As projects get more complicated, managers stop learning from their experience. It is important to understand how that happens and how to change it.

The Experience Trap

by Kishore Sengupta, Tarek K. Abdel-Hamid, and Luk N. Van Wassenhove

If you were looking for an experienced manager to head up a software development team, Alex would be at the top of your short list. A senior manager, Alex has spent most of his career running software projects. His first responsibility was developing scientific software for NASA, and since then, he has overseen ever more complex projects for commercial enterprises and government agencies.

Alex was typical of the several hundred project managers who participated in our research initiative on experience-based learning in complex environments. We invited him to test his skills by playing a computer-based game that

Dave Wheeler

entails managing a simulated software project from start to finish – making the plans, monitoring and guiding progress, and observing the consequences. We set goals for him: finish on time and within budget, and obtain the highest possible quality (as measured by the number of defects remaining).

Alex's decisions and outcomes were representative of the group as a whole. He started with a small team of four engineers and focused mostly on development work. That tactic paid off in the short run. The team's productivity was high and development progressed quickly. However, when the size of the project grew beyond initial estimates, problems cropped up. Because Alex still chose to keep the team small, the engineers had to work harder to stay on track. Consequently, they made many mistakes and experienced burnout and attrition. Alex then tried to hire more people, but this took time, as did assimilating the new hires. The project soon fell behind schedule, and at that point Alex's lack of attention to quality assurance in the early phases started to show up in snowballing numbers of software errors. Fixing them required more time and attention. When the project was finally completed, it was late, over budget, and riddled with defects.

After the game, we asked Alex to reflect on the simulation. Did the project's growth take him by surprise? Was he shocked that the number of defects was so high or that hiring became difficult to manage? Alex – like most of his fellow participants – replied that such surprises and shocks have, unfortunately, become regular occurrences in most of the projects in which he's been involved.

Quality and personnel headaches are not what most companies expect when they put seasoned veterans like Alex in charge of important projects. At this stage of their careers, they should know how to efficiently address problems – if not prevent them altogether. What we discovered in our experiments, however, was that managers with experience did not produce high-caliber outcomes. In our research, we used the simulation game to examine the decision processes of managers in a variety of contexts. Our results strongly suggest that there was something wrong with the way Alex and the other project managers *learned* from their experiences during the game. They did not appear to take into account the consequences of their previous decisions as they made new decisions, and they didn't change their approach when their actions produced poor results.

Our debriefings indicated that the challenges presented in the game were familiar to the participants. We asked them to rate the extent to which the game replicated their experiences on real-life projects on a scale of 1 to 5, where 5 meant "completely." The average score was 4.32, suggesting that our experiments did accurately reflect the realities of software projects. So, though the managers had encountered similar situations on their jobs in the past, they still struggled with them in the simulations. We came to the conclusion that they had not really learned from their real-life project work, either.

In the following pages we'll identify three likely causes for this apparent breakdown in learning, and we'll propose a number of steps that organizations can take to enable learning to kick in again.

Quality and personnel headaches are not what most companies expect when they put seasoned veterans in charge of important projects.

Why Learning Breaks Down

When anyone makes a decision, he or she draws on a preexisting stock of knowledge called a mental model. It consists largely of assumptions about cause-and-effect relationships in the environment. As people observe what happens as a result of their decisions, they learn new facts and make new discoveries about environmental relationships. Discoveries that people feel can be generalized to other situations are fed back, or "appropriated," into their mental models. On the face of it, the process seems quite scientific – people form a hypothesis about a relationship between a cause and an effect, act accordingly, and then interpret the results from their actions to confirm or revise the hypothesis. The problem is that the approach seems to be effective only in relatively simple environments, where cause-and-effect relationships are straightforward and easily discovered. In more complex environments, such as software projects, the learning cycle frequently breaks down. In the experiments we carried out with our study participants, we identified three types of real-world complications that were associated with the cycle's breakdown.

Time lags between causes and effects. In the real world, there are delays between causes and effects, and it may become difficult to link them, let alone specify the relationship between

Kishore Sengupta (kishore.sengupta@insead.edu) is a professor at Insead in Fontainebleau, France. Tarek K. Abdel-Hamid (tkabdelh@nps.edu) is a professor at the Naval Postgraduate School in Monterey, California. Luk N. Van Wassenhove (luk.van-wassenhove@insead.edu) is a professor at Insead.

them. To see how project managers cope with this issue, we asked participants in our research to play a simulated game in which they managed a medium-size satellite-software development project that grew significantly in size as more product requirements were added. Each participant had to oversee the project in one of four operating environments we'd created, which varied in terms of the time that lagged between a decision to hire and the arrival of new team members, and between the team members' arrival and their assimilation. Participants had to make a decision on the staffing level of the team every two months in a project that took around 18 months to complete. We then assessed managers' ability to handle time lags, by comparing their hiring decisions both with the decisions made by a theoretical naive manager who never accounted for time lags and with the decisions made by a theoretical perfect manager who always did.

Regardless of the hiring and assimilation delays in their respective project environments, all participants made more or less the same decisions as our naive benchmark. That shows they were unable to incorporate the effects of time lags into their planning decisions and suggests their mental models were based on a simple environment in which there was little or no delay between a decision and its result. The length of the lag mattered: Participants in environments with longer hiring and assimilation delays had more difficulty coping than participants who experienced shorter delays. The type of lag was also material: Subjects had greater difficulty handling assimilation delays, which are much less visible than hiring delays. The ability to manage lags deteriorated sharply – and disproportionately – when subjects were required to manage long hiring lags followed by long assimilation delays. Subjects working under those conditions incurred 83% more effort (in personnel time) and took 40% longer to complete the project than those making decisions in the low hiring- and assimilation-delay environments.

Interestingly, in many cases the participants decided to hire more staff late in the project, which ran counter to what they later said managers ought to do. In postgame debriefings we asked subjects to describe appropriate hiring policies to adopt when projects ran late. Most of the experienced managers stated that they would refrain from hiring and look to other options such as reframing the project, zeroing in on a few key priorities, or extending the deadline for completion. However, that was clearly not what they actually did. In a follow-up experiment where participants managed a second project after the debriefing, the same behavior persisted: Those managing projects with long time lags still hired more staff late in the project. This suggests that even when people had or acquired knowledge, they did not necessarily learn how to act on it.

Fallible estimates. In software development, initial estimates for a project shape the trajectory of decisions that a manager makes over its life. For example, estimates of the productivity of the team members influence decisions about the size of the team, which in turn affect the team's actual output. The trouble is that initial estimates usually turn out to be wrong.

To see how managers handle fallible estimates, we conducted another experiment. In it, we examined a cycle of decisions wherein managers received initial estimates of the project team's productivity and were then asked periodically to provide their assessment of the team's actual productivity, based on progress made. Each manager got one of three initial estimates of how many tasks the team would accomplish per person per day. One estimate was low, one medium, and one high – reflecting the wide range of values that different estimation tools can produce for the same project. The managers had to provide updated estimates of the team's productivity at three points during the game: the end of the design phase (fifth month), the middle of the coding phase (10th month), and the end of the coding phase (15th month). At each point, the managers received progress reports on the project's status with new estimates of productivity and were advised to review them before providing their own estimates of the team's productivity.

The participants were told that their productivity estimates would be used for making the adjustments to the project's staffing levels and schedules. In reality, however, the game disregarded the estimates. The idea was to give all subjects identical status reports, so we could compare how people's productivity estimates evolved over time. Our hypothesis was that people's productivity estimates would converge (people starting with low estimates would raise them over time and those with high estimates would lower them).

So what happened? The managers' productivity estimates did not converge over time. What's more, there was a clear bias toward conservativeness: All their estimates drifted downward. That was true not only for managers given high initial productivity estimates but also for those whose initial estimates were low. And when faced with two estimates of productivity (their previous estimate and the new number provided by the status reports), they accorded greater weight to the *lower* of the two figures in revising their estimates. We suspect that this conservatism can be explained by managers' attempts to game the system to get more resources.

Initial goal bias. Project managers usually begin with a set of goals related to cost, time, and other factors. But most projects change in scope or encounter the unexpected, which frequently renders early targets obsolete. When that happens, managers need to revise their targets accordingly.

To see if managers did amend their targets in response to changes in scope, we asked two different groups of subjects to manage a project that increased substantially in requirements. Each group received an initial set of two targets. The "cost group" subjects were asked to stay within budget (944 person-days of effort) and deliver the product on schedule (within 272 working days). The "quality group" subjects were asked to deliver the product on schedule and with the fewest number of defects. It was clearly stated that these were initial targets only, based on information available at the time, and that participants' success would be evaluated on the overall outcome. The increase in scope happened a quarter of the way into the game. At that point, managers could have opted to revise their initial targets by projecting budget or time overruns while sticking to initial quality goals. Although we did not ask players to explicitly reevaluate their targets, we were careful to leave the possibility open to them.

Neither group readjusted targets in light of the new information. Instead, players in both groups stuck to their original targets, and as a result they all failed to achieve an optimal outcome. In an effort to keep costs down to the initial target, the cost team made far fewer hires than was ideal and sacrificed completion time. Although these players kept the cost overrun down to 59%, they took 17% more time to complete the project and their number of defects rocketed to 1,950. The quality team, on the other hand, employed too many people. The players in this group hit their defect target, but they still finished 9% over schedule time and came in a whopping 107% over budget. In some cases sticking to initial targets actually created counterproductive outcomes. In trying to meet budgets, the "cost group" subjects often paid little or no attention to quality assurance. In the process, they created so many errors that the effort it took to fix them substantially drove up the cost of the project.

These results suggest that if not explicitly required to reevaluate objectives, managers will continue to pursue the targets set at the outset of a project, even when events render the targets inappropriate. It's not hard to see where that bias comes from. Very early in their careers, people incorporate into their mental models the notion that it's important to meet externally set targets. This bias is often reinforced in managerial life. Revising targets is seen as an admission of failure in many companies, and managers quickly realize that their careers will fare better if they stick to and achieve initial goals – even if that leads to a worse overall outcome. With the bias so firmly embedded in the mental model, it's hardly surprising that it affected decision making in our simulation, even though the participants understood that success would be measured by the project's results.

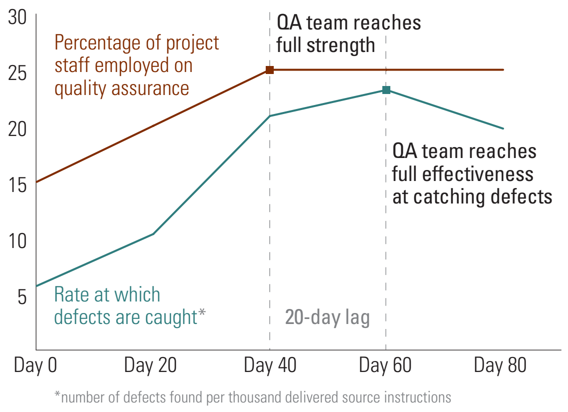
Despite their experiences with complex projects, the veteran managers do not meaningfully improve the mental models they have formed in simpler contexts.

• • •

We conclude that managers find it difficult to move beyond the mental models that they have developed from their experiences in relatively simple environments or that have been passed on to them by others. When complications are introduced, they either ignore them or try to apply simple rules of thumb that work only in noncomplex situations. What they don't do is materially improve the quality of their mental models to take into account the realities of complex

Cognitive Feedback in Complex Projects

This chart shows managers that there's a long lag between an effect, the number defects caught in a software development project, and its cause, hiring additional quality assurance staff. The two lines indicate a 20-day lag between the time the QA team reaches full strength and its full effectiveness at catching defects. The chart also suggests that the team may be able to reduce the QA staff after day 60, when the rate at which defects are found drops, which most likely indicates that the team has become experienced and is making fewer mistakes. Armed with such data, managers can make better staffing decisions. The connection between cause and effect is clearer when you have the feedback in this form: visual, immediate, and with the benefits of hindsight to guide you.



projects. This conclusion has two important implications for companies that continue to emphasize learning on the job:

First, the impressive backgrounds of people like Alex will have little bearing on their ability to manage complex projects. Many companies routinely find that replacing one veteran project manager with another has no impact. Despite their experience with complex projects, both managers do not meaningfully change the mental models they've already formed in simpler and usually similar contexts. In some cases, in fact, companies might even be better off hiring someone who didn't have experience. That's not to say that different managers don't make different decisions or that circumstances may not conspire to make a particular project turn out well, or even that a few managers aren't consistently successful over time. The point is that most managers, even those with impeccable résumés, fail to turn in consistently good, let alone improving, performance on the projects they run. Even when managers do consistently better their performance, the improvement is probably the result of some subtly different past experience rather than systematic and incremental learning from complex projects.

The second implication is a corollary of the first. If it makes little difference whom you put in charge, then managers will end up ascribing responsibility for failures not to their own decisions but to some other factor: overambitious planning or the demands of the finance department (or – as is often the case – a salesperson promising too much to the client and then setting unrealistic goals for the project). When that kind of belief takes hold, managers start to look in the wrong places for solutions to their performance problems. That can be a recipe for disaster.

Fixing the Experience Learning Cycle

Although our research indicates the experience learning cycle has broken down for most managers of complex projects, it can be mended. There are a number of practical steps organizations can take to get managers to start learning in complex situations. Some of our recommendations *accept* the deficiencies of the experience learning cycle and involve helping managers work around them by supplying other types of learning. Other approaches aspire to *reduce* the deficiencies of the cycle through improved discovery and appropriation. Companies that adopt these recommendations will quickly find that their ability to improve project-management performance continually increases.

Provide more cognitive feedback. Project environments are rich in information, particularly feedback on outcome, which is delivered through status reports. But in environments where cause-and-effect relationships are ambiguous, outcome feedback is not an effective mechanism for discovery or for identifying reasons underlying a specific problem. What managers need is feedback that provides insights into the relationships among important variables in the project environment, particularly as the project evolves. This is called *cognitive feedback.* For an example, see the exhibit "Cognitive Feedback in Complex Projects," which depicts the relationship between the level of quality assurance and the rate at which defects are caught in the first 80 days of a project. In this case, the manager has chosen to start the project with a relatively low level of quality assurance and has increased it over time. The rate at which defects are caught increases correspondingly, but with a lag, and disproportionately because more effort is now devoted to detection. The rate then decreases, signaling that most of the defects are being detected, and the manager can now maintain quality assurance at this level or even reduce it. While such feedback is not error free, it enables managers to learn about complex dynamic relationships. Our research has demonstrated clear benefits from it: Managers who were provided with cognitive feedback in our simulations showed a deeper understanding of their environments and made decisions that resulted in better outcomes. We recommend that companies invest in making cognitive feedback a part of regular project status reports. What's more, we've found such feedback to be even more effective when data from different projects are combined, so that the impact of actions across multiple projects can be examined.

One leading provider of corporate software that we know employs cognitive feedback in its development projects. The consensus among executives there is that this has helped project managers develop better insights. Their decisions also appear to have improved: The proportion of problem projects has fallen by 56% in three years, the company calculates.

Apply model-based decision tools and guidelines. Our research consistently demonstrates that managers can't do adequate mental bookkeeping in the dynamic aspects of software project management. Bare intuition is not enough: Managers facing decisions need the assistance of tools that combine formal models and heuristics. Consider staffing decisions. When a manager makes several hires, there is a hiring delay and an assimilation delay with each. Over time it becomes difficult for the manager to assess current and predict future team productivity, especially if the staff suffers attrition. But if the manager is provided with tools that can calculate the effects of additions and turnover for several periods, he will obtain a clearer picture of the expected cumulative impact on team productivity over the medium term. In addition to formal models, such tools can contain mechanisms such as trip wires for projects in trouble (flagging when a manager should consider reducing scope, for example) or rules about the appropriate balance of development work and quality assurance at various stages. Our research shows that such tools improve decision making and help new managers get up to speed faster.

A leading provider of software we worked with has an extensive portfolio of decision support systems for this very purpose. The firm's managers can use the systems to gauge likely attrition, analyze the effects of new hires on team productivity, and get guidance on such questions as whether it is useful to hire at all at late stages in the project. The managers that use the decision support tools report feeling significantly more in control of their projects and demonstrate much better project performance. The company also has one of the best reputations for quality in the industry.

Calibrate your forecasting tools to the project. The tools that organizations rely on to generate project estimates should be calibrated to the project's specific context – to the industry, the local environment, and the skills of the available staff. Many organizations, however, simply import project-management forecasting tools from other contexts and other companies. One software company we studied had just adopted a tool from an aerospace company. Organizations compound estimation problems by basing their model assumptions on data from past projects without scrubbing the data first (that is, without accounting for any unusual circumstances encountered by those projects). Not surprisingly, the resulting estimates tend to be unreliable and have little credibility with project managers. When they lack faith in the estimates, project managers will rely on their own perceptions and revert to applying rules they've developed for simple situations. To avoid this, companies must do everything they can to instill managers' faith in the projections, and that means customizing forecasting models to project needs and cleaning up the data used to drive assumptions and infer relationships. Also, the more managers invest in gathering and processing their own data, the better their forecasting will become. This is one area in which simplistic "best practice" benchmarking from successful project managers can be very dangerous.

The research and development center of one leading producer of semiconductors has developed a way to reduce estimation fallibility. For every completed project, the center "normalizes" outcomes in a three-step process that identifies unusual events, roughly calculates their impact, and then deducts the impact from the results. The scrubbed values then go into the estimation models.

The more managers invest in gathering and processing data, the better their forecasting will become.

Set goals for behavior, not targets for performance. Another weakness of estimation tools is that their projections are usually based on product size (for example, how many lines of code or function points), which is extremely difficult to predict in the planning stages. Moreover, product deliverables can change over time in ways that are difficult to anticipate. Thus, initial estimates don't make good goals. Indeed, when so used, they promote inappropriate responses such as ad hoc trade-offs between cost and quality, and lead to poor outcomes.

Yet software projects universally employ cost and schedule targets based on early predictions. And when managers know they will be measured against targets based on unreliable estimates, they seek additional slack by opting for "safe" estimates and then proceed to squander the slack through make-work and by embellishing the project with unnecessary features. There is thus a strong case to be made for rethinking the way goals are set.

In particular, companies need to understand that estimates function best as devices for planning and control, and goals as mechanisms for promoting desired behavior. We recommend that when they're establishing goals, organizations follow a two-step process: First they should decide on the behavior they wish to foster, and then they should set goals that encourage such behavior. In a single project, an organization might decide it wants its managers to minimize turnover on the project team (doing so can increase productivity and learning, and reduce errors). This can then be an explicit part of the goal set. To meet that goal, managers would have to formulate ways to cushion their teams from schedule pressures and from the impact of normal attrition.

Companies would be better advised to leave their junior hires to fend for themselves and to focus their training budgets on people higher up the corporate hierarchy.

We've found that when managers have responsibility for multiple projects, their goals should promote behavior that maximizes the success of the portfolio (rather than individual projects). In setting such goals, the organization must give managers a certain degree of freedom, allowing them, for instance, to negotiate trade-offs between scope and schedules to preserve team stability or prevent problems from infecting other projects. Additionally, to ensure greater commitment, organizations must give managers a say in composing the goals.

Develop project "flight simulators." It's clear that live projects don't provide a good learning environment. It is, however, possible to construct artificial environments that can be managed so that complexity does not overwhelm learning. For an analogy, consider the use of flight simulators in aviation. Skills for flying planes are highly model-specific: Pilots need to undergo extensive training every time they switch models (or even move from, say, a freight version of a Boeing 747 to a passenger version). Flight simulators are an essential part of that process. Appropriately constructed "flight simulators" can play a similar role in project management, as virtual worlds for training and immersion. The need for them is especially pronounced because project managers now move across organizations more often than they did in the past. Since knowledge has a situation-specific (or company-specific) aspect, each time managers change companies or work contexts they need to learn about the relationships in the new environment, such as which factors drive productivity or quality. We suggest a graduated training program, where managers can start with lenient environments, in which the relationships to be discovered are simple. The trainees would then move through progressively more demanding environments, where the relationships become more complex and the feedback is less reliable. (That can be engineered by continually increasing time lags between causes and effects.) As the trainees progress, we suggest, programs should increase their focus on dynamic relationships – such as the connection between hiring decisions and quality assurance outcomes – because these are the ones that are hardest for managers to understand.

This flight simulator approach worked well for one maker of satellite software we worked with. The company has developed a project-management game that incorporates the realities of its own environment – such as the factors that have the most impact on quality and productivity in its business – and successfully mimics the processes and outcomes of actual projects done by the company. New managers use the game to learn the essentials of project management before taking on project responsibilities. Initial results are promising: The managers have shown considerably better insights about the dynamic relationships at work in their projects, and the projects' performance has also improved.

• • •

The problems with the learning cycle we've described are certainly not the only breakdowns that occur in learning. Nor do we pretend that our recommendations will fix all the problems. But the studies we've conducted provide compelling evidence that learning on the job simply won't work in any but the most basic environments and that managers can continue learning only if they're given some formal training and decision support specifically tailored to the challenges they will face. As it happens, companies typically spend training dollars most heavily on entry-level hires and usually import project-planning tools wholesale from other companies. Senior recruits are expected to hit the ground running and best practices are supposed to be just that. These expectations are precisely why so many experienced managers fail when they take on new responsibilities. Companies would be better advised to leave their junior hires to fend for themselves, to focus their training budgets on people higher up the corporate hierarchy, and to stop looking for quick fixes from other places.

Reprint R0802F
To order, see page 139.