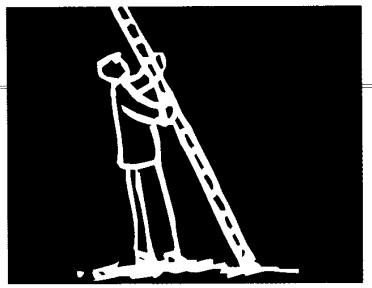


How the Learning Curve Affects CASE Tool Adoption

CHRIS F. KEMERER, *MIT Sloan School of Management*

◆ *Why do some organizations buy integrated CASE tools only to leave them on the shelf? Part of the answer may lie in a misinterpretation of the learning curve and its affect on productivity.*

With the rising cost of software development, tools for integrated computer-aided software engineering offer solutions to productivity and quality problems that plague the profession. But while most software developers accept the idea that integrated CASE can help lower costs and increase productivity, the state of practice is less optimistic. Organizations tend to adopt integrated CASE only in a limited form or they abandon a good percentage of the technology soon after it is implemented.

One study shows that one year after introduction, 70 percent of CASE tools and techniques are never used, 25 percent are used by only one group, and five percent are widely used, but not to capacity. In a different survey of more than 200 leading organizations, less than 25 percent of the staff were using front-end CASE tools. In another survey of 63 leading organizations, only 24 percent were using CASE at all. Another study reports that one organization is not using 80 to 90 percent of the CASE tool packages it purchased.[1]

Yet there is an obvious need for such tools. The already high demand for software continues to grow, and there is a shortage of qualified software developers. Indeed, one cause of quality shortfalls in delivered software could very well be the participation of marginally qualified individuals in its development. So with this relatively scarce supply of software-development labor, it makes good sense to substitute development capital in the form of CASE tools. Some think of this as software development's favorable evolution from a craft-type activity to one more closely resembling an engineering or manufacturing operation.[2]

So why aren't organizations embracing the idea of integrated CASE in more than just theory? One problem is that the first

0740-7459/92/0500/0023/$03.00 © IEEE

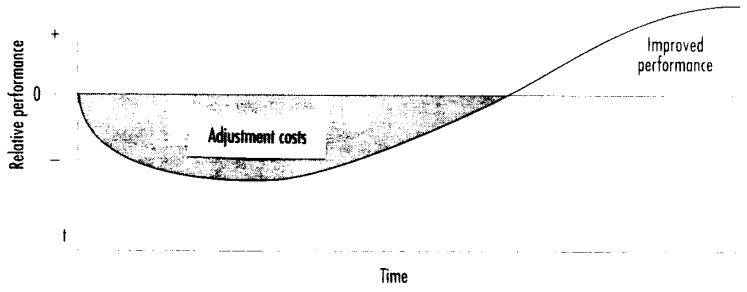


Figure 1. Performance over time with learning effect.

project written with an integrated CASE tool typically fails to deliver improved results. Academicians and practitioners say that the learning curve, described in the box on p. 26, can partially explain this phenomenon. But merely identifying learning as a source of the problem is not enough. Managers need more information to justify their investment in CASE technology. They need a way to predict the extent of the learning curve and data to estimate its parameters so that they can determine their return on investment or similar measures for CASE tool adoption.

Ultimately, knowing the factors that favorably influence the rate of learning, not merely the observed learning rate, will be what managers find the most useful. But a necessary first step is the ability to measure the current state of the process.

INFLUENCE OF LEARNING CURVES

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.

One interpretation of the learning curve is that initial projects are relatively more expensive than later projects and are even likely to be more expensive than projects produced under the old technology. In a survey of more than 60 sites, W. Bruce Chew and colleagues translated this effect into the *S* curve in Figure 1.[3] The curve, which represents actual performance, dips below 0 on the relative performance scale, indicating that performance on initial projects with the new technology is worse than performance on projects with the old technology. This effect eventually wears off, but it is not what adopters of process innovations usually expect. They often project a zero increase in performance followed by a rise that eventually plateaus. Thus, they are disappointed when expected benefits do not materialize as soon as planned and may abandon the technology before realizing any net benefit.

The problem for CASE tool adopters is that no one agrees on how the learning curve is likely to affect tool adoption.

Adding to the confusion is that CASE tools are relatively new, and there is no published data on the learning-curve effect — although a number of observers have postulated a model like Figure 1. The problem for CASE tool adopters is that no one agrees on how the learning curve is likely to affect tool adoption. For example, a model from Software Productivity Research predicts that the *S* curve for CASE tools crosses the 0 relative productivity level in about six months. A near identical graph from the Gartner Group shows the *S* curve for integrated CASE tools crossing after more than a year. A mix and match of vendor solutions reach the same level only after more than two years. On the other extreme is a survey of CASE users by CASE Research Corp., which found that more than one-third of all back-end (lower) CASE users claimed full proficiency in only one to two months. Finally, a report in *CASE Outlook*, while noting the absence of quantitative studies, nonetheless offers the following prediction: "... plan on a productivity reduction of 50 percent for six months, parity for the next six months, and 30 percent to 50 percent improvement thereafter [including tool-specific learning-curve effects]."[1]

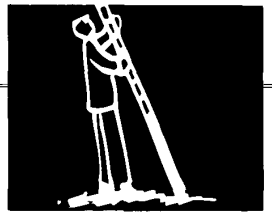
Clearly, the industry has only contradictory evidence to provide CASE tool adopters — which may be part of the reason the tools aren't being adopted very quickly. (The difference may also be because time is chosen as the model's axis instead of projects, which more accurately reflects how learning occurs.) This view of learning emphasizes costs, rather than the traditional learning benefits, as the box on p. 26 describes. The goal of this article is to go beyond that view and show how learning-curve models can help managers in adopting integrated CASE tools.

ADAPTING MODELS TO INTEGRATED CASE

Although there are many learning-curve models, it is not easy to adapt them to estimating learning curves for integrated CASE tools. A number of issues — both theoretical and those having to do with implementation — present formidable obstacles to using traditional models, which were created to predict the performance of manual workers performing repetitive tasks. Users of integrated CASE tools are essentially knowledge workers performing tasks, that (at least at first approximation) are not so repetitive.

Theoretical issues. These issues include sensitivities peculiar to knowledge work, the diversity of tasks, and the confusion between tool learning curves and learning curves for supporting technologies.

Knowledge-work sensitivities. Although all the original applications of learning curves involved manual tasks, there was no reason to believe similar effects would not be found in knowledge work — tasks like system design and analysis. Such effects are



arguably stronger for tasks that are not physically constrained. Air-frame construction and printed-circuit-board assembly, for example, ultimately encounter physical constraints such as maximum speed of operation, especially if safety is a priority. So a natural assumption is because knowledge work does not have these binding constraints, traditional learning-curve models will apply at least equally well.

Unfortunately, it doesn't work out that way. Classic learning-curve models assume production categories are either

- large lots or batches of relatively low cost units (like semiconductors) or
- tens or hundreds of very large identical or nearly identical units (like airplanes or ships).

Software projects have elements of both, but fall neatly into neither. A software project can be viewed as the production of many relatively atomic units (like source lines of code or function points). However, these tend to be aggregated into units with nonuniform granularity — program size varies widely, for example — which correspond to odd-size batches. This view obviously disrupts the classic learning-curve model.

On the other hand, if the unit of analysis is the software project itself, roughly corresponding to airplanes in the second category, the units are clearly not identical. This issue of the repetitive versus nonrepetitive nature of software development is receiving a lot of attention because it relates directly to software reusability. Software developers tend to treat each project as unique, when, in fact, research suggests that less than 15 percent of the code created is actually unique, novel, or specific to individual applications.[4]

An appropriate approach may be, then, to treat each project as a batch — in which the batch size is a measure of software size — and adjust the model to account for a possible wide variation in batch size. A modeling approach similar to that for a microeconomic production process may be quite suitable. In this approach, learning is merely one independent variable, which together with other variables (like the amount of input), is given equal opportunity to influence the result.

For example, slightly modifying the model of Argote and colleagues gives you

$$\ln q_t = \alpha + \beta \ln K_{t-1} + \chi \ln L_t + d \ln W_t + \varepsilon$$
$$K_t = \lambda K_{t-1} + q_t$$

where q_t is the output in time period t, K_t is knowledge gained during t, L_t is the labor input during t, W_t is the capital input during t, and λ is a depreciation of the knowledge parameter.

In this model the effects of learning (K_t) are separable from other possible effects, such as changes in the mix of capital and labor inputs or in scale.

Task diversity. Systems-development tasks can be anything from requirements analysis to testing and documenting source code. These diverse tasks are likely to reflect different rates of learning, and be supported to different degrees by the integrated CASE tool. The issue here is how much the task mix differs from project to project. If it differs markedly and if different tasks exhibit highly different rates of learning, the results may be anomalous. There are several ways to avoid this. One is to take great care in selecting as homogeneous a set of projects as possible to model. Another is to incorporate additional variables in the model to account for this mix discrepancy. Finally, discrete tasks within the project can be modeled separately. Each of these suggestions, whether done together or separately, carries with it some practical difficulties.

Organizations that have already adopted integrated CASE tools are the best data-collection sites, but finding them isn't easy.

Another problem is caused by how CASE tools provide different levels of support for different project tasks. For example, John Henderson and Jay Cooprider found that current tools differ significantly in their ability to support cooperative design activities.[5]

With this different level of support, plus a possible mix of activities across projects, an earlier project may be (anomalously) more productive because a particular life-cycle phase or task within a phase was both significant and relatively well supported. A later project, on the other hand, might be relatively unlucky on both counts. Moreover, an earlier project may use a different version of the integrated CASE tool.

Tool versus supporting methods. The distinction between learning the integrated CASE tool and learning the underlying or supporting methodology has received some attention in both the trade press and academic writing. Texas Instru-ments' Information Engineering Facility and Knowledgeware's Information Engineering Workbench are examples of learning the tools, while information engineering is an example of learning a supporting methodology. Distinguishing types of learning is fundamental to such notions as "readiness for integrated CASE," in which developers are recommended to delay adopting integrated CASE tools until they are fully comfortable with the underlying methodology.[6]

This distinction suggests using separate models to track the individual rates of learning and to track variables relating to the training received by the staff assigned to integrated CASE projects. Although research has documented the importance of training to learning,[7] organizations tend to underinvest in training. To estimate the benefit of training, managers can adopt a version of Paul Adler and Kim Clark's model:

$$\ln \frac{q_t}{l_t} = \ln \alpha + \beta \ln X_{t-1} + \chi \ln T_{t-1} + \varepsilon$$

where q_t over l_t is productivity during time period t and T_{t-1} is the cumulative hours spent in training during $t-1$.

Implementation issues. After the functional form of the learning-curve model is established, work can proceed on empiri-

LEARNING-CURVE MODEL: A FLEXIBLE MEASUREMENT TOOL

Part of adopting an industrial process is to go through a learning curve that measures the rate at which the average unit cost of production decreases as the cumulative amount produced increases. Learning curves do not relate solely to individual learning, although some authors have attempted to restrict it in this way, using terms like "experience curves" or "progress functions" to denote group or organizational learning. But more often "learning curve" is used in the broadest sense, as it is in this article.

Learning curves are also not restricted to the measurement of low-skill labor; their effects have been observed in skills like heart surgery, for example.

Several factors contribute to the learning curve, including

- labor efficiency, both in production and management;
- improved methods and technology;
- product redesign, with the reduction or elimination of costly features;
- production standardization, with a reduction in the number of setups or changes; and
- effects from economies of scale.

These factors are sometimes characterized as autonomous learning (automatic gains from learning by doing) and induced learning (conscious efforts by managers to observe and improve the process).

Software development exhibits all these factors. Production standardization, for example, is exhibited by organizations that batch small maintenance changes into a few releases. This might also be interpreted as an effect from economies of scale. Greater experience with tools and applications has also been widely suggested as improving software-development productivity.

Much has been written about learning curves. Louis Yelle gives a comprehensive review.[1] Three models are the most established: the traditional model (with variants) to estimate average unit cost and more recently a model devel-

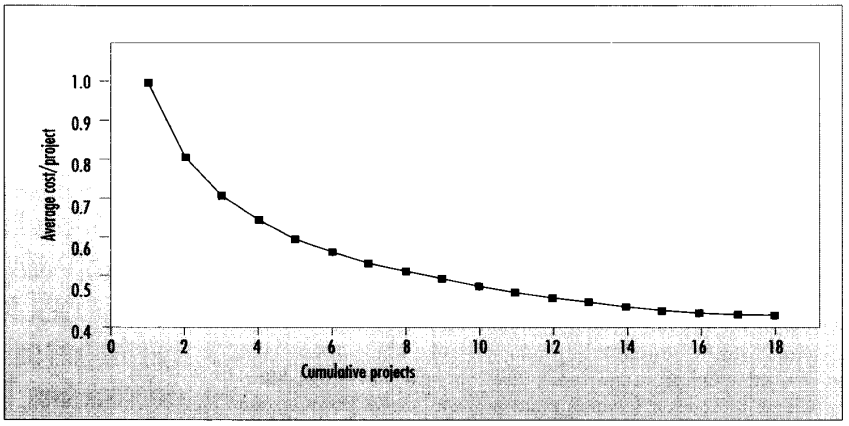


Figure A. *Traditional 80-percent learning curve* ($\beta = .322$).

cally validating models in integrated CASE settings. The three biggest tasks in implementation are locating a suitable data site, collecting the data, and validating the results.

Finding a suitable site. The best place to start looking for a site is in organizations that have already adopted integrated CASE tools. But this is no easy task for several reasons.

- Even if a suitable site can be found, most organizations simply do not collect the performance data necessary to quantitatively evaluate learning about integrated CASE tools. Work-hour data by project, if captured at all, may reflect sloppy or even incorrect bookkeeping. Staff members or managers may not report actual work hours if doing so will put a project over budget. Even worse, they may charge them to another project or to an overhead account, which will further compound the error if data is used for future planning.
- If an organization *does* keep detailed project data, it won't do much good unless management has also kept detailed data by person or by project phase. It is easy to imagine a situation in which a new project n is expected to demonstrate the effect of learning but, because not enough members of project n–1's team are transferred to the new project, they cannot. Without data on who charged the hours (and their level of experience with the integrated CASE tool), researchers cannot readily discern this scenario.[8] Some sites may have several integrated CASE projects going on in parallel, which create multiple organizational learning curves unless management makes a significant effort to transfer the knowledge gained.
- Even if organizations have collected a lot of data already, other data will probably be required to carefully construct a learning curve. Practitioners are understandably concerned about demanding anything extra from an already overburdened IS staff.

Given these problems, finding a site with enough completed integrated CASE projects and a relatively similar set of team members may seem like an impossible task. But the number of suitable sites is growing, albeit slowly, as integrated CASE catches on. Because modern technology and modern process and product measurement often go hand in hand, organizations that have adopted such tools are more likely to have modern measurement practices as well. Moreover, integrated CASE tools aren't cheap. Senior management has probably mandated the implementation of measurement to monitor the process as a prerequisite of adoption.

Thus, while practical data problems are significant impediments for most organizations to implement these models, they may prove much less formidable to early adopters of integrated CASE technology.

Collecting data. To minimize the effect on staff, researchers may have to limit the sample to a small group of hopefully representative projects. To reach a sufficient sample size, they can either use data from

oped by Linda Argote and colleagues and a model by Paul Adler and Kim Clark.

Traditional model. The earliest industrial learning curve is the Wright or cumulative-average curve, represented by

$$y = \alpha X^{-\beta} + \varepsilon \quad \beta > 0$$

where y is the average cost, α is the cost of the first unit, X is the total number of units, and β is the learning rate parameter, which can be estimated using least-squares regression after taking logarithms of both sides:

$$\ln y = \ln \alpha - \beta \ln X$$

β is sometimes expressed in percent, which reflects the percentage of decline in average cost with each doubling of cumulative volume:

$$\beta = \frac{\ln (\%)}{\ln 2}$$

Typical percentage rates observed in practice are from 70 to 95 percent. Figure A shows the curve for an 80-percent learning rate.

Smaller percentages indicate a relatively steeper learning curve, implying more rapid cost decreases. Therefore, when learning-curve researchers refer to steep learning curves, they are actually referring to a favorable event — as opposed to the popular use of the term, which implies something bad to overcome.

Recent research. The learning-curve model has been successfully used in a variety of settings and continues to be the source of significant research. Recent effort has focused on managing both the transfer and loss of learning.

The transfer of learning is the study of how knowledge gained at one site or installation is transferred to others. This transfer can be either internal, as from a pilot or leading-edge facility to the rest of the organization, or external, as in attempting to bootstrap on the efforts of other firms.

Adler and Clark describe three types of internal learning transfer: across the development/manufacturing interface, from start-up operations to other facilities, and ongoing cooperation between facilities. In all three cases, they find evidence of sharing but also suggest that more could be done to contribute to this.

Argote and colleagues examined transfer across shipyards building Liberty ships in World War II.[2] They found that, while there was some evidence of initial learning transfer (shipyards starting later generally showed higher initial levels of productivity than earlier shipyards), no other significant learning transfers seemed to take place.

Thus both Adler and Clark and Argote and colleagues show that the transfer of learning across organizations is limited.

A topic mentioned in the learning-curve literature but rarely studied is the loss of learning, or forgetting. The literature generally assumes a steady production process after start-up, with continuing benefits until the process is replaced. Recent experimental research conducted at Florida State University by Charles Bailey suggests that forgetting is a significant loss in procedural tasks that are interrupted for long periods. He found that the amount of forgetting was a function of the amount learned and the passage of time, but not the learning rate. The study by Argote and colleagues also noted forgetting, referring to it as the lack of persistence of learning. Their results suggest that cumulative output overstates the gains to be had from learning when the process has been significantly interrupted.

REFERENCES

1. L. Yelle, "The Learning Curve: Historical Review and Comprehensive Survey," *Decision Sciences*, Feb. 1991, pp. 302-328.
2. L. Argote, S. Beckman and D. Epple, "The Persistence and Transfer of Learning in Industrial Settings," *Management Science*, Vol. 36, No. 2, pp. 140-154.

completed projects or wait for data from future projects. The first option requires much care to ensure accurate data. Collecting historical data is often particularly problematic because many IS departments have high turnover. It is not at all unusual to begin collecting data on a completed project only to discover that the project manager or some other key individual no longer works there. Such projects may have to be excluded from analysis because records require interpretation or supplements from these key informants.

If researchers opt to use data from future projects, they should be aware that individuals collecting the data may perceive alternative uses (particularly managerial control) for it. For example, self-reported data on source lines of code or function points may be misrepresented to give the impression of high personal productivity.

Another problem in collecting future data is that the average-size firm may take a long time to generate enough new projects to make data collection meaningful. This is particularly true in evaluations of integrated CASE tools because many tools are designed to be of the greatest (perhaps any) value only on large systems. A firm is likely to do these large systems projects only infrequently, and of course, being large, they take a long time to complete.

A third problem is that the waiting period for data collection — given integrated CASE's slow adoption and relatively quick abandonment — puts research results at risk. Changes in the site's business or technology may obscure any meaningful results. As time passes, new versions of the tool will become available. These later versions may aid performance by providing more functions or greater ease of use, or they may actually hinder learning because they become more complex. Which effect dominates clearly depends on the site and tools.

An additional danger of long-term data collection is the loss of learning when tool adoption is interrupted (see box above).

Validating results. A final implementation problem is assessing the validity of the results once the research is complete. Integrated CASE tools, like other new technologies, are likely to be initiated in only one or a few specially selected pilot projects. These pilot projects may or may not be representative of systems projects as a whole.

If volunteers are solicited, there will clearly be some selection bias toward rapid technology adopters or simply staff members who are dissatisfied with their current work. Even if management selects pilot projects, the Hawthorne effect — the tendency for workers to show increased productivity under any new situation in which their performance is being monitored — may still dominate. It may, therefore be difficult to get a representative sample, and later projects may be sufficiently different to obscure learning effects.

Of course, the ultimate problem with any field study may be its external validity. Even if researchers can show the effect of an integrated CASE tool and support a causal relationship with statistical data,

they cannot assume that the results will extend to other firms, even similar ones. The implementation of integrated CASE tools at one organization may have been well received by staff eager for its use, well educated in theoretical background, and gently introduced to tool mechanics through excellent training and ongoing support. The same tool at another site may have been received with hostility, with the staff feeling it had been forced on them by management.

Staff acceptance is only one example of how organizations may differ, and therefore how a tool can fail to have the same impact across sites. Differences in applications mix, technical environment, personnel, management, users, backlog, organizational structure, and history can also affect results. For example, the makeup of project team members and their experience with new technologies plays an important role. The culture of the organization — in particular, the degree of local resistance to change — is also a factor. It is important to collect learning-curve data at many sites to test the effects of differences in these environmental conditions.

Learning-curve models clearly have much to offer organizations. Besides being able to quantitatively document the productivity effects of integrated CASE tools by factoring out the learning costs, managers can use model results to estimate future projects with greater accuracy. Without this depth of understanding, managers are likely to make less-than-optimal decisions about integrated CASE and may abandon the technology too soon.

Data from the models has other, more sophisticated, uses as well, which could lead to a greater understanding of how learning occurs, what factors affect it, and how learning time can be shortened — all of which are instrumental in reducing tool-adoption costs. For example, managers can control, and to some extent anticipate, most of the benefits associated with learning.

An important research benefit would be to look at how the emerging discipline of software engineering can benefit from the knowledge these types of models provide, with emphasis on the underlying concepts of formal models developed in other, more mature engineering disciplines. Learning-curve work could provide insight into how researchers could usefully adapt concepts developed in other domains to aid the understanding of software delivery.

Another useful general outcome would be to prove the value of process measurement to software engineering. While much lip service is given to the need for and importance of measurement, its adoption has been slow and easily abandoned. But if managers continue to apply the results of measurement programs, their value will be justified and investment in them sustained. With a sound measurement base many other software-engineering process improvements may be possible. Therefore, the knowledge gained from a greater understanding of the software-technology adoption process will aid the implementation of not only integrated CASE, but also potential innovations like object-oriented technologies.

Finally, much has been written about the nationwide trend toward a service-sector economy. One related issue is the low productivity of service-sector work and the general inability to effectively measure and increase it. Software development falls in the high end of the service-sector categories, an area of increasing concern and importance for worldwide competitiveness. Increasing the understanding of software development could benefit other high-end service-sector categories as well. ♦

ACKNOWLEDGMENTS

Research support from MIT's Center for Information Systems Research and helpful comments from three anonymous referees on an early draft of this article are gratefully acknowledged.

REFERENCES

1. C. Kemerer, "Learning Curve Models for Integrated CASE Tool Management," Working Paper 231, MIT Center for Information Systems Research, Cambridge, Mass., Nov. 1991.
2. M. Shaw, "Prospects for an Engineering Discipline of Software," *IEEE Software*, Nov. 1990, pp. 15-24.
3. W. Chew, D. Leonard-Barton, and R. Bohn, "Beating Murphy's Law," *Sloan Management Rev.*, Spring 1991, pp. 5-16.
4. T.C. Jones, "Reusability in Programming: A Survey of the State of the Art," *IEEE Trans. Software Engineering*, May 1984, pp. 488-494.
5. J. Henderson and J. Cooprider, "Dimensions of IS Planning and Design Aids: A Functional Model of CASE Technology," *Information Systems Research*, Sept. 1990, pp. 227-253.
6. J. Nosek, G. Baram, and G. Steinberg, "Ease of Learning and Using CASE Software: An Empirical Evaluation," working paper, Temple University, Computer and Information Sciences Dept., Philadelphia, Mar. 1990.
7. P. Adler and K. Clark, "Behind the Learning Curve: A Sketch of the Learning Process," *Management Science*, Mar. 1991, pp. 267-281.
8. R. Banker, S. Datar, and C. Kemerer, "A Model to Evaluate Variables Impacting Productivity on Software Maintenance Projects," *Management Science*, Jan. 1991, pp. 1-18.

Chris F. Kemerer is a Douglas Drane Career Development associate professor of information technology and management at the MIT Sloan School of Management. His research interests are the measurement and modeling of software development.

Kemerer received a BS in economics and decision sciences from the Wharton School of the University of Pennsylvania and a PhD in industrial administration from Carnegie Mellon University.

He has published numerous articles on software measurement and modeling. He serves on the editorial boards of *Communications of the ACM*, *Information Systems Research*, *Journal of Organizational Computing*, *Journal of Software Quality*, and *MIS Quarterly*.

Address questions about this article to Kemerer at MIT Sloan School of Management, E53-315, 50 Memorial Dr., Cambridge, MA 02139.