Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

An Empirical Study of Learning Curve Theory's application to the Software Development Effort-Experience Relationship in Software Engineering and Project Management

Parag Pendharkar
Penn State Harrisburg

Girish Subramanian
Penn State Harrisburg

An Empirical Study of Learning Curve Theory's application to the Software Development Effort-Experience Relationship in Software Engineering and Project Management

Parag C. Pendharkar
School of Business, Penn State Harrisburg
pxp19@psu.edu

Girish H. Subramanian
School of Business, Penn State Harrisburg
ghs2@psu.edu

ABSTRACT

The experience of software developers in a given tool is often perceived as an endogenous factor in software cost estimation. Many manufacturing applications have modeled resources devoted to output production, and worker's knowledge as learning curves. In this paper, we use and test the learning curve theory in software industry. We illustrate that a programmer's effort exponentially decreases as his/her tool experience increases. After certain amount of tool-experience, the effort reduction asymptotically stabilizes. We believe that the learning curve theory and its application provide an excellent opportunity for IT project managers to better utilize the experience of their software personnel.

Keywords

Learning Curves, Software Effort, CASE Tools.

INTRODUCTION

We examine the relationship between a programmer experience in an Integrated Computer-Aided Software Engineering (ICASE) tool and the software development effort. In our study, we hypothesize that the software development effort will reduce exponentially as programmers gain more experience in the usage of ICASE tools. Further, the exponential reduction in development effort may be different for different ICASE tools.

Evidence for exponential learning curve is found in several software engineering applications. Laranjeira (1990) and Henderson-Sellers (1997) note that relative cost range for object-oriented projects decrease exponentially with increase in life cycle phase. Boehm (1981) observed the exponential decreasing relationship between the uncertainty in software project cost estimates and the life cycle phase of the product. The existence of learning curves in using ICASE tool was conjectured in Chew et al. (1991) and Kemerer (1992).

The *learning curve* theory was first introduced in aircraft industry, where it was found that as workers gained experience in the production of airplanes the time required producing the airplanes decreased (Finch and Luebbe, 1991). A learning curve expresses the decreasing time it takes to perform a repetitive task as one continues from one cycle to another cycle of the task (Fauber, 1989). Nicks (1982) argue that, as quantity of product doubles costs are reduced by learning at some steady rate called the "learning rate." According to Conway and Schultz (1959), learning rate depends on two factors, pre-production and actual production. Finch and Luebbe (1991) investigated the impact of learning rate and its variability on the project length. Kevin Self (1995) reports that there are two types of learning curves.

LITERATURE REVIEW AND HYPOTHESES

Software development and maintenance represents significant expense to the organization (Banker and Slaughter, 1997). The software development for an organization involves the risk of delays in implementation and inability of an organization to abandon a troubled project (Pendharkar et al., 1998).

There are four essential aspects of software development -- the tools, methodology, people, and management (Foss, 1993; Subramanian and Zarnich, 1996). Empirical research presents a systematic productivity measurement model for CASE tools to aid organizations to assess the benefit of CASE tools (Banker and Kauffman, 1991). Tools, methods, and personnel

experience are indicated as variables that influence software development effort in the systems development effort model (Wrigley and Dexter, 1991).

The use of CASE has been shown to increase productivity levels measured in function points delivered per man-month for First Boston Bank (Banker and Kauffman, 1991). Other studies also report productivity improvements through the use of ICASE tools (Necco et al., 1989; Norman and Nunamaker, 1989; Swanson et al., 1991). ICASE tools that support the disciplined approach have a better chance at increasing productivity levels (Vessey et al., 1992).

Poor software productivity is attributed to the absence of ICASE tool training for systems personnel (Yourdon, 1988). ICASE users often experience a productivity decrease for the first 3 to 6 months, and it often takes 12 to 18 months before productivity gains are visible (Keyes, 1990). Kemerer (1992) conjecturing the existence of learning curve in ICASE tools wrote "Integrated CASE tools have raised the stakes of the learning issue. Because these tools cover the entire life cycle, there is more to learn, and therefore the study of learning – and the learning-curve phenomenon – is becoming especially relevant."

The exact relationship between tool experience and development effort is not formally established. Given that exponential decreasing relationships have been observed in related software engineering applications (Boehm, 1981; Laranjeira, 1990; Henderson-Sellers, 1997) of learning curves, it can be argued that the relationship between development effort and tool experience is an inverse exponential relationship given by the following generalized mathematical formulation:

$$y = a_1 e^{-a_2 x}.$$

Where, the variables y and x represent effort and tool experience respectively. The constants a_1 and a_2 are constants that are related to the type of tool and learning rate respectively.

So, we hypothesize:

Hypothesis1a: Increase in tool experience results in an exponential decrease of software effort.

Hypothesis 1b: The tool experience-software effort relationship can be expressed in exponential decay form as $y = a_1 e^{-a_2 x}$.

RESEARCH STUDY AND RESULTS

Texas Instruments (TI) CASE tool IEF and Electronic Data Systems (EDS) CASE tool INCASE are the two CASE tools examined in this study. The projects in the data set reported in Subramanian and Zarnich (1996) used Rapid Application Development (RAD) and the traditional Systems Development Life Cycle (SDLC) development. The RAD method (Martin, 1991) is claimed to give much faster development and higher quality results. The IEF and INCASE tools can be used to support a wide range of software development projects. We use the software project data from Subramanian & Zarnich (1996) study to test the hypothesized nonlinear relationship.

In order to use Subramanian and Zarnich (1996) data to estimate the learning curve between programmer's ICASE tool experience and software development effort, we make a following assumption.

Assumption: The programmers' learning curves between their ICASE tool experiences and software development effort are independent and identically distributed (iid).

The iid assumption allows us to use cross-sectional data for estimating learning curves. Further, real life dynamics does not have any impact on the functional form of the learning curves.

Assuming iid learning curves, we learn the relationship between development effort and ICASE tool experience without assuming any *apriori* functional form. We use artificial neural networks to learn the relationship from the training data. According to the following proposition, neural networks can learn any function to any level of accuracy, and the function learned by neural network may not be considered to have any researcher bias.

Proposition: An artificial neural network using sigmoid transfer function with n neurons in the input layer and one neuron in the output layer can approximate any continuous function $f: \Re^n \rightarrow \Re$ in any domain with any given accuracy.

Proof: The reader is directed to Tromp (1993) for full proof of this proposition.

Although a neural network can learn any function to any level of accuracy, there may not be a way of knowing the type of function learnt by the neural network if all the inputs are continuous. However, Pendharkar and Subramanian (1999) illustrated that if the input vector contains exactly one continuous input, several other categorical inputs, and one continuous output variable then the exact function formulation for the continuous input and continuous output can be known.

Since our data set contains exactly one continuous input, two categorical variables and one continuous output variable, the exact function between continuous input and output variable can be elicited using a neural network. In order to learn the non-linear effort-experience function, we use a three layer neural network with three input nodes, seven hidden layer nodes and one output node. The number of hidden layer nodes was selected based on popular heuristic of $2n+1$ (where n is number of input nodes) (Bhattacharyya and Pendharkar, 1998). The three inputs were software development methodology (RAD =1, SDLC = 0), Tool used (IEF = 1, INCASE =0) and the tool experience in years. We chose software effort in man months as our output variable. We used all the 40 projects for learning the connection weights in a neural network. The 40 projects sample size for learning was adequate as the heuristic for learning sample size is 10 times the number of input variables (Bhattacharyya and Pendharkar, 1998). After training the neural network, we used an experimental design to systematically mine the patterns that the neural network had learnt.

We perform two experiments to identify effort-experience functions for IEF CASE tool and INCASE CASE tool respectively. In each of these two experiments, we fixed the first two inputs (methodology and tool) and varied the tool experience from its initial value of 0 to 0.99 in the intervals of 0.03. Our first experiment used SDLC methodology (methodology=0) and IEF tool (CASE=1) for the two fixed input values. After fixing the two inputs and varying the third input, we obtained the software effort values generated by the trained network. Figure 1 illustrates the graph of the software effort generated by the trained neural network for the different input values of tool experience. As shown in Figure 1 with increase in the tool experience the software effort decreases.

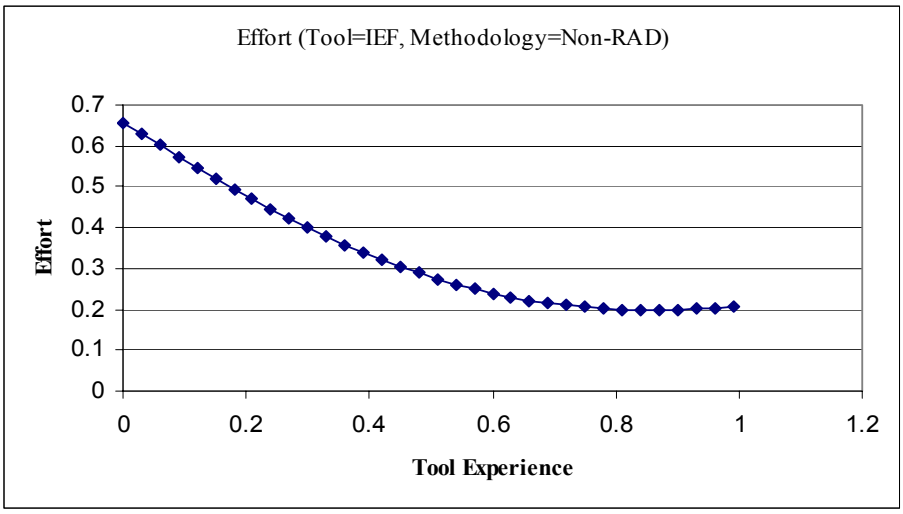

Figure 1: Software Effort vs. IEF Tool Experience for Non-Rapid Application Design

For our second experiment, we kept the first two inputs fixed at RAD methodology (methodology= 1) and INCASE tool (CASE=0) and varied the tool experience systematically. Figure 2 illustrates the results of our second experiment. Figures 1 and 2 illustrate the non-linear patterns that were learned by the neural network. It can be seen that both in Figure 1 and 2, higher tool experience leads to decrease in the software effort.

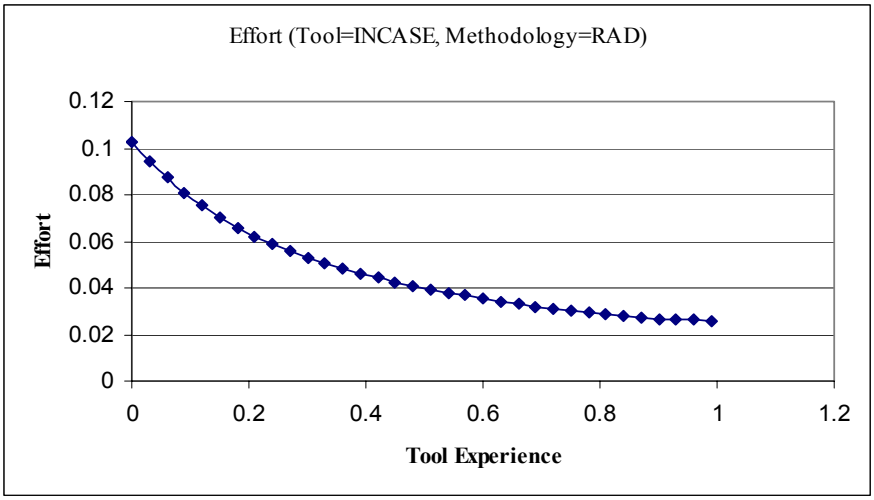

Figure 2: Software Effort vs. INCASE Tool Experience for Rapid Application Design

After an initial evidence of existence of learning curve, we specifically test for the existence of hypothesized relationship. In other words, we test if increase in tool experience decreases the software effort exponentially and if the relationship can be represented, in exponential decay form, as

$effort = a_1 e^{-experience*a_2}$.

Using a non-linear least square regression, we test our hypothesis for IEF and non-RAD, and INCASE tool and RAD methodology. Tables 1 through 4 show the results of our non-linear regression tests for the two experiments respectively.

Source	DF	Sum of Squares	Mean Square	F Value	Prob (F)
Regression	1	0.6927	0.6927	1341.93	0.00001**
Error	32	0.0165	0.0005		
Total	33				

Table 1: The non-linear regression analysis results (IEF-NonRAD)

R^2=0.9767; **Significant at α=0.01

1.1.1.1 Parameter	Estimate	Std. Error	t-value	Prob (t)
a_1	0.6436	0.0102	62.64	0.00001**
a_2	1.5113	0.0445	33.92	0.00001**

Table 2: The estimates for the constants and the t-test results (IEF-nonRAD)

**Significant at α=0.01

Source	DF	Sum of Squares	Mean Square	F Value	Prob (F)
Regression	1	0.0145	0.0145	911.23	0.00001**
Error	32	0.0005	0.0000		
Total	33				

Table 3: The non-linear regression analysis results (INCASE-RAD)

R^2=0.9650; **Significant at α=0.01

1.1.1.2 Parameter	Estimate	Std. Error	t-value	Prob (t)
a_1	0.0923	0.0018	50.54	0.00001**
a_2	1.5113	0.0567	27.99	0.00001**

Table 4: The estimates for the constants and the t-test results (INCASE-RAD)

**Significant at α=0.01

Scenario	*1.1.1.2.1 Model Learnt*	Differential Equation
IEF and SDLC	$Effort=0.69e^{-1.51*experience}$	$d(effort)/dexp.= -0.69\ exp.$
INCASE and RAD	$Effort=0.09e^{-1.51*experience}$	$d(effort)/dexp.= -0.09\ exp.$

Table 5: The summary of the functions learnt by the ANN

Table 5 summarizes the results of our experiments. The results indicate support for our hypothesis.

CONCLUSION

We have shown that learning curve theory can be used to better understand the experience-effort relationship. The general form of learning curve is exponential, and managers, for a given ICASE tool, can use the historical data to estimate the parameters and degree of the learning curves. Parameters of the learning curve, when known, can be used to develop reliable future impact of experience on effort. Thus, our approach can apply to practice and help project managers.

REFERENCES

1. Banker, R. D., and Slaughter, S. A. (1997). A field study of scale economies in software maintenance, *Management Science*, 43, 12, 1709-1725.
2. Banker, R.D., and Kauffman, R. J. (1991). Reuse and productivity in integrated computer-aided software engineering : An empirical study, *MIS Quarterly*. 15, 3, 375-401.
3. Bhattacharyya, S., and Pendharkar, P. C. (1998). Inductive, evolutionary and neural techniques for discrimination: A comparative study, *Decision Sciences*, 29, 4, 1998, 871-899.
4. Boehm, B. (1981). *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall.
5. Conway, R. W. and Schultz, A. (1959). The manufacturing progress function, *Journal of Industrial Engineering*, January-February, 39-54.
6. Fauber, C. E. (1989) Use of improvement (learning) curves to predict learning costs, *Production and Inventory Management Journal*. Third Quarter, 57-60.

7. Finch, B. J., and Luebbe, R. L. (1991). Risk associated with learning curve estimates, *Production and Inventory Management Journal*. Third Quarter, 73-76.
8. Foss, B. W. (1993). Fast, faster, fastest development, *Computerworld*, 27, 22, 81-83.
9. Henderson-Sellers, B. (1997). Corrigenda: Software Size Estimation of Object-Oriented Systems, *IEEE Transactions on Software Engineering,* 23, 4, 260-261.
10. Kemerer, C. F. (1992). How the learning curve affects CASE tool adoption, *IEEE Software*, 9, 5, 23-28.
11. Keyes, J. (1990). Gather a baseline to assess CASE impact, *Software Magazine*, 10, 1990, 30-43.
12. Laranjeira, L. A. (1990). Software Size Estimation of Object-Oriented Systems, *IEEE Transactions on Software Engineering*, 16, 5, 510-522.
13. Martin, J. (1991). *Rapid Application Development*, Macmillan Publishing Co.
14. Necco, C., Tsai, N. W. and Holgeson, K. W. (1989). Current usage of CASE software, *Journal of Systems Management*, 40, 5, 33-37.
15. Nicks, J. E. (1982). *Basic Programming Solutions of Manufacturing*. Society of Manufacturing Engineers, Dearborn, MI.
16. Norman, R. J., and Nunamaker, J. F. (1989). CASE productivity perceptions of software engineering professionals, *Communications of the ACM*, 32, 9, 1102-1108.
17. Pendharkar, P.C. and Subramanian, G. H. (1999). Mining software effort estimation functions using certain neural networks, *Proceedings of 4th INFORMS Conference on Information Systems and Technology*, 220-234.
18. Pendharkar, P.C., Rodger, J. A. and Kumar, S. (1998). Operational auditing and testing in client/server systems, *Review of Accounting Information Systems*, 2, 1, 17-22.
19. Subramanian, G. H. and Zarnich, G. E. (1996). An examination of some software development effort and productivity determinants in ICASE tool projects, *Journal of Management Information Systems,* 12, 4, 143-160.
20. Self, K. (1995). Learning curve boomerangs, *IEEE Spectrum*, December, 17-17.
21. Swanson, K., McComb, D., Smith, J., and McCubbrey, J. "The application software factory: Applying total quality techniques to systems development," *MIS Quarterly*, 15, 4, 567-579.
22. Tromp, E. (1993). Practical implications of a theoretical analysis of the behaviour of a multi-layer neural network, *Report No. 93N063*, University of Twente, Department of Engineering.
23. Vessey, I., Jarvenpaa, S. L. and Tractinsky, N. (1992). Evaluation of vendor products: CASE tools as methodology companions, *Communications of the ACM*, 35, 4, 90-105.
24. Wrigley, C. D., and Dexter, A. S. (1991). A model for measuring information system size, *MIS Quarterly*, 15, 2, 245-257.
25. Yourdon, E. (1988). *The Decline and Fall of the American Programmer*, Englewood Cliffs, NJ: Prentice-Hall.