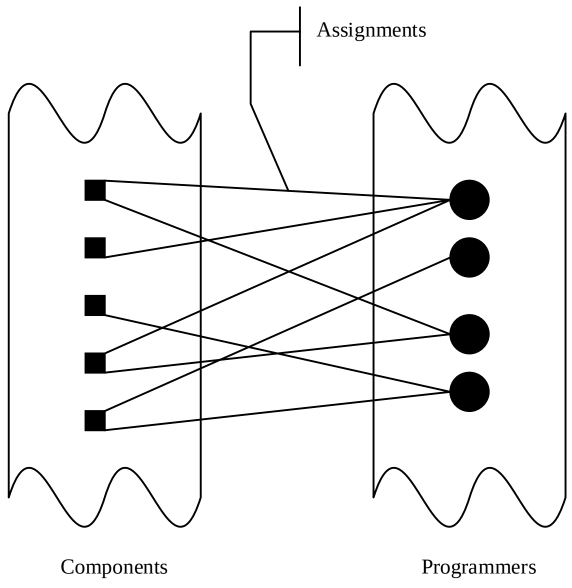
Contemporary development practices characterize a software application as interacting, independent components. These components may be Commercial-Off-The-Shelf (COTS) components, internally developed reusable components or newly developed software artifacts. To accurately predict effort in Component Based Software Development (CBSD), a fine-grained approach is needed to identify and classify the relevant cost factors.

The limitations of traditional models are problematic for effort estimation in CBSD. Effort estimation in CBSD is concerned with deriving estimations for small, concurrent development projects. The small projects allow for quantitative and direct measurements of the factors influencing cost. The measured factors need to be continuous and easily transferable from project to project. The limitations of traditional models demonstrate the need for a new approach to effort estimation in CBSD.

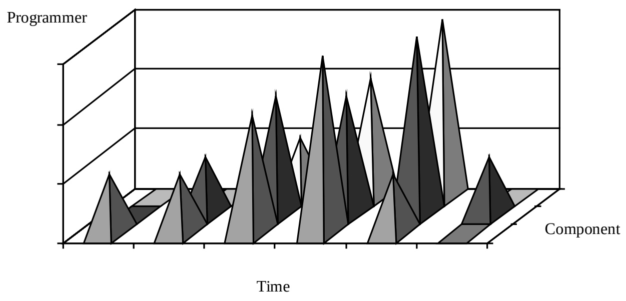
Description of Research

The primary characteristic of CBSD that is not captured by traditional cost estimation models is the effect of scheduling at the component level. Components may be developed early or late relative to a given project; also the schedule for a given component may be compressed or relatively spread out. In our work, we treat time as a "snapshot" in order to examine the state of the system. A snapshot consists of one "time-unit"; in our work, we consider the impact of varying the time intervals associated with one unit.

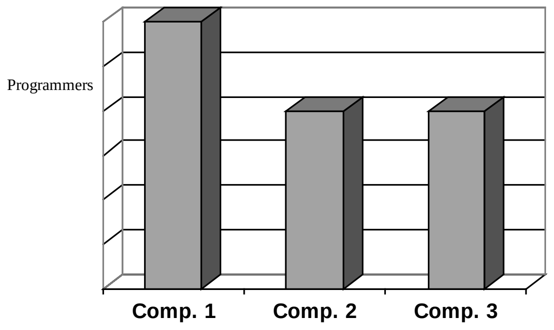
In CBSD, a programmer can be working on multiple components at any given time. Conversely, at any given time a component may have multiple developers. During a single time unit, the project can have multiple components under development by multiple programmers. The following figure captures the project state at a single time unit.



This approach to effort estimation is captured by a three dimensional view of the development process. In this three dimensional model, the *x* axis captures the component under development, the *y* axis represents time and the *z* axis portrays the programmers on the project. The following diagram displays a three-dimensional view of the development process.

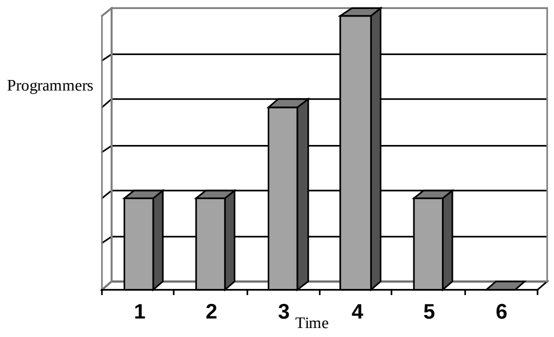


The primary thrust of this research involves the metrics and properties of CBSD that can be derived by passing planes through the three dimensional model. These planes are called "contextual planes." By passing a plane through the model that is perpendicular to an axis, we capture the "context" of the development activity as it relates to the other two dimensions. For example, by passing a plane perpendicular to the time axis, we can examine the number of components and the number of developers active for a particular time unit (see the following figure). By passing a plane perpendicular to the



Components under Development at a given time unit

component axis, we can examine the number of programmers working on a particular component and the number of time units required by the component (see the figure below). This view of CBSD allows an effort estimation model to examine metrics and effort factors at a fine-grained level of detailed that is not captured by traditional effort estimation models.



Development activity for a single component

Using this concept of components, time units and programmers, we can derive a suite of metrics that characterizes the effect of scheduling on CBSD. We have developed formal definitions of these metrics. However, for the sake of brevity, the metrics are informally described below.

- ***Intensity***-The ratio of the quantity of actual time spent on a component to the number of time units scheduled for the component.

- ***Concurrency***-The degree to which multiple programmers are working simultaneously on a single component.

- ***Fragmentation***-The degree to which a single programmer is working simultaneously on multiple components.

- ***Component Project Experience***-The number of components that have been completed as part of the project prior to work beginning on a particular component.

- ***Programmer Project Experience***-The number of components that have been previously completed by the programmers assigned to a particular component.

- ***Team Size***-The number of programmers assigned to a particular component.

These metrics provide a starting point to examine effort estimation in CBSD. The metrics capture the dynamic nature of component development that distinguishes the paradigm from traditional software development. In order to determine the application of the metrics to effort estimation, three research questions must be answered.

Question 1	*Which of the proposed metrics best predict component development effort?*

The processes involved in CBSD differ from those of traditional software development. The proposed metrics represent a new area of research in measuring the processes of CBSD. Their efficacy in estimating effort must be determined.

Question 2	*Which of these proposed metrics significantly add to the predictive ability of COCOMO?*

COCOMO is a well-studied and accepted effort estimation model. By augmenting the COCOMO model with the proposed metrics, a new model can build upon the experience inherent in the COCOMO technique.

Question 3	*Can the proposed metrics be used as the basis for a more parsimonious substitute for COCOMO?*

This research study addresses six quantitative metrics for CBSD. An effort estimation model that incorporates a few quantitative parameters would be beneficial to the software industry. It would provide the groundwork for further exploration into effort estimation in CBSD.

Empirical Study

In an effort to answer these research questions, a formal research study has been undertaken with data provided by a regional software contractor that specializes in information management solutions for state governments. The firm is currently under contract to provide the state government with a client-server solution to replace a COBOL-based ISAM database system. The legacy system contains over 3000 screens and 1 million lines of COBOL code. The new solution will incorporate client-server databases and end-user systems deploying a contemporary windowing operating systems. The new system will support in excess of 400 users distributed through a statewide network. The system will support departmental accounting, a large-scale bidding system for state infrastructure, inventory control and departmental personnel systems.

The 16-member development team has duties distributed among software development, computer networking and database administration. The development environment consists of a series of networked workstations utilizing a windowing operating system. The application development environment consists of COTS libraries, application generators, database systems, and internally developed domain specific reuse libraries.

There are two types of data examined in this study: product data and process data. The product data for this study were obtained from Hierarchical-Input-Process-Output (HIPO) diagrams for each of over 400 components. Using the preliminary HIPO diagrams, Unadjusted Function Points (UFP) were computed for each HIPO using the counting rules outlined by Dreger [2]. Function Points are a technology independent measure of the functionality of a system as seen by the user [2, 4].

The process data are obtained from the time accounting system used for billing. The data includes a daily activity log giving:

- The date the work was done.
- The task number on which the work was done.
- The identification number of the employee performing the work.
- The number of hours worked in quarter hour increments.

From this log, a mapping is made from task number to HIPO number and from the employee identification number to employee job title. The job title is used as a relevant estimate of experience.

Data Analysis

In order to compare the predictive ability of the component-based parameters against a known model, organic-mode Intermediate COCOMO estimates were calculated for the components of this project. An additional measure of programmer experience was obtained by surveying the team in regards to their experience in general and their experience with the various factors involved in this development environment. The experience data is used by the Intermediate COCOMO model. The Unadjusted Function Point was used to determine Lines of Code estimates for input into the Intermediate COCOMO model. After determining a Lines of Code estimate, each HIPO was examined to determine the impact of the 15 cost factors required by the Intermediate COCOMO Model. Effort estimations were made for each component using the Component Level Estimation Form (CLEF) given by Boehm [1].

The data covers development efforts for 400+ components. The data will first be subjected to a COCOMO analysis to establish a baseline for addressing the research questions. Question 1 will be addressed by subjecting the six CBSD metrics and the actual effort data to a correlation analysis. Using this analysis, we can determine metrics that affect effort. In a split-half analysis, we will use approximately half of the component data and the results of the correlation analysis to address question 2. The COCOMO model will be augmented by the CBSD metrics and a regression performed. If the regression demonstrates that this augmented model is more predictive than our COCOMO baseline, we will validate the resulting model using the second half of the component data. Finally, again using a split-half analysis, we will examine the CBSD metrics augmented with size and reuse metrics to answer question 3. This examination will use a regression analysis compared to the baseline COCOMO estimates. If improvement is found, we will validate the new model using the second half of the component data.

Conclusion

This research identifies and quantifies parameters that impact development effort in CBSD. The parameters identified in this research specifically examine the characteristics of CBSD. They capture data that is available at the fine-grained level of detailed afforded by CBSD. This fundamental research lays the foundation to examine the processes utilized during CBSD. The research has significant implications in the areas of effort modeling, CBSD process understanding, and continued dialog of the differences between CBSD and traditional development practices

References

1. Boehm, B., *Software Engineering Economics*, Prentice-Hall, Inc., 1981.

2. Dreger, J., *Function Point Analysis*, Prentice-Hall, Inc., 1989.

3. Gries, D., “On Structured Programming” in *Software Design Strategies*, Bergland, G., Gordon, R., eds., IEEE Computer Society, 1981.

4. Fenton, N., Pfleeger, S., *Software Metrics: A Rigorous and Practical Approach, Second Edition*, PWS Publishing Company, 1997.