Determinants of Variability in Function Point Estimates

Ashok Subramanian
University of Missouri

Mary C. Lacity
University of Missouri

Cost control has become the motto for most business entities today. Information Systems departments are acutely aware of the need to manage their resources in a cost-effective manner. Consequently, efforts are underway to develop reliable metrics for measuring the size and complexity of software development projects. These metrics would then enable IS managers to effectively manage their portfolio of IS development projects by making optimum resource allocation decisions. Function Points represent one such metric which has gained much acceptance in the IS community. This study is an extension of previous studies which have examined the reliability of the function point metric. Specifically, the major components of Total Unadjusted Function Points (TUFPC) variability were determined and their relative significance was assessed. Students in a major Midwestern university were trained to estimate function points using Dreger’s rules for counting function points. The results indicate that in order to minimize errors in TUFPC estimates, function point counters must focus on identifying the appropriate number of function points. The classification of functions into appropriate classes or the assignment of appropriate complexity weights did not appear to significantly influence the accuracy of TUFPC estimates. The study concludes with recommendations for practitioners as well as future research in this area.

Despite the growing maturity of the information systems (IS) field, one of the greatest challenges still facing practitioners is the management of software projects. Many large software projects are delivered late, over budget, and often, with questionable quality delivered to end users. Many “best practices” in management have been developed as remedies, including user participation on project teams, structured systems analysis and design techniques, and quality assurance functions. Despite these practices, business unit managers, who pay for applications, find it difficult to determine the quality of delivered IS products. IS managers also find it difficult to estimate project cost and completion times. Further, beyond an individual project, business unit managers question the IS department’s performance in terms of productivity and effectiveness, compared with external organizations. These questions are becoming increasingly important as outsourcing options become more viable. Business unit managers want a way to assess their internal IS functions vis-a-vis market offerings (Lacity et. al., 1995).

For many years there have been calls for application metrics which can help address these management issues. Many practitioners subscribe to the adage “If you cannot measure it, you cannot manage it”. Many software metrics have been developed, but typically, these metrics only satisfy a few management objectives. Lines of code (LOC), for example, calculate programmer productivity but provide poor estimates of project completion times and costs because LOC cannot be counted until the program coding is complete (Ince, 1990; Davis, 1992; Grupe and Clevenger, 1991).

Case-based reasoning models based on human experts, estimate software completion dates, but fail to estimate the value of the system to the user (Ince, 1990; Mukhopadhyay et. al., 1992). Key-business indicators, such as the number of invoices supported, provide estimates of the value to users, but do not provide project completion and cost estimates. Furthermore, all these metrics are dependent on individual technolo-
gies and industries, consequently, they provide a poor basis for external benchmarking of productivity and quality.

The only widely accepted software metric that provides a comprehensive measure is function points. Function points, developed by Albrecht (Albrecht, 1979; Albrecht and Gaffney, 1983), estimate software size based on the functions delivered to users quantified in the form of outputs, inputs, queries, files, and interfaces to other systems. As such, function points provide a measure from the users’ perspective i.e., what users actually see of an information system. Because the user interface is designed earlier in the system development life cycle than actual programs, function points can be calculated earlier than other metrics such as lines of code, thus providing better estimates of project cost and completion times (Strehlo, 1993; Verner and Tate, 1992). Because the user interface does not depend on any underlying technology or specific type of application, function points provide a measure that can compare productivity of IS personnel and quality of systems across technologies, applications, and industries (Grupe and Clevenger, 1991; Kemerer and Porter, 1992; Kemerer, 1993).

Function points are growing in acceptance. In 1986, the International Function Point User Group (IFPUG) was established. It grew to over 600 members by 1994. Today, IFPUG is active in countries including Australia, Canada, France, Germany, Italy, and the UK. IFPUG is actively involved in training and certifying practitioners to count, use, and manage function points within their organizations (Ibba and Longstreet, 1995). In addition to IFPUG, a number of consulting firms specialize in function point metrics. At a recent conference in Rome, Capers Jones, President of Software Productivity Research, presented results from an analysis of function points for over 6000 projects representing many industry sectors. In addition, function points have also been used as evidence of vendor incompetency in courtrooms in the U.S. and Canada (Jones, 1996). Despite the growing acceptance of function points, the reliability of the function points metric has been questioned (Tate and Verner, 1991; Valett and McGarry, 1989). In addition, political resistance from technical staff can sabotage measures (Jones, 1996). These two issues are discussed below.

The reliability issue resulted from research that shows a low inter-rater reliability, i.e., two individuals may arrive at different function point counts for the same systems. Prior research has focused on ways to improve function point reliability by simplifying the metric, or by determining the sources of function point variance, so that function point counters can be better trained (Jones, 1988; Kemerer and Porter, 1992; Low and Jeffery, 1990). For example, Low and Jeffery (1990) conducted an experiment in which experienced function point counters estimated the number of function points for two systems. They found that function complexity and file counting rules provided the greatest sources of variation. Kemerer and Porter (1992) surveyed function point counters to determine their error rates in identifying or classifying functions as defined by the Counting Practices Manual (CPM) function point counting rules. Based on their survey results, which indicated nine sources of errors, they conducted three case studies to determine which of the nine errors cause the most fluctuation in function point counts. They found that failure to follow the rules for counting menus and interface files were the greatest sources of variation.

The political resistance issue is a result of IS staff counting function points which in turn are used to measure their own productivity and quality of work. One of the authors experienced this issue firsthand, when working as a systems analyst. The author’s employer company implemented a function point counting program, however, significant political behavior became evident among the IS staff. A typical scenario, for instance, would be the following: Supervisor A has 5 analysts who support 10,000 function points. Supervisor B has 4 analysts who support, approximately, the same number of function points. Supervisor A is concerned that when management reviews these figures, they will conclude that Supervisor A has one too many analysts. Therefore, Supervisor A is motivated to liberally interpret function point counting rules in order to identify more functions, thus, justifying the higher head-count. Low and Jeffery (1990) have also recognized the role of politics in the estimation of function points.

Our contribution to the field of function points is to address the reliability and political resistance issue with a possible low cost solution. We believe that end users can be trained to count function points. The solution is low cost because once trained, end users can count function points instead of recruiting highly paid IS professionals or external consultants. The political resistance issue would be addressed because when function points are counted by end users, the metric is not used to measure end user productivity or quality of work. Furthermore, end users will gain valuable insights into the actual size and quality of their systems, as well as the productivity of their IS staff.

We address the reliability issue by decomposing errors in function point counts into three components - errors in identification of functions, errors in classification of functions, and errors in weighing of functions. We then assess which of these components has the most influence on the final function point count. Our results indicate that identification errors have the most significant influence on the accuracy of TUFFP estimates. This suggests that training should be targeted such that end users understand exactly what constitutes a function, so that they can identify the correct number of functions in their systems. Classifying and weighing of