# An Examination of Some Software Development Effort and Productivity Determinants in ICASE Tool Projects

GIRISH H. SUBRAMANIAN AND GEORGE E. ZARNICH

GIRISH H. SUBRAMANIAN has a Ph.D. in computer and information science from Temple University. He is currently Assistant Professor of Information Systems at the School of Business in Penn State, Harrisburg. His research interests include software engineering, software cost estimation, CASE technology, information technology adoption, and the strategic value of information systems. He has publications in *Communications of the ACM, Decision Sciences*, the *Journal of Systems and Software*, and *Journal of Computer Information Systems*.

GEORGE E. ZARNICH received a B.S. degree in computer science from Edinboro University and a M.B.A. from Pennsylvania State University at Harrisburg. He is currently a senior consultant with Technology Solutions Company. Previously, he was a systems engineer with Electronic Data Systems.

ABSTRACT: Integrated computer-aided software engineering (ICASE) tools and their effect on software development effort and productivity have gained interest in recent research. This research studies the applicability of function points and technical complexity factor as software development effort estimators for ICASE projects. In addition, the effect of three factors—ICASE tool type, systems development method, and ICASE tool experience—on software development productivity is studied. ICASE-based software projects from Texas Instruments and Electronic Data Systems were used in this empirical research. Function points accounted for 74 to 82 percent of the variance in software development effort. Technical complexity factor, however, had only a small indirect effect on software effort. While software productivity differences between the ICASE tool types could not be confirmed, productivity was significantly higher for the rapid application development method in comparison with the productivity associated with the traditional systems development life cycle method. Higher levels of ICASE tool experience were associated with significant increases in software productivity.

KEY WORDS AND PHRASES: function points, integrated CASE tools, rapid application development, software development effort estimation, systems development method.

FUNCTION POINTS (FP) METHOD [1] CURRENTLY PROVIDES THE ONLY ESTABLISHED industry standard of software size measurement in the area of systems development [19]. A high degree of correlation between function points and the work effort involved in developing software was demonstrated in non-CASE software projects [1]. Inter-rater and intermethod reliability of FP measurement is also shown to be sufficiently high, justifying its continued use as a measure of software size and as an estimator of software development effort [18]. The applicability of FP method as a software development effort estimator in the context of ICASE-tool-based software development is shown in a sample of nineteen software projects [4]. The research in [4] emphasizes the need for future research to study the applicability of their findings with a larger sample size and in multiple organizations. Hence, this research replicates the research in [4] by empirically studying the applicability of FP as an effort estimator in two ICASE environments using a larger sample of forty ICASE projects.

FP is also used to measure software development productivity [3, 4]. Empirical research presents a systematic productivity measurement model for ICASE tools to help organizations assess the benefit of CASE tools [3]. An extension to the research in [3] would be the examination of factors that are claimed to impact software development productivity for projects using ICASE tools. Tools, methods, and personnel experience are indicated as variables that influence software development effort in the systems development effort model [35]. As productivity is defined in [3] as a ratio of software effort to software size, these three variables would also impact software development productivity. Hence, this research studies the effect of the type of ICASE tool, systems development method, and ICASE tool experience on software productivity.

Texas Instruments' (TI) ICASE tool, Information Engineering Facility (IEF), and Electronic Data Systems' (EDS) ICASE tool, Integrated CASE (INCASE), are the two ICASE tools examined for significant productivity differences between them. Second, the effect of systems development method on software productivity is examined by comparing productivity differences between projects that used rapid application development (RAD) and projects that used traditional systems development life cycle (SDLC) development. The RAD method [21] is claimed to give much faster development and higher-quality results. Finally, the effect of different levels of ICASE tool experience on software productivity is studied.

## Review of ICASE Tools and RAD Methodology

THE ICASE TOOL PROVIDES SUPPORT FOR ALL PHASES of the system life cycle. The ICASE tools of the 1990s [9, 10, 11, 30] provide support for:

- Creating a model of user requirements in a graphical form;
- Creating software design models for both the data and the procedural code;
- Error checking, consistency checking, analysis, and cross-referencing of all system information;
- Building prototypes of systems and enabling a simulation of the system;
- Enforcing organizational standards for specification, design, and implementation activities in the system development life cycle;
- Generating code directly from the design models;

- Providing automated support for testing and validation;
- Providing support for reusable software components—in the form of designs, code modules, and data elements;
- Providing interfaces to external dictionaries and databases;
- Reengineering, restructuring, and reverse engineering of existing systems; and
- Storing, managing, and reporting system-related information and project management information.

Development in an ICASE tool environment is carried out by people trained to use modern toolsets, which integrate prototyping, graphical modeling, repositories, testing tools, and other aspects incorporated in ICASE toolsets [21].

"The best development teams have 10 times the productivity of the average" [21]. A key aspect of RAD is its use of small autonomous (SWAT) teams and joint application development (JAD) with user involvement in systems development. The effective use of teams in software development has gained considerable interest in recent research [15, 26, 33, 37]. JAD involves selected user representatives in a meeting to define or design an information system [7]. JAD imposes a structure for the meetings along with a clear agenda. Additional details on JAD are available in [2, 8, 28, 32, 34].

In SDLC, user specifications are frozen prior to the technical design phase. During this time, business needs continue to evolve and change while the software application system is being developed. By the time implementation occurs, the system is very often outdated or in need of many changes. With RAD, shorter elapsed time between design and implementation often means the system is much closer to the user/business needs. Constant involvement by the customers through joint application development (JAD) sessions provides a much better system in terms of meeting customer needs.

There are basically four essential aspects of fast software development—the tools, methodology, people, and management [13]. RAD techniques provide for five steps, including data modeling, prototyping, optimizing, integration, and deploying [5]. By utilizing the ICASE tools in RAD development, the tools enforce precision in diagramming data models and flows. The prototyping has also become an important part of the RAD development life cycle. The idea of prototyping is to put the basic system in the user's hands quickly so that the process of refinement can be completed [14]. The ability of ICASE to generate code, test and modify it quickly, and regenerate it provides the ability to operate in this prototype-like fashion. Prototyping is the basis for production, and hence the prototype evolves into the end product. Using these techniques, phased implementation has become more common. The most obvious benefit in using RAD is to achieve better-designed software that will require less maintenance later [12].

Studies by Capers Jones [16] have shown that conventional application development techniques—that is, structured life cycle with programming productivity aids—yields productivity in the coding phase of eight to ten function points per person-month. The goal is to increase productivity to a range of 100 to 150 function points per person-month [23]. A key challenge in implementing ICASE software is to get developers to use RAD as a supporting methodology, which adds to the workload up front but saves time in the long run.

## Research Propositions and Data Collection

### Applicability of Function Points as Effort Estimator for ICASE Projects

IT IS ARGUED THAT A MEASURE OF THE "FUNCTION" that the software is to perform can be used to systematically derive an estimate of program size [1]. The FP method is used for the prediction of program size and, eventually, the software development effort. The FP method espouses that the function points delivered to the user are:

a. number of external input types;
b. number of external output types;
c. number of master files;
d. number of inquiries; and
e. number of interfaces.

A weighted sum of these factors can be adjusted using the technical complexity factor (TCF) which is dependent on fourteen variables. The result is the adjusted function points that is used in software sizing and effort determination. Appendix A provides a detailed explanation of function points calculation for our study, with an example. Empirical testing of the function points method demonstrates the superiority of this approach over an approach based on the size in lines of code [17]. Function points also can be used in measuring productivity. The adjusted function points delivered per person-month of work effort is an effective measure of productivity used for ICASE tools [3, 4].

TCF is used as an indirect determinant of effort in the FP method [1]. TCF is used to adjust FP which has a direct effect on software development effort. Both unadjusted function points (UFP) and adjusted function points (AFP) are shown to be significant estimators of software development effort in nineteen ICASE projects [4]. Hence, we propose:

*Proposition 1: In ICASE-tool-based software development, function points are a significant estimator of software development effort.*

One could argue that TCF can also have a direct effect as an estimator of effort for ICASE projects. However, it was shown that in non-CASE environments the impact of the fourteen complexity factors is small [17], and hence the direct effect may not be present. The direct effect of TCF as an estimator of effort for ICASE projects, however, was not studied in [4]. Hence, we propose:

*Proposition 2: In ICASE-tool-based software development, TCF is a significant estimator with a direct effect on software development effort.*

## Software Development Productivity Determinants

The use of ICASE has been shown to increase productivity levels measured in function points delivered per person-month for First Boston Bank [3]. Other studies also report productivity improvements through the use of CASE tools [24, 25, 29]. CASE tools that support the disciplined approach have a better chance at increasing productivity levels [31]. This research examines the presence of productivity differences between two similar ICASE tools—IEF and INCASE—in an attempt to find out the impact of ICASE tool selection on software productivity. Both ICASE tools provide all functionalities expected from these tools. A detailed description of these two ICASE tools is provided in appendix B.

The faster development of RAD results in the delivery of more function points per person-month resulting in higher productivity. The claim of the superiority of the RAD method to SDLC for ICASE tool projects is empirically verified in this research.

ICASE tool experience is critical in achieving higher productivity. Poor software productivity is attributed to the absence of CASE tool training for systems personnel [36]. ICASE users often experience a productivity decrease for the first three to six months, and it often takes twelve to eighteen months before productivity gains are visible [20]. It is amply clear from this finding that a higher level of CASE tool experience obtained over a longer period of time would result in higher productivity.

The effect of these three factors on software productivity is presented in the software productivity research model (figure 1). Software productivity is defined as the function points delivered per person-month and is consistent with its use in [3, 4, 16, 23].

From this research model, we propose:

> *Proposition 3: There are no significant productivity differences between the two ICASE tools—IEF or INCASE.*
>
> *Proposition 4: There are no significant productivity differences between RAD and SDLC methods for ICASE-based software projects.*
>
> *Proposition 5: There are no significant productivity differences between different levels of ICASE tool experience.*

The data set originally contained 53 projects (19 IEF and 34 INCASE) that were developed within the last five years. From this set, 40 projects (12 IEF and 28 INCASE) appear in the study since they had complete and accurate information necessary for further analysis.

The FPs were calculated by the project managers and documented. For the purposes of this research, the FPs were independently calculated by us for all projects and this independent FP measurement was used in the study. Our FP calculation agreed quite closely with the calculations of the project managers. Both organizations have well-defined procedures, standards, and quality control steps for calculating FPs and documenting them.

The data set collected includes an assigned project number (1 through 40), tool name (IEF or INCASE), methodology (RAD or SDLC), tool experience (low, medium, or

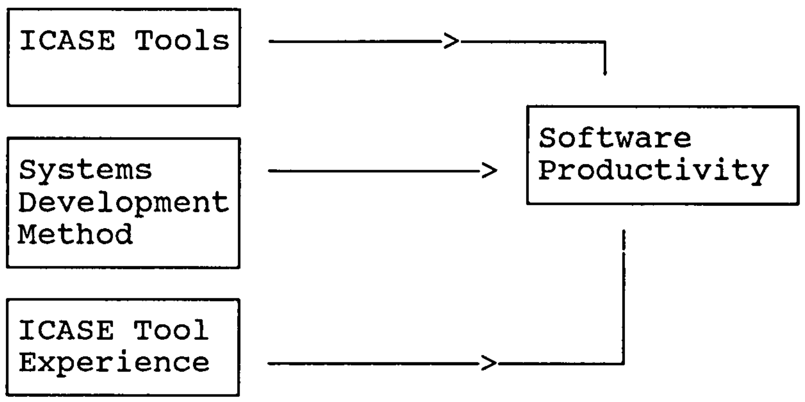

*Figure 1.* Software Productivity Research Model

high), actual effort in person-months, adjusted function points (AFP), unadjusted function points (UFP), and technical complexity factor (TCF). Software productivity measured as the number of function points per person-month (FPMM) is calculated as AFP divided by effort. The TCF values did not vary much across the projects (mean = 0.8923; S.D. = 0.1126) which would suggest that the projects were comparable.

In defining the data, project and tool are self-explanatory. The methodology is defined as RAD or SDLC. The ICASE tool experience category is broken into three levels. We chose this three-level classification in consultation with selected project managers from these organizations. The first level—low experience—is defined as no project member has over 1.5 years of experience utilizing the respective ICASE tool. The third level—high experience—is defined as at least one-half of the project members have over three years of experience utilizing the respective ICASE tool. The second level—medium experience—is determined as the project team falling somewhere between the low and high levels of experience.

## Results

THE INTERCORRELATIONS AMONG UNADJUSTED FUNCTION POINTS (UFP), adjusted function points (AFP), technical complexity factor (TCF), and effort are shown in Table 1. All correlations in Table 1 were significant at the 0.01 level except the correlations of TCF with UFP, AFP, and effort. The correlations of UFP with AFP shown in Table 1 are comparable to the correlation value of 0.981 shown in earlier research [4, p. 142]. In order to test proposition 1, a regression of effort as the dependent variable and function points as the independent variable was conducted and the results are provided in Table 2.

It is amply clear from the $R^2$ values in Table 2 that UFP or AFP or its log transformations explain about 73 to 82 percent of variance in the dependent variable effort or its log transformation. Log transformations were used in a recent estimation

Table 1. Correlation Results

| | UFP | AFP | TCF | Effort |
|---|---|---|---|---|
| UFP | 1.000 | 0.9947 | 0.2174 | 0.9519 |
| AFP | 0.9947 | 1.000 | 0.2729 | 0.9723 |
| TCF | 0.2174 | 0.2729 | 1.000 | 0.3213 |
| Effort | 0.9519 | 0.9723 | 0.3213 | 1.000 |

Table 2. Regression Results of Effort versus Function Points

| Dependent variable | Independent variable | $R^2$ | *F* value | Signif. of *F* | *T* value | Signif. of *T* |
|---|---|---|---|---|---|---|
| Effort | UFP | 0.7388 | 107.5 | 0.0000 | 10.37 | 0.0000 |
| Effort | AFP | 0.7479 | 112.7 | 0.0000 | 10.62 | 0.0000 |
| Log(effort) | Log (UFP) | 0.8105 | 162.5 | 0.0000 | 12.75 | 0.0000 |
| Log(effort) | Log (AFP) | 0.8195 | 172.6 | 0.0000 | 13.14 | 0.0000 |

model to "stabilize the error variance while preserving the linearity of the original relationship" [22], and hence our research results also include log transformations.

Since the research in [22] clearly emphasizes the importance of using scattergrams in effort estimation, the scatterplots shown in figures 2 and 3 were generated. Figures 2a and 2b show the scattergrams of the regression between log-effort and log-UFP. Figure 2a shows the scatterplot of both the actual and predicted effort values versus log-UFP with the associated 95 percent confidence interval. Figure 2b shows the plot of the studentized residuals versus predicted values. Figures 3a and 3b show the scattergrams of the regression between log-effort and log-AFP. Figure 3a shows the scatterplot of both the actual and predicted effort values versus log-AFP with the associated 95 percent confidence interval. Figure 3b shows the plot of the studentized residuals versus predicted values. From these figures, it is clear that the regression assumptions such as homoscedasticity and normality hold good for the regressions of log-effort versus log-UFP and log-effort versus log-AFP. The similarity of the distribution between the UFP and AFP distribution adds additional support for the comparability of the projects.

Proposition 1 is supported as the *F* and *T* values are significant in all the regressions in Table 2. Hence, function points is a significant estimator of software development effort for ICASE projects. This result reaffirms the finding in [4] that FP is a significant estimator of development effort. The UFP and AFP are comparable in their predictive ability on effort which is similar to the comparable predictive ability of "raw-function-counts and function-points" in [4]. In fact, the $R^2$ values of UFP and AFP with effort in Table 2 are comparable to the $R^2$ values of 0.75 and 0.76 in earlier research [4, p. 143].

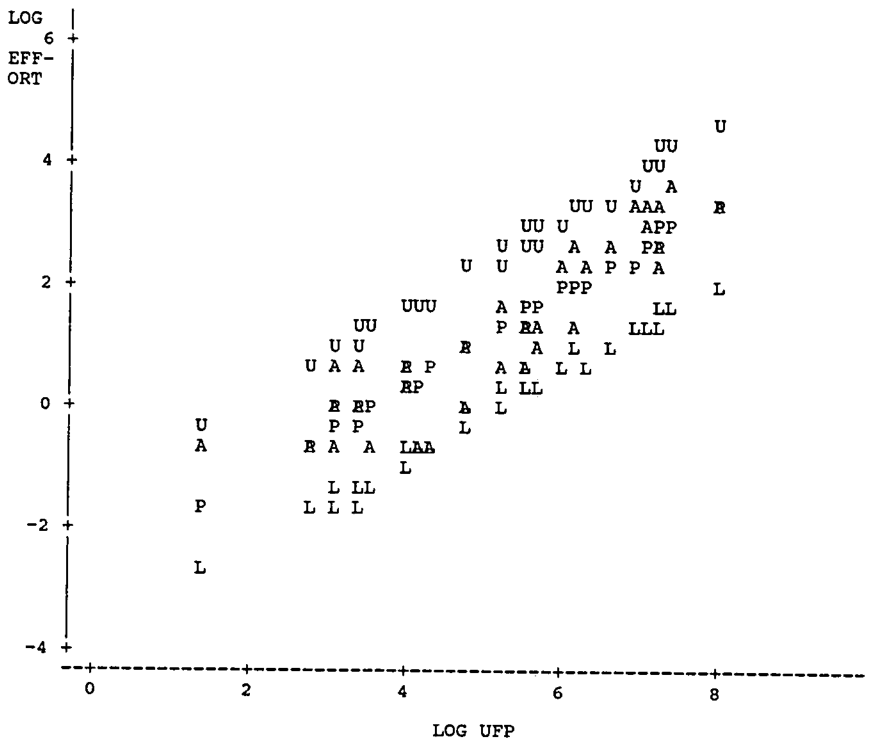

*Figure 2a.* Plot of Predicted versus Actuals for Regression of Effort versus UFP (A: actual; P: predicted; L: lower 95% confidence level; U: upper 95% confidence level)

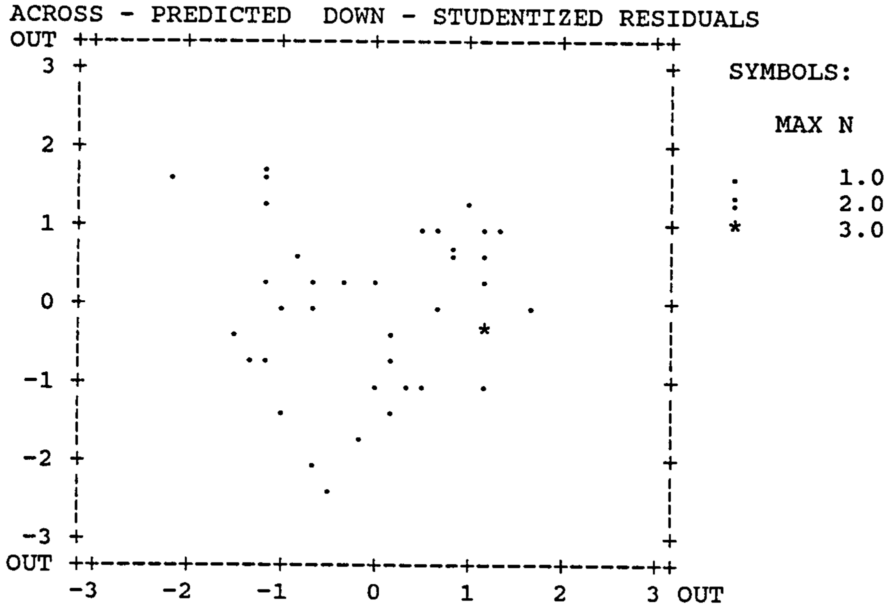

*Figure 2b.* Standardized Scatterplot for Regression of Effort versus UFP

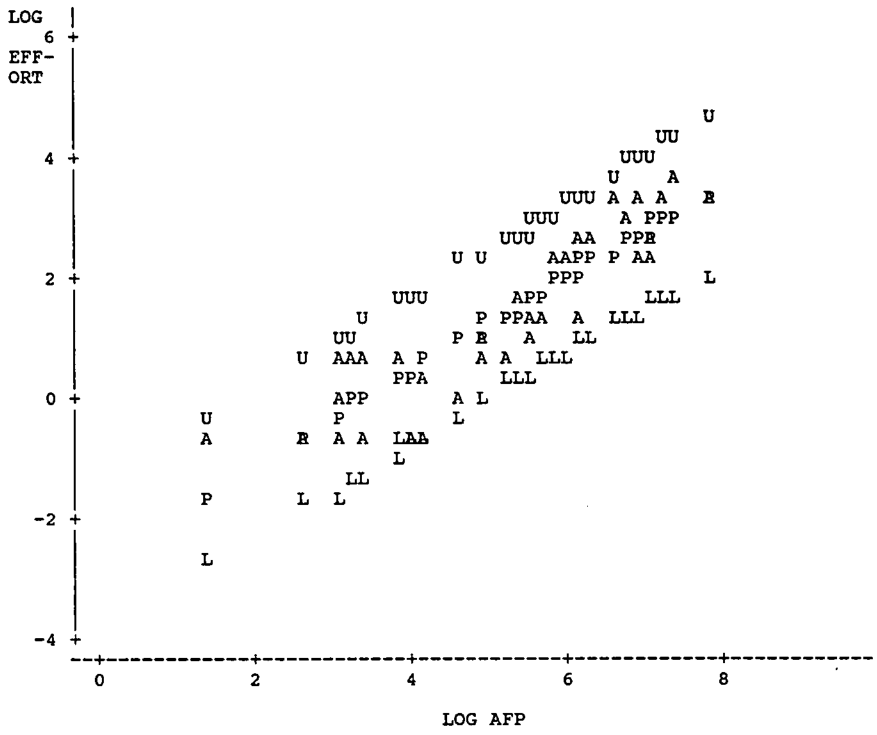

*Figure 3a.* Plot of Predicted versus Actuals for Regression of Effort versus AFP (A: actual; P: predicted; L: lower 95% confidence level; U: upper 95% confidence level)

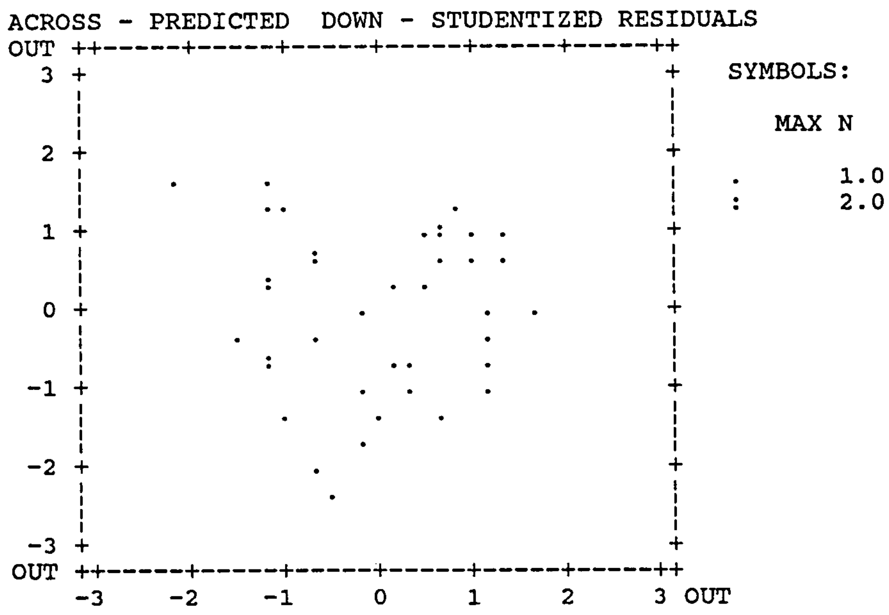

*Figure 3b.* Standardized Scatterplot for Regression of Effort versus AFP

Table 3. Regression results of Effort versus TCF and UFP versus TCF

| Dependent variable | Independent variable | $R^2$ | $F$ value | Signif. of $F$ | $T$ value | Signif. of $T$ |
|---|---|---|---|---|---|---|
| Effort | UFP | 0.7388 | 107.5 | 0.0000 | 10.37 | 0.0000 |
| | TCF | | Not entered | | 0.331 | 0.7426 |
| UFP | TCF | 0.0697 | 2.847 | 0.0997 | −1.69 | 0.0997 |
| Log(UFP) | TCF | 0.1337 | 5.866 | 0.0203 | −2.44 | 0.0203 |

Proposition 2 poses the argument that TCF could also have a direct effect on effort. Proposition 2 is tested by (a) a stepwise regression of effort with UFP and TCF, and (b) a regression of UFP and TCF. The results are summarized in Table 3.

TCF was not entered as a significant estimator of effort in the stepwise regression of effort with UFP and TCF. Hence, TCF does not have a direct effect on effort. Proposition 2 is not supported as TCF is not a significant estimator of software development effort. TCF's effect on UFP or Log(UFP) is also small, resulting in a small indirect effect of TCF on effort. The small indirect effect of TCF is also confirmed by the small increase in $R^2$ value associated with the regression of effort and AFP over the $R^2$ value of effort and UFP as shown in Table 2. A possible reason for this finding could be that TCF did not vary much among the forty projects and hence would have only a small, indirect effect on effort. It is also argued that the TCF factors may be less relevant for ICASE projects [4].

Finally, the effect of the two ICASE tools, the RAD method, and ICASE tool experience on software productivity through oneway analyses of variance (ANOVAs) are shown in Table 4 and through two factor ANOVAs are shown in Table 5. A three-factor analysis of variance with twelve cells could not be conducted with a sample size of 40.

Scheffe's $T$ test was used to test the significance of the mean differences and test propositions 3, 4, and 5. The Scheffe test results are the same for both oneway ANOVAs (Table 4) and two-factor ANOVAs (Table 5). Significance of the productivity differences between the two ICASE tools IEF and INCASE could not be confirmed as we fail to reject proposition 3. The use of RAD method for ICASE projects does result in a significant increase in software productivity in comparison to SDLC projects. This result clearly establishes the appropriateness of the RAD method in ICASE-tool-based development as it holds the key for achieving higher productivity. Finally, higher levels of software productivity are possible only as systems personnel gain more years of experience in the ICASE tool as higher levels of experience result in increased (high > medium > low) productivity. This result will help organizations develop realistic short-term and long-term expectations on the impact of ICASE tool on software productivity. As can be seen from Table 5, the interaction effects are not significant.

Table 4. One-Way ANOVA Results

| Variable | Classification and frequency | Mean | Standard deviation | *F* value | Signif. of *F* | Scheffe *T* test |
|---|---|---|---|---|---|---|
| ICASE tool | IEF (12) | 60.99 | 29.36 | 0.6600 | 0.4212 | Not significant |
| | INCASE (28) | 53.29 | 35.76 | | | |
| Systems development | SDLC (29) | 45.58 | 28.69 | 9.780 | 0.0034 | Signif. |
| | RAD (11) | 82.00 | 32.98 | | | |
| ICASE tool experience | Low (25) | 41.79 | 26.74 | 8.0600 | 0.0013 | Signif. |
| | Medium (12) | 73.34 | 34.01 | | | |
| | High (3) | 99.72 | 10.90 | | | |

Table 5. Two-Factor ANOVA Results

| Two factors | Main | Signif. of *F* | Scheffe *T* test | Interaction | Signif. of *F* |
|---|---|---|---|---|---|
| Tool and development | Tool | 0.3299 | Not signif. | Tool* development | 0.0605 |
| | development | 0.0024 | Signif. | | |
| Development and experience | Development | 0.0016 | Signif. | Development* experience | 0.3796 |
| | experience | 0.0151 | Signif. | | |
| Experience and tool | Experience | 0.0008 | Signif. | Experience* tool | 0.1619 |
| | tool | 0.2794 | Not signif. | | |

## Conclusion

SIZE IS A MAJOR ESTIMATOR OF EFFORT IN NEARLY ALL effort estimation models in non-CASE environments (e.g., COCOMO [6], ESTIMACS [27]). FP is shown to have high correlations with effort for non-CASE projects [1]. The results of this research provide additional support to the findings in [4] by showing FP to account for 73 to 82 percent of the variance in effort for forty ICASE projects in two organizations. Hence, this research and the findings from [4] add additional credibility and value to the task of keeping track of FPs for ICASE projects as they are good estimators of software development effort.

TCF, however, adds little value to the estimation of effort. TCF does not have a significant, direct effect as an estimator of effort. It has only a small, indirect effect in adjusting UFP and influencing effort. Previous empirical research for non-CASE projects has shown that TCF has only a small impact on software effort [17]. The fourteen factors that constitute TCF may be less relevant for ICASE environments [4] and hence TCF may need to be redefined.

While individual examples and claims on the favorable impact of CASE tools on software productivity exist, research on upper CASE tools did not find a significant impact on software productivity due to the lack of a disciplined approach or trained personnel [31]. ICASE tools, however, are shown to have a significant effect on software productivity [3].

The RAD method uses prototyping and small teams to effectively exploit the automated capabilities of ICASE tools to achieve rapid software development. RAD method and ICASE tools in conjunction result in productivity levels significantly greater than the productivity levels obtained from using ICASE tools and an SDLC method. It is quite possible that RAD is ideally suited for ICASE environments and results in a significant productivity advantage not possible through the use of SDLC. Further research is needed to study the advantages of using RAD in ICASE environments.

Use of an ICASE tool alone would not ensure high software productivity. Trained personnel are crucial for reaping the benefits of CASE tools [31]. This research has shown that there are significant productivity differences between different levels (low < medium < high) of ICASE tool experience. Low ICASE experience results in a productivity of forty-two FPs per person-month. High ICASE experience, with more than twice the experience of low ICASE, results in ninety-nine FPs per person-month which is more than twice the low ICASE productivity. Organizations need to have realistic short-term and long-term productivity expectations from ICASE tools. Similarly, CASE tool vendors need to emphasize in their claims of high productivity from ICASE tools that these productivity levels are possible only as software personnel gain years of experience in using the ICASE tool.

## REFERENCES

1. Albrecht, A.J., and Gaffney, J.E. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions of Software engineering, 9*, 6 (November 1983), 639–648.

2. August, J.H. *Joint Application Design: The Group Session Approach to Systems Design*. Englewood Cliffs, NJ: Yourdon Press, 1991.

3. Banker, R.D., and Kauffman, R.J. Reuse and productivity in integrated computer-aided software engineering: an empirical study. *MIS Quarterly, 15*, 3 (September 1991), 375–401.

4. Banker, R.D.; Kauffman, R.J.; and Kumar, R. An empirical test of object-based output measurement metrics in a computer aided software engineering (CASE) environment. *Journal of Management Information Systems, 8*, 3 (Winter 1991–92), 127–150.

5. Baum, D. Go totally RAD and build apps faster. *Datamation, 38*, 19 (September 15, 1992), 79–81.

6. Boehm, B.W. Software engineering economics. *IEEE Transactions on Software Engineering, 10*, 1 (January 1984), 4–21.

7. Carmel, E.; Whitaker, R.; and George, J.F. PD and joint application design: a transatlantic comparison. *Communications of the ACM, 36*, 6 (June 1993), 40–48.

8. *EDP Analyzer*. Developing high quality systems faster. *24*, 6 (June 1986), 1.

9. Electronic Data Systems. Analysis of system development time using INCASE—rapid application development. Technical Report, Plano, TX, 1992.

10. Electronic Data Systems. Introduction to integrated CASE. Technical Report, Plano, TX, 1992.

11. Electronic Data Systems. INCASE 11.0—product overview. Technical Report, Plano, TX, 1992.

12. Fersko-Weiss, H. CASE tools for designing your applications. *PC Magazine, 9*, 2 (January 30, 1990), 213–251.

13. Foss, W.B. Fast, faster, fastest development. *Computerworld, 27*, 22 (May 31, 1993), 81–83.

14. Gremillion, L.L., and Pyburn, P. Breaking the systems development bottleneck. *Harvard Business Review, 61*, 2 (March–April 1983), 130–137.

15. Hyman, R. Creative chaos in high-performance teams: an experience report. *Communications of the ACM, 36,* 10 (October 1993), 56–61.
16. Jones, C. *Programming Productivity.* New York: McGraw-Hill, 1986.
17. Kemerer, C.F. An empirical validation of software cost estimation models. *Communications of the ACM, 30,* 5 (May 1987), 416–429.
18. Kemerer, C.F. Reliability of function points measurement: a field experiment. *Communications of the ACM, 36,* 2 (February 1993), 85–97.
19. Kemerer, C.F., and Porter, B.S. Improving the reliability of function point measurement: an empirical study. *IEEE Transactions on Software Engineering, 18,* 11 (November 1992), 1011–1024.
20. Keyes, J. Gather a baseline to assess CASE impact. *Software Magazine, 10,* 10 (August 1990), 30–43.
21. Martin, J. *Rapid Application Development.* New York: Macmillan, 1991.
22. Matson, J.E.; Barrett, B.E.; and Mellichamp, J.M. Software development cost estimation using function points. *IEEE Transactions on Software Engineering, 20,* 4 (April 1994), 275–287.
23. Mimno, P. RAD: the enabling technologies. *CASE Trends, 2,* 3 (May–June 1991), 20–25.
24. Necco, C.; Tsai, N.W.; and Holgeson, K.W. Current usage of CASE software. *Journal of Systems Management, 40,* 5 (May 1989), 33–37.
25. Norman, R.J., and Nunamaker, J.F. CASE productivity perceptions of software engineering professionals. *Communications of the ACM, 32,* 9 (September 1989), 1102–1108.
26. Rettig, M., and Simmons, G. A project planning and development process for small teams. *Communications of the ACM, 36,* 10 (October 1993), 44–57.
27. Rubin, H.A. Macroestimation of software development parameters: the ESTIMACS systems. In *SOFTAIR Conference on Software Development Tools, Techniques, and Alternatives,* Arlington, VA. New York: IEEE Press, July 1983, pp. 109–118.
28. Rush, G. A fast way to define system requirements. *Computerworld, 19,* 40 (October 7, 1985), 11–12.
29. Swanson, K.; McComb, D.; Smith, J.; and McCubbrey, D. The application software factory: applying total quality techniques to systems development. *MIS Quarterly, 15,* 4 (December 1991), 567–579.
30. Texas Instruments. IEF technical description—methodology and technical overview. Technical Report, Dallas, TX, 1992.
31. Vessey, I.; Jarvenpaa, S.L.; and Tractinsky, N. Evaluation of vendor products: CASE tools as methodology companions. *Communications of the ACM, 35,* 4 (April 1992), 90–105.
32. Whitaker, R.; Essler U.; and Oestberg, O. Participatory business modeling. Lulea Technical University (Sweden) Report, 1991.
33. Walz, D.B.; Elam, J.J.; and Curtis, B. Inside a software design team: knowledge acquisition, sharing, and integration. *Communications of the ACM, 36,* 10 (October 1993), 62–77.
34. Wood, J., and Silver, D. *Joint Application Design: How to Design Quality Systems in 40% Less Time.* New York: Wiley, 1989.
35. Wrigley, C.D., and Dexter, A.S. A model for measuring information system size. *MIS Quarterly, 15,* 2 (June 1991), 245–257.
36. Yourdon, E. *The Decline and Fall of the American Programmer.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
37. Zultner, R.E. TQM for technical teams. *Communications of the ACM, 36,* 10 (October 1993), 78–91.

## APPENDIX A: Function Points Calculation with an Example

### A. Count and Classify Five User Function Types.

1. External Input Types

- Transactions from the user;
- Transactions from other applications;
- Data and Control Input Types;

Classification:

| | |
|---|---|
| Low | Few data-element types. |
| | Few internal-file types referenced. |
| | Human factoring not a major concern of the input type. |
| High | Many data-element types. |
| | Many internal-file types referenced. |
| | Human factoring considerations significantly affect design of input type. |
| Medium | Between high and low. |

2. External Output Types
- Transactions to the user;
- Transactions to other applications;
- Data and Control Output Types;

Classification:

| | |
|---|---|
| Low | Few data-element types. |
| | Few internal-file types referenced. |
| | Human factoring not a major concern of the output type. |
| High | Many data-element types. |
| | Many internal-file types referenced. |
| | Human factoring considerations significantly affect design of output type. |
| Medium | Between high and low. |

For Reports:

| | |
|---|---|
| Low | One or two columns. |
| | Simple data transformation. |
| High | Multiple and intricate data-elements. |
| | Multiple and complex file references. |
| Medium | Multiple columns with subtotals. |
| | Multiple data-element transformations. |

3. Logical Internal File Types (Master Files)
- Each logical file within a database;
- Each logical group of data from the viewpoint of the user;

Classification:

| | |
|---|---|
| Low | Few record types. |
| | Few data-element types. |
| | No significant performance or recovery considerations. |
| High | Many record types. |
| | Many data-element types. |
| | Performance and recovery are significant considerations. |
| Medium | Between high and low. |

4. External File Types (Interfaces)
- Each Input File Type;

- Each Output File Type;
- Data and Control Information;

Classification:

| | |
|---|---|
| Low | Few record types. |
| | Few data-element types. |
| | No significant performance or recovery considerations. |
| High | Many record types. |
| | Many data-element types. |
| | Performance and recovery are significant considerations. |
| Medium | Between High and Low. |

5. Query Types

- Queries from the user;
- Queries from other applications;

Classification:

| | |
|---|---|
| Low | Few data-element types. |
| | Few internal-file types referenced. |
| | Human factoring is not a major concern. |
| High | Many data-element types. |
| | Many internal file-types are referenced. |
| | Human-factoring considerations significantly affect the design of the query. |
| Medium | Between high and low. |

For Reports:

| | |
|---|---|
| Low | One or two columns. |
| | Simple data transformation. |
| High | Multiple and intricate data transformations. |
| | Multiple and complex file references. |
| Medium | Multiple columns with subtotals. |
| | Multiple data-element transformations. |

## B. Calculate Unadjusted Function Points (UFP)

| | Low | | Medium | | High | | Total |
|---|---|---|---|---|---|---|---|
| External input | $X$ * 3 | + | $X$ * 4 | + | $X$ * 6 | = | 999999 |
| External output | $X$ * 4 | + | $X$ * 5 | + | $X$ * 7 | = | 999999 |
| Log. internal file | $X$ * 7 | + | $X$ * 10 | + | $X$ * 15 | = | 999999 |
| External file | $X$ * 5 | + | $X$ * 7 | + | $X$ * 10 | = | 999999 |
| Query | $X$ * 3 | + | $X$ * 4 | + | $X$ * 6 | = | 999999 |
| Total unadjusted function points: | | | | | | | 999999 |

## C. Calculate Technical Complexity Factor (TCF)

Fourteen technical factors: data communications, online update distributed functions, complex processing, performance, reusability, heavily used configuration, installation ease, transaction rate, operational ease, online data entry, multiple sites, end-user efficiency, and facilitate change.

Degree of Influence Factor Weights:

| | |
|---|---|
| 0 | Factor not present |
| 1 | Insignificant influence |
| 2 | Moderate influence |
| 3 | Average influence |
| 4 | Significant influence |
| 5 | Strong influence throughout |

TCF = 0.65 + (0.01 * sum of degree of influence factor weights for the fourteen factors).

## D. Calculate Adjusted Function Points (AFP)

AFP = UFP * TCF.

## An Example for Calculating Function Points: Truck and Bus/Request Tracking System.

*A, B. Count and classify user types; calculate unadjusted function points.*

| Project | Low | | Medium | | High | | Total |
|---|---|---|---|---|---|---|---|
| External input | 0 * 3 | + | 1 * 4 | + | 3 * 6 | = | 22 |
| External output | 0 * 4 | + | 0 * 5 | + | 1 * 7 | = | 7 |
| Log. internal file | 0 * 7 | + | 0 * 10 | + | 1 * 15 | = | 15 |
| External file | 0 * 5 | + | 1 * 7 | + | 1 * 10 | = | 17 |
| Query | 3 * 3 | + | 8 * 4 | + | 3 * 6 | = | 59 |
| Total unadjusted function points: | | | | | | | 120 |

*C. Calculate TCF.*

| | | | |
|---|---|---|---|
| Data communications | 4 | Online update | 3 |
| Distributed functions | 4 | Complex processing | 3 |
| Performance | 3 | Reusability | 3 |
| Heavily used configuration | 4 | Installation ease | 2 |
| Transaction rate | 3 | Operational ease | 4 |
| Online data entry | 2 | Multiple sites | 4 |
| End-user efficiency | 3 | Facilitate change | 1 |

TCF = 0.65 + (0.01 * sum of degree of influence factor weights); TCF = 0.65 + 0.01 * 43 = 1.08.

*D. Calculate adjusted function points (AFP).*

AFP = UFP * TCF = 120 * 1.08 = 129.6.

## APPENDIX B: Description of ICASE Tools Used in Research Study

This research investigation chose software development projects that used the Information Engineering Facility (IEF) integrated CASE tool from Texas Instruments or the Integrated CASE (INCASE) from Electronic Data Systems. The IEF software provides automated support through a set of tightly integrated tools. The IEF product supports software development through planning, analysis, design, and construction toolsets. IEF is widely recognized by industry analysts as one of the leading application development solutions on the market today. Texas Instruments can claim as users of IEF companies such as Coca-Cola, Sony, American Airlines, and EDS. Testimonials of the IEF product have been received from Xerox for the development of a logistics system, and Canadian Airlines for their frequent flier program software.

The Information Engineering Facility (IEF) toolsets are available on a variety of workstations and can generate systems to run on multiple platforms ranging from workstations to mainframes. IEF generates applications based on analysis and design level diagrams allowing users to separate themselves from the specifics of the computer environment. For example, an analyst does not necessarily have to know the programming language "C" to generate and port an application to a customer's machine. As business or technological conditions change, users can easily redistribute tasks or regenerate software for new platforms. Another benefit of IEF is the ability to clearly define relationships between diagrams and objects. IEF provides the ability of the software to enforce integrity constraints on the developing system. IEF looks at a business model from five different layers.

1. The architectural layer is a high-level, broad view of the organization to be automated. It consists of three interlocking architectures that form a high-level blueprint for meeting the organization's goals and objectives.
2. The conceptual layer is a model of the data relationships, activities, and business rules on which the systems will be based. This model subdivides the concepts introduced at the first layer into subordinate pieces and describes them in much more detail.
3. The external layer is a model of system behavior as experienced by an end user of the system. It contains specialized information about the conceptual layer of particular interest to users, such as screen layouts and function key assignments.
4. The implementation layer is a specialization of the external layer. It maps the external details about the system to a specific computer hardware and software environment.
5. The last level, the execution layer consists of data bases and applications executing on specific computing equipment. This layer becomes the reality of the model that stands above it [30].

The importance of the ICASE tool as a strategic advantage in application software development has been recognized by EDS corporate executives. This can be exemplified through the alliance with Amdahl Corporation to form the EDS/Anteres

Alliance. Previously, the INCASE product suffered from a corporate neglect that had allowed the product to fall behind its primary competitor IEF. The renewed emphasis on updating the product and making it competitive displays the importance of ICASE product for EDS in achieving and maintaining a strategic advantage [9, 10, 11].

The technical similarities between the two products (IEF and INCASE) are remarkable. As with IEF, INCASE also provides for a multilevel approach to application software development. The primary difference between the products is that at the time of this writing, INCASE does not support a graphical user interface (GUI) capability. This restriction severely handicaps the INCASE product as nearly all user-friendly environments now incorporate GUI in application software development. This GUI feature is currently under development within the EDS/Anteres Alliance. The primary features of the INCASE product are the full system life cycle coverage, reusable code hierarchy, and its integrated documentation facility. The INCASE software runs on a high-end 486 PC or Sun Sparc10 workstation under the Unix operating system. Generated code and applications can be ported to numerous environments as with IEF. Provided all necessary system networking has been set up, multiple users can have the ability to work within the same project in the ICASE environment. The experiences of the INCASE tool have been primarily contained within internal EDS corporate applications, and General Motors applications (EDS's parent corporation). Applications range from inventory systems to corporate job posting and training systems.