Four Failures in Project Management

Kenneth G. Cooper

© Kenneth G. Cooper

The project was slated to design and build an exciting first-of-a-kind product. But during the development effort, unexpected problems emerged... changes in the product specifications, shortages of qualified people, material supply delays. Costs escalated, and work fell far behind schedule. The project's future was threatened, and the work was interrupted. Eventually, project objectives were scaled back and the work was completed -- years late and at a cost more than double the original budget.

Another project conducted just two hundred miles away also aimed to develop a revolutionary product -- this one a new software system. Ill-defined design specifications and changes plagued this project as well; here, too, qualified staff were difficult to find. Schedule pressures mounted as months and years passed beyond targeted delivery. Extra staff, more overtime, tighter schedule targets all failed to bring it under control, as it too finished years late, spending far more than double its planned cost.

If only the managers of both projects could have compared notes -- talked with one another about solutions attempted and failed -- they might have discerned common problems and learned effective solutions. But this seems to be too tall an order even for projects within the same company, let alone for these projects separated by... two centuries.

The product sponsor in the first tale of project woes was George Washington; Paul Revere supplied the required copper and brass fittings. The special timber that delayed the work's progress eventually provided the project's product with its nickname, "Old Ironsides." The U.S.S. Constitution inaugurated the U.S. Navy and was, in 1794, a new nation's introduction to the challenge of large project management. Today the challenge remains largely unanswered, as evidenced by the problems, failed management efforts, and outcome, all so perfectly duplicated two hundred years later by a prestigious New York banking firm in its effort to develop a financial software system.
If those of us who manage projects have proven one thing, it is that we know how to fail; after all, we do it so consistently. A recent worldwide survey proved what we all suspected: the majority of all development projects fail to meet their time and cost targets\(^1\). Projects and problems just as those described are lamented regularly in company boardrooms and business publications. For those whose business is projects -- construction firms, defense contractors, design agents -- the business impacts are obvious, large, and bad. For all other technology-dependent companies, the effects on business -- revenue, growth, profits, market share, reputation -- are just as large and just as bad.

Computer hardware and software companies miss projected new product introduction dates with such regularity that on-time performance is news. Automobile companies struggle to reduce new car development costs and time, so as to compete more effectively. Banks, insurance firms, and other service institutions conduct in-house or contracted development efforts to improve their operational efficiency and quality. Telecommunications companies compete, in an increasingly complex arena, through system development efforts that aim to reduce costs and broaden product and service offerings.

This chapter is targeted at all the project customers who have sat by, chagrined over the ever-growing cost estimates and ever-slipping completion date targets, or who have jumped in themselves to "help" manage. It is for the company executives who have listened to project managers report first glowingly, then sadly, the state of key development efforts, who set the business policies and environment within which those projects must be executed, and who lead businesses whose survival depends on succeeding with new development efforts. And this chapter is for all project managers who must deal with the aforementioned, while working with independent-minded engineers, otherwise-incentivized matrix department heads, remote vendors, even consultants and lawyers, in a dozen meetings a day, to manage projects that are under-

funded, tightly scheduled, under-staffed, and closely watched, to develop products that are technically ambitious, ill-defined, and critical to the success of the company.

Teams at my firm have conducted analyses of over seventy major projects; their operations touch five different decades, including the next to come. The products of their development and build efforts include power plants, software systems, aircraft, tunnels, electronics, ships, control systems, missiles and automobiles. Their technologies range from the stable to the unheard-of; their locations span half the globe. Some we have helped to succeed; we have retrospectively diagnosed the failure of others. With that mix of products, technologies and industries among seventy-five projects, we might expect seventy-five different sources of project problems and failures (this vendor, that radar...); project-specific excuses abound. Instead, we found four fundamental drivers of project failure. Given the luxury of dispassionate analyses, we saw that how organizations performed on these characteristics determined the magnitude and consistency of project failure. We saw that all four are manageable. Improvement on any one of the four produced improvement in project success, and improving all four yielded dramatic success. None is any surprise to the experienced project manager, but the near-universal tendency to think “but this project is unique” dramatically reduces the appreciation of just how much these manageable, systemic conditions drive project performance [See Box], and leads managers to repeat these avoidable sources of failure.

_The four failures are:_

1. Failure To Know What to Expect...
   _Great Expectations_

2. Failure To Know What to Watch...
   _Half-Blank Tape Measures_

3. Failure To Know What to Do (and To Do It)...
   _Counterintuitively Counterproductive Countermeasures_

4. Failure To Know What’s What...
   _Lessons Not Learned_
Just How Much Difference Does Managing Make?

In a recent opportunity to compare performance on two similar-product system development programs at the same company, the degree of performance difference caused just by the way managers manage became apparent. The programs were planned as (A) a 300,000-hour effort over 3 years, and (B) a 400,000-hour effort over 4 years. Both saw their own peculiar conditions and work scope growth. The more troubled program (A) experienced a 200% schedule overrun and 300% cost overrun. The other (B) finished on budget and on schedule, and many reasons for their good fortune were cited by observers -- a "better" customer, better tools and hardware, better labor market conditions. Some even noted "better management" as a cause (although the majority with this opinion had themselves managed parts of the program; others discounted the objectivity of that causal diagnosis.)

Program A was understandably regarded as a cost and schedule disaster, Program B a success. Despite this perception, the organization's top management was dissatisfied with the extent to which other programs were adopting the processes and practices employed by the managers of the successful program. Our analysis task was to separate the external causes from the contribution of internal management practices, and to quantify the sources of cost and schedule improvement. Managers of other programs then would know which practices to emulate in order to achieve better performance.

Simulation models of the two programs enabled us to see how they would have performed with equal work scope. Another simulation test removed all "external" differences -- setting the vendor, hardware, and labor market conditions of the poor-performing Program A to those experienced by the on-target Program B. Finally, several model changes were made to reflect the different internal task scheduling, staffing policies, and work practices implemented by managers of the on-target Program B.
Together, the full set of model changes accounted for all of the actual cost and schedule performance difference between the two programs. After netting out the modest work scope differences, internal management practices caused over 60% of the cost improvement (a half-million hours saved) and over 70% (about 2 years) of the schedule performance improvement. Were it not for the improved management practices, the on-target program would have required twice its achieved cost and nearly half-again the time.

Many of the two programs' differences in management practices are subjects of this chapter's discussion of the Third Failure, "Counterintuitively Counterproductive Countermeasures."
1. FAILURE TO KNOW WHAT TO EXPECT: Great Expectations

It is axiomatic that if we set project targets poorly (usually meaning over-optimistically), we will perform “poorly” (relative to those targets). If it were just that simple, projects would miss their targets by exactly the amount of inaccuracy in the targets. But the consequences are much worse than any obvious one-for-one linear inaccuracy in the planning of the product and its budget and schedule.

Product Definition

Ask any home-building contractors what they fear most, and you will consistently hear “changes.” Or ask any defense system developers what got them in so much trouble on that problem-ridden contract, and the answer will likely be “changes.” The new automobile development’s slow pace? Changes. The slippage in the market date for that new long-awaited software? Changes. Spec changes, feature changes, requirements changes, design changes, change orders, change notices, change paper, late adders, revisions, re-releases, ...

Rework in all its forms and names is an inevitable part of projects (see the Second Failure). The instability of the initial product definition is a major contributor to the amount of change that occurs in a project. Devoting more time and effort “up front” to the product’s definition is a high payback investment. The use of common “platforms” and “objects” across multiple development projects are increasingly used as successful practices to reduce change in product specifications.

The clarity and stability of product definition deserves all the attention it can get. Our analyses of dozens of project show that for every obvious hour of technical product change avoided, two to four (or more) hours of subsequent effort are saved. This is due to the much reduced efficiency at which unplanned, out-of-sequence changed work is conducted, as well as the “knock-on” rework effects of changes (we’ll have to move that
wall to re-route that piping so that we can re-locate that junction box we need for that new electrical device.) We have seen product definition problems in and of themselves cause 50-100% growth in projects’ time and cost.²

If the project is an internal development, the consequences obviously extend even further, as the new service/system/product is delayed along with its benefits and its revenue-producing life. If an externally-funded project, would we not expect the “changes” clause in all such contracts to compensate contractors properly? No. We are universally inept under-achievers in estimating and explaining the full impacts of changes. An entire body of contract dispute case law regarding “delay and disruption” testifies to our collective inability to define correctly and adequately at the time of a change its full costs.

Cost Budget

This one seems pretty obvious: set your budget too low, and miss it by that amount; set it too high, and you become a hero. The latter case is rare, as optimism, competing projects and budget-tightening pressures usually translate into low cost budgets for projects. Why should we care? Why, aside from the obvious, is a too-low budget a source of project failure?

Inadequate budgets create two categories of problems -- one for the projects themselves and one for the organizations in which the projects are executed. At the project level, a too-low budget causes several conditions that actually lead to higher costs than if the project were to have a higher budget. These conditions begin with the inevitably heightened budget pressure of a too-low budget. Good tools, equipment, and

² For a delightful example, see the Cary Grant-Myrna Loy movie classic “Mr. Blandings Builds His Dream House,” wherein the I’ve-seen-it-all contractor explains the ballooning house-building cost stemming from a “simple” mid-project change the Blandings wanted. The example is a fictional house, but you can get a similar, albeit less humorous, explanation from any experienced manager of big construction, defense system development, shipbuilding, automobile design, electronic systems, or software projects.
materials needed for a productive effort are more likely to be short-changed. Worker morale is worsened by excessive budget pressure, and the short-sighted actions it often prompts, hurting worker productivity further, and driving costs up more.

In no way are budget pressures more hurtful than in their adverse effect on staffing decisions, especially late in the game. The inadequate project budget is allocated among tasks, becoming inadequate task budgets. As those work tasks near their hoped-for completion, budget pressures are at their worst, depressing the application of staff to the work at hand at the very time they are most needed -- needed for finding and fixing rework, and concluding a high-quality "product" that will be used by downstream tasks. This typical budget-induced late staff-cutting merely pushes the remaining rework to a later stage, where it will be executed less productively and cause even more rework on dependent items. Total project costs climb further upward.

It need not happen this way. The manager of the tremendously successful "Peace Shield" air defense program at Hughes Aircraft Company, C.W. ("Chuck") Sutherland, defied tight budget pressures in managing his resources. Where conventional wisdom said otherwise, he chose the seemingly risky path of "overstaffing" the back end of design work stages, expressly for the purpose of finding and fixing rework. His team sought to generate a higher-quality interim product on which later work stages could build. This, with many other such moves, yielded an on-budget project, and a high quality system completed in seven months early. Air Force Service Acquisition Executive Darleen Druyun observed, "In my 26 years in acquisition, this [Peace Shield] is the most successful program I've ever been involved with, and the leadership of the U.S. Air Force agrees."3

With these cost-affecting project conditions being driven by the budget itself, the organizations in which such projects operate are handicapped. For them it is even more difficult to budget new work accurately based on their performance on past work. There

3 Program Manager, March-April 1996
are always many other possible sources, reasons, and excuses available for the casual causal diagnosis of over-budget projects. The organization has legitimate difficulty sorting out the extent to which (or even if) chronic underbudgeting is contributing to the gap between the project targets and the achievements.

But if there is one thing that drives the business of a project even more than the cost budget, it is the schedule.

**Schedule**

Setting and achieving an aggressive schedule target is perhaps the most sacred of all sacred cows in the field of project management. It is also the source of some of the most destructive behavior and phenomena in projects.

The quest for early project completion is understandable. New products early to market command a favored position. New systems enhance business' operating efficiency. Early/on-time projects save the costs of "marching armies" of support personnel, and enhance a contractor's standing among customers.

With these laudable outcomes in mind, customers, executives and managers together routinely agree on overly-ambitious project schedule targets, only for the project to spiral downward in an ever-degrading set of conditions induced by the very targets it sought. Indeed, the result is that both schedule and cost targets are typically exceeded by far more than the original amount of inaccuracy. While avoidable, these schedule-induced phenomena are typical and systemic:

- *excessively overlapped work stages* create low-productivity, high-rework, time-consuming conditions in the downstream stages, which are typically started far too early, in an effort to make (and show) "progress";
• *schedule pressure* induces a “get the product out” mentality that sacrifices completeness and correctness, even proper task sequence, for the short-term appearance of progress, thus sowing seeds of later problems;

• *inefficient, costly resource use*, such as costly “body-shopping” or excessive sustained overtime, generate control and productivity penalties that cost the project both extra dollars and extra time;

• *worker morale*, especially when schedule pressures are overlaid on budget pressures, suffers when problems mount on an over-target project, further hurting productivity and total project performance.

In the remainder of this chapter, we will examine the operation, and quantify the consequences, of these phenomena.
2. FAILURE TO KNOW WHAT TO WATCH:

Half-Blank Tape Measures

Imagine that you have contracted for your house to be built, and that the standard tape measure used by all the contractors has long segments -- half the tape, in fact -- that are uncalibrated. What sort of product would you expect from their efforts? We are, in effect, equipping managers with half-blank tape measures as the standard tools for planning and monitoring projects.

When we were first asked by a contractor client to build a computer-based model capable of accurately simulating the performance of a large project from the start of design through the completion of construction, large portions of the project effort simply could not be described or explained by applying the standard available tools. How could this be?

The Critical Path Method (CPM) has long dominated among techniques for project planning. This method provides a framework in which the duration of, and linkages between, individual tasks can be planned. From this, the sequence of tasks may be identified which, if one element on the path were to be delayed, would translate to a delay in the entire project. It is an accepted, often required, technique for planning projects and for testing schedule impacts. It is the basis for virtually every piece of popular project management software offered. Properly constructed and updated, it is an extremely useful planning tool. And yet CPM is an inadequate model for managing complex development projects.

The typical means for monitoring project progress and ongoing cost and schedule performance are variations of earned value systems. These provide for setting work and budget standards for individual tasks. Progress on the tasks, and cost and schedule variations, are assessed by comparing actual effort and cost with the task budgets. The earned value system for project monitoring is, like CPM, an accepted, and often contractually required, project management technique. Truthfully and faithfully employed,
it is a highly disciplined monitoring method. And, like CPM, earned value is an inadequate model for managing complex development projects. So what’s missing?

What is missing, as any experienced project or program manager knows, is rework. For all their utility, conventional methods treat a project as being composed of a set of individual, discrete tasks. Each task is portrayed as having a definable beginning and end, with the work content either "to be done" or "in process" or "done." No account is taken of the quality of the work done, the release of incomplete or imperfect task products, or the amount of rework which will be required. This is particularly inappropriate for development projects, in which there is a naturally iterative process of design/engineering.

Indeed, our analyses have shown that rework can account for the majority of work content (and cost) on complex development projects! While this varies significantly among projects and project types, it is hardly ever a matter of a single re-visit of a particular task. Instead, several iterations are typical, often far removed in time from the scheduled and actual conduct of the first round of work on the task. This is readily seen in, for example, the release of initial engineering drawings, "A" revisions, "B" revisions, "C" revisions... (for those companies or projects which actually monitor such information). Companies experienced in complex projects know to expect this, and have developed rules of thumb to count on 2 (or 3 or 4...) revisions per engineering product. Even so, this expectation rarely is incorporated explicitly in work planning and management systems--because the techniques don't allow it. Worse are the cases where this rework cycle is not explicitly anticipated or monitored. Here they are not only working with half a tape measure--they're reading it with their eyes closed! These are not unintelligent people nor project-naive companies; on the contrary, they include the most technically sophisticated individuals conducting and managing complex developments in firms whose very existence depends on successful project performance. What we need, then, is a different view of development projects, one which recognizes the rework cycle, plans for it, monitors it, and helps managers reduce its magnitude and duration.
We need a method that reflects a more strategic view of projects, one which accounts for the quality of work done and the causes of productivity and rework variations. We need to be able to see more clearly than is allowed by "traditional" methods how changing external conditions and our own management actions alter staff productivity and the rework cycle—and how the consequences spread through an entire project. We need a new framework applicable across a range of projects, reflecting that which is common and that which is unique among projects. Only thus may we more consistently and rigorously extract, learn and apply lessons that will yield sustained improvement in project management and performance.

Such a new framework has emerged from the application of System Dynamics simulation methods to a wide range of development projects. In a manner quite dissimilar to CPM/PERT models, it treats a project not merely as a sum or a sequence of discrete tasks, but as flows of work in which there are multiple rework cycles. Because of the significant rework content of development projects, this framework is able to reconcile with one another a project's person-hours spent, tasks/items performed, elapsed time, and much more. Not only can the Rework Cycle model structure accurately simulate the actual recorded history of projects, but it can provide powerful forecasting and "What if" managerial capabilities.

First built for a ship design project at Litton4, the Rework Cycle model has since been applied accurately and successfully to over seventy different projects—defense electronics systems at Hughes Aircraft, telecommunication software systems at AT&T, aircraft design and production at Northrop, electric utilities' power plant engineering and construction, and dozens of other programs and projects.

At the core of the model structure is a different but straightforward view of project work—one which recognizes the rework cycle. Repeated applications of this more

realistic model have proven it to be logically correct and, when codified as a working simulation model, numerically accurate. Its uses have brought benefits (as in dollars) that are large (as in billions) to the businesses that have adopted it. This simple addition to the traditional view of projects has proved to be a powerful diagnostic and management capability.

The Traditional View of A Project

First re-casting the traditional view of a project's (or project stage's) tasks to be done, tasks in process, and tasks done as a more continuous stream of work...

![Diagram showing the traditional view of a project](image)

The pool of tasks in work to be done is depleted over the course of time, such that at the end of the project, nothing is left there, and all the tasks fill the pool of work done.

Taking this diagrammed "model" a small step further by recognizing that it is people working at some (varying) level of productivity that causes the work to get done...

![Diagram showing the traditional view of a project with productivity](image)

Changing the number of people along the way, or somehow influencing their productivity, alters the pace of work getting done.
Let us look at the shape of some key measures of project performance that would occur under this traditional view. Plotted below are graphs of "Work To Be Done," project (stage) staff, and percentage complete, as computed by a simulation model using the diagrammed structure...

These may show the way we plan the effort to go; this may be the way we hope things will go. But let's be honest–how many of us have seen even a remotely ambitious development project actually perform this way?

A Better View

A better view recognizes the existence of rework cycles. Below, what is termed the quality of work executed should be thought of as a "valve" controlling the portion of the work flow being done that will or will not require rework...

View the diagrammed structure as physical pools in which work resides and pipes through which work flows. It's easy to see that a "quality" measure that could vary (over
time) in the range of 0 to 1.0 diverts more or less of the work being done into the rework cycle. So long as this measure of quality is less than 1.0, some work being done--even rework itself--will continue to move into and through the rework cycle.

The distinction drawn between productivity and quality is important. Staff may exhibit high "productivity," but be putting out work of low "quality" that requires later reworking. In this condition, the net throughput to the pool of work really done is low.\(^5\)

The pool of rework requires staff to expend effort to execute it--to alter / correct/ complete the work items needing revision. With this addition to the model, different project performance would occur. As shown in the plots below, somewhat more familiar and realistic patterns are simulated when rework is generated and executed...

Recognizing the allocation of additional staff effort to execute rework, and the resulting slowdown in the pace of final completion, provides an accurate description of work on a project ... almost.

---

\(^5\) This distinction between productivity, quality, and rework has the added benefit of making all of these factors measurable and monitorable. Total throughput of work items in a project stage (lines of code, tons of steel, drawings, numbers of units, tests conducted, or ...) can be measured over time much as in traditional systems, and compared to the number of hours spent in the same time frames, so as to monitor a legitimate measure of productivity. Numbers of revisions and rounds of revisions, can be monitored over time so as to derive a tangible measure of quality, as described in "Benchmarks for the Project Manager" in the March 1993 issue of the Project Management Journal.
The Reality

In reality there is a critical "waystation" in which elements of rework linger until identified as *needing* rework. We have termed this waystation *undiscovered rework* ...

Undiscovered rework consists of those tasks or work products that contain as-yet-undetected errors of commission or omission, and are therefore perceived, *and reported by all traditional systems*, as being *done*.

The completed model of the rework cycle yields simulation-generated behavior that is characteristic of all development *projects* ...

The precise quantities and timing obviously differ among projects, but the behavior is common: as the initial round of work *nears conclusion*, previously undiscovered errors
become apparent, requiring more staff for a longer time; the perceived and reported progress significantly slows as the magnitude of recognized rework grows, and an extended completion effort ensues as the last elements of undiscovered rework emerge.

Most rework is discovered by "downstream" efforts or testing, but months (or even years) may pass before this discovery occurs! During this time dependent work will have incorporated these errors, or technical derivations thereof, and enter its own rework cycle. The more tightly-scheduled and parallel the project tasks, the more of a "multiplier effect" on subsequent rework cycles.

This final element of the rework cycle, undiscovered rework, plays a pivotal role in the propagation of problems through a project. Lurking undetected—as a software "bug," or design miscalculation, or wrongly-placed bulkhead—it causes productivity loss, delays, and more rework on dependent tasks. Undiscovered rework is the single most important source of project cost and schedule crises. It is the great killer of projects (and of new products and of careers), and no traditional systems even acknowledge its existence. Most projects seem to be planned and managed as though it does not exist. Not so for the distinguishably successful, however: then-program manager of Northrop-Grumman's "Team West" (to McDonnell-Douglas's prime role) for the new F/A-18 E/F fighter jet aircraft was Lou Carrier. He propagated throughout the program an awareness of the rework cycle, and the need to monitor and find rework early, and fix it. The "E/F" program became the first ever to receive the Department of Defense Excellence in Acquisition Award, and was described by Navy officials as “the model for all DoD procurement programs of the future.”

The specific technical content of undiscovered rework is by definition unknown at any point in the program. But it is imperative to program management success that undiscovered rework be:

---

6 The Northrop Grumman Review, Issue One 1995
(a) acknowledged (and plans and schedules set accordingly so as to reduce the
disruption of the "surprise");

(b) actively sought out (while earlier discovery may feel unpleasant at the time, it
is important not to "kill the messenger" -- rather, to encourage early technical
problem identification. Here, it is what you don't know that hurts you most.

(c) prevented as much as possible, i.e., improving "quality" in the rework cycle
(easier said than done, since managers cannot mandate by edict the
achievement of higher quality -- see the Third Failure for examples not to be
followed.)
3. FAILURE TO KNOW WHAT TO DO (AND TO DO IT):
Counterintuitively Counterproductive Countermeasures

The Third Failure is not knowing what to do as a manager of a challenging project, or when faced with the prospect of an ongoing project missing its targets. Whether it arises from poor planning, monitoring, prior managing, or any changed conditions, the prospect of poor performance means that managers will seek pre-emptive or corrective countermeasures. In most cases, the countermeasures will be obvious, intuitive, accepted, almost second nature... and wrong.

While we as managers have little absolute control, we have enormous influence on project performance. Our decisions and actions work through multiple cause-effect paths to affect the rework cycle and the project outcome. All too often, the countermeasures we use in response to typical problems along the way are counterproductive. Rather than helping, they aggravate conditions because we so often expect only the obvious impact, and so rarely anticipate the many other cause-effect paths by which the decisions influence the project.

As project managers, our most important decisions involve resource management: How much do how many of which resources work when on what, under what conditions? In the discussions that follow, we will describe the most common failings of project resource management and their many consequences.

*The $2,000 Hour*

We all know managers who have agonized over one or two percentage-point differences in salary or wage changes for individuals, amounting to less than a dollar an hour. The same managers will, without a second thought (indeed, without knowing) pay $2,000 or more for each effective hour of work, when that work is performed by project staff working extended *overtime.*
Figure 1 adds to the rework cycle structure a "path" showing a chain of cause-effect relationships, indicated by the thin arrows. Using information from the "pools" of work in the rework cycle (but without knowing, of course, the size or content of as-yet undiscovered rework), managers or their aides periodically estimate the progress made to date in the project, or in a stage of the project. Based on this, they assess the extent to which additional staffing is needed, to try to finish the remaining work on the prevailing schedule. A common and reasonable response to an indication that one is falling behind schedule is to supplement the effective staffing by the temporary use of overtime. This avoids the cost, hassle, and long-term commitment of bringing in additional people through hiring or transfer.

But a few weeks of overtime easily extends into more sustained overtime usage, as a key milestone approaches, or as the schedule gap stubbornly refuses to close (e.g., with the continuing discovery of more rework to do). As a purposeful choice, or because the "end" remains tantalizingly (and misleadingly) close, the overtime is continued. The intended direct effect (increasing the FTE people) is achieved. However, the secondary effect (through fatigue) on worker productivity and work quality, though often acknowledged in spirit, is consistently under-estimated (this path of unintended secondary influence is shown in the diagram with dashed-line arrows -- this is what causes the counterintuitive effect.)
Figure 1. Paths of influence from the Use of Overtime

We used our project simulation model to generate the performance of a typical four-year, million-person-hour (planned) project under two different conditions. In the first simulation was typical overtime usage, but only with the intended direct effect on staffing -- no unintended secondary effects on productivity or work quality. In other words this is an artificial -- and unrealistic -- world (but one that often seems to be the view of some managers). In the second simulation we activated the secondary influences of overtime, at the strength of effect that we have observed operating in dozens of development projects.

In this (more realistic) case, the per-person productivity drops and rework increases (by 25%), as a result of overtime-induced fatigue. Overall, the same project scope requires 15% more effort to complete when using extensive and sustained overtime.
All right, we know it's expensive, but we're trying to meet a schedule here!

Well, it might be worth the extra cost of the effort, if only it helped the schedule. Instead, though, the work requires an additional two months of time to reach completion, as a result of the added rework cycling.

From analyses of several projects, we calculated the cost of each effective full hour of project work output, when real productivity and rework "penalties" were taken into account. Figure 2 summarizes the results for typical engineering and production staff working sustained overtime (for 2-3 months or more) at a level of 4, 8, and 12 hours per week (per person).

Figure 2: Real Output Gained from Different Levels of Sustained Overtime

At a sustained level of 4 hours per week, both engineering and production staff achieve 1½ - 2 hours' worth of real, effective extra work output. If managers are paying wages for the overtime hours, perhaps even an overtime premium, the cost for each
effective hour of output achieved is staggering. In the most benign condition, achieving about 2 extra output hours for 4 overtime hours, each extra output hour costs about $150 (4 hours x $50/hour x 1.5 overtime premium = $300; $300 ÷ 2 output hours = $150/output hour). At 12 hours per week sustained overtime, engineers' extra 0.4 hour of output (1% beyond a 40-hour week) means an effective cost of over $2,000 per hour of gained output.8

In the production efforts analyzed, the "cross-over point" to no net extra work from sustained overtime of appears at about the 10-hour mark8; for engineers, the cross-over point is (just) beyond 12 hours. Regardless of the precise numbers, use of sustained overtime is far less productive than widely believed. Even "free" salaried overtime will eventually hurt schedule performance.

It's The Law

Unavoidable at the start or in moderation, bringing new people into a project organization (by hiring or transfer) has a degree of counterproductive impact that is, again, widely underestimated or ignored. But the impacts -- potentially harsh ones -- will occur nonetheless.

In the classic book *The Mythical Man-Month*, the author offers up "Brooks's Law": "Adding manpower to a late software project makes it later." Admittedly a

---

7 12 hours x $50/hour x 1.5 overtime premium= $900; $900/0.4 output hour = $2,250 per output hour .
Use your own company's wage rates and overtime pay premium to calculate your projects' real overtime output costs.

8 The most recent analysis we found elsewhere, a November 1980 report of a Construction Industry Cost Effectiveness Task Force, "Scheduled Overtime Effect on Construction Projects," published by The Business Roundtable, showed similar, even some more extreme, phenomena among construction workers. Interestingly, the same cross-over point, 10 hours weekly overtime (sustained for two months), was identified as yielding no extra real output; more overtime than that produces less than a standard work week.

simplification with exceptions, it still captures the spirit of a very real and damaging set of phenomena.

The diagram in Figure 3 shows another set of cause-effect paths of influence that work over the course of time, when managers decide to hire in response to a perceived need for additional staff. The obvious intent is to bring in new people who will supplement the staff available to work the project.

With any substantial hiring or even transferring of people unfamiliar with the project, the new people enter with less experience or skill than those already on board (despite individual exceptions, this holds true, especially for technically demanding work or specialized skill needs). So the average skill level of the growing staff drops, at least long enough for new people to "get up to speed" (which can take years). Worse, the more hiring occurs in any constrained labor market, the lower the entry skill level will be (assuming you're hiring the most appropriately talented first). It's as though you are hiring from a "barrel" of eligible candidates, and the more you take, the closer you come to scraping the bottom of the barrel.
As if that were not enough, there is an added "kicker." Newly hired individuals tend to have a higher attrition rate than longer-time experienced people -- from mistakes in hiring, recruitment expectations not met, less loyalty, wanderlust... A higher attrition rate requires still more new hiring, just to remain even -- let alone grow the staff more. So a vicious circle develops, a "churning" with little forward progress, while new people are hired, departure rates increase, and still more (brand) new people are hired to sustain the organization.

It gets worse still. Any problem response that in turn adversely affects productivity or quality, such as lowered skill levels from hiring (or sustained overtime fatigue, for that matter) can trigger a sinister set of self-reinforcing impacts on a circular
path of cause and effect. As the same diagram indicates, the hiring, which yields skill-
worsened productivity and quality, slows the pace of progress relative to the schedule
(especially as the additional rework generated by less skilled people is discovered). With
less than planned progress, we clearly need...what?: More people! (Thus more hires, new
people, skill dilution, productivity and quality reduction, slowed progress ...) Further,
there is the extra coaching and training time invested by experienced staff, and the
disruption of a rapidly growing organization. Lest you think this is an academic
concoction, be assured that we have observed this exact phenomenon in many
organizations -- it starts innocently and builds insidiously, becoming a trap into which one
can easily fall.

In engineering control terms, it's called a positive feedback loop ("positive"
because it's self-reinforcing), but there's nothing positive about its effect on the project
cost. Even the schedule-remedying intent can be thwarted by the addition of a cycle or
two of extra rework required as a result of the reduced work quality. Hence, Brooks's
Law... to which we can add the equally simplified and immodestly-named "Cooper's
Corollary": Adding (many) people to a late development project makes it cost more --
lots more.

When we activate all of these secondary effects in our typical project model, total
hours climb another 45%, and more rework accounts for nearly 2/3 of that increase.
Indeed, the rework effort grows to be nearly half of the total hours spent on the design.
The added rework cycling also extends the time of performance by another three months.
In a particularly vicious circle, the additional delay induces the managers to employ more
overtime, too -- thus incurring even more secondary impacts on productivity and work
quality (and cost and schedule).
Do You Have a Reservation?

True progress in development efforts is inherently difficult to assess, so, understandably, we have devised surrogate measures which are easier to monitor during the conduct of projects. We count drawings issued, lines or segments of code written, the “earned-value” of milestones met, or even simply hours spent, as approximations of true work progress made. The problem is that having been taught, or feeling obligated (and under pressure), to be precise, we consistently make the mistake of believing such estimates -- and taking action based on them. While they certainly are precise (793 drawings, 45.2%, ...), they are not accurate. It's like measuring time with a much too-fast digital watch -- quite a precise display of 4:52, but three significant digits are superfluous when you can't even trust the first to be accurate\(^\text{10}\).

In deciding how and when to staff, we too often depend on the precise but inaccurate "digital watch" when we assess the readiness of prerequisite work products to support the execution of the next dependent tasks. The "bean count" of prerequisite work done may indicate we can staff up on the dependent stage of work. But we need to take better account -- even if approximate -- of the quality of the work logged as "done" when making staffing decisions. The quality of that work -- the extent to which it will or will not need reworking -- is important to gauge, because staffing and working dependent tasks will incorporate the same errors, or derivations thereof, and thus create more rework cycling in the downstream efforts as well.

The prerequisite work may take any of several forms -- design information on drawings, or customer-supplied information or equipment, vendor products, software specs, steel framing, electronic circuitry ... In the case of engineering drawings, some

\(^{10}\text{Better to have a good working analog watch -- with just the hour hand! We offered up "Progress Ramps" as approximately accurate translators of monitored progress in "The Rework Cycle: Benchmarks for the Project Manager," Cooper, K. G., Project Management Journal. March, 1993.}
contractors annotate known missing segments or information with a "reservation."\(^{11}\) Despite the reservation, however, the drawing is counted as released, and the dependent efforts (within the design stages, or in procurement, or in production) are staffers as though the tally of drawings reflects work really done.

**Figure 4. Paths of Influence from Staffing on Low-Quality Work**

The diagram in Figure 4 displays this additional pathway through which management decisions affect projects and their rework cycle. In determining appropriate staffing levels, managers consider the apparent availability of prerequisite tasks' work products. But poor quality of that prerequisite work will cause unintended reductions in subsequent productivity and work quality. And the more aggressively the project (or

\(^{11}\) Literally, the word "reserved," or a similar notation on the diagram or document.
stage) is staffed in this condition, the more people and work that are subject to these adverse impacts.

When we add to the simulation of the same project these unintended but real productivity and quality effects caused by staffing dependent work while "quality" problems remain in upstream work products, project costs climb an additional 40% and completion time is four months later still. Rework is up disproportionately (it is nearly ¾ of the total increase in hours -- the rest from reduced productivity), now accounting for over 50% of the design effort. In other words, the simulation is now sufficiently realistic as to behave like most difficult development projects.

Those of us to whom project managers report, whether inside the organization or as paying customers, are often the most guilty parties. Eager to see progress on an important development, we commit one of two serious errors ...

...One: We micro-manage, instructing the project manager to staff up (prematurely) -- not knowing any better ourselves. Or we effectively do so by criticizing the manager (or as a customer, the manager's executive) for failing to staff up (enough/according to plan/in the face of missing a near-term schedule milestone). We assert that the low staffing will "jeopardize the future of the project," and we lay the basis for, or induce the fear of, retribution. Retribution can take many forms -- job loss or demotion, delayed payments, disallowed costs, the prospect of losing future business, intransigence on a variety of to-be-approved or negotiated items, even legal claims. All this when the smartest, best thing to do for the project cost and final completion schedule is to restrain staffing temporarily.

...Two: We create an environment or relationship through intimidation in which the project manager, if he or she wishes to remain employed, knows better than to "own up" to prior work not really being ready. Thus, the project manager staffs as though the prior work had no problems.
Your Parents Were Right All Along

**True story:** A group of bright young managers were in a company training course to prepare them for the rigors of taking on full program management responsibilities. Confronted with a scenario in which the program they were "managing" was falling behind schedule, they were asked about their responses to this prospect. The glee in their eyes was visible, their excitement palpable. Throughout the responses was the nearly unanimous feeling, expressed by one young fellow who had observed and absorbed the way things really work, "We really turn the screws on the engineers."

**Figure 5. Paths of Influence from Schedule Pressure**

![Diagram illustrating the paths of influence from schedule pressure.](image-url)
Not everything we do as managers involves a tangible action. What we say and the "incidental" gestures we make influence the people around us, and the people who work for them, more than we know -- or is it exactly as we know? To exert schedule pressure on those around us is a natural, and nearly universal, managerial response to lagging progress. A little bit is good, sharpening the senses and increasing productivity. But like unsolicited criticism, a little goes a long way.

The diagram in Figure 5 demonstrates the intended path of influence of schedule pressure -- an increase in productivity. It also shows the unwanted side-effects. At some point in your childhood, someone, probably your mother or father, told you that "haste makes waste" -- usually after you had rushed through something only to find you had made a mistake that required you to re-work it with more effort than would have been required if you had taken the care to do it right the first time.

Despite the slogan posters prevalent on the walls of project offices ("If you don't have time to do it right the first time, where will you find the time to do it again?"), it doesn't seem we've learned the lesson our parents tried to so hard to deliver. "The harder I work, the behind I get" is the worker's response to the slogan posters. But work hard (and fast) they do, knowing managers want to see product. The quality impact, (the later need to do it again) is far less visible, but no less real.

Equally real are longer-term impacts of sustained pressures. Every experienced manager has seen the considerable morale impacts when the staff is pressured from all sides to improve not only on the schedule, but also the costs and the quality as well.

Beyond these secondary effects on productivity and quality, Figure 5 depicts yet another cause-effect pathway of influence often activated by schedule pressure. In such a highly-pressured condition, engineers (or programmers or ...) are induced to work on something -- anything -- that will demonstrate "progress" is being made. This usually translates into working more and more out-of-sequence, that is, on items that plans or
cold logic indicate should be done later in the work sequence, when more of their prerequisites -- products or information -- would be available. "But surely parts of those items could be worked now [even if we aren't quite as efficient in doing them], and, why, we could even release those items [so they will add to our bean-count, even if they need to be re-worked later]." So, with this familiar refrain, of which we are nearly all guilty, we trigger the desired short-term apparent progress gains. And we set the stage for even more trouble later when the resulting rework gets recognized.

Once more we'll add these effects, at the strengths we have seen them operate in many development projects and programs, to the simulation model. The results should be no surprise. That young manager-in-training and his turn-the-screws plan, though typical, hurts final project performance far more than any temporary gain, through its unintended effects on the rework cycle.

The poorer performance generally shows up in the latter half of the effort, as the pressures mount well into the project. Rework effort now accounts for about 60% of the total hours spent on design, and causes the vast majority of the 100,000-hour increase.

Indeed, all of the previously cited effects are aggravated as the project performance has become worse (more real) -- part of what one of our clients has termed a "death spiral." Oddly, some find that term more descriptive than the "mutually-reinforcing interconnection among several causal feedback loops."

How far we've come. It was such an innocent-looking project at the start. Each step along the way added some more troubles and realism. The "normal" managerial responses, all aimed to exert control, brought unintended side-effects and penalties through paths of influence generally not well-understood, or at least under-estimated.
Figure 6 helps illustrate just how far we "progressed." We added 600,000 hours to the design effort (6a, "X" plots show the original conditions, "P" plots the final) through productivity losses (6b) and more rework (6c). Especially rework -- it alone increased by 500,000 hours. A plan-beating staff profile that had peaked in mid-1992 at 150 people became a serious overrun, with a second peak at 250 people, a year later.

By mid-1993, when the build effort that was using the design product was under way, the original simulation displayed a perception of design readiness (7a) near 95% (and accounting for the undiscovered rework, really was at 75% -- see 7b). The final, real project simulation displayed a perceived design readiness of 85% at the same time (7a), but much more undiscovered rework meant the design product was really only 50% complete (7b).
While we've focused on the design effort, the build (construction/production) effort has been being simulated all along. And the difference between the two extreme conditions, in terms of engineering readiness and the magnitude of design changes, is enough to cause more than a doubling in build labor costs (7c) -- and a full year's slippage. Rather than comfortably beating the budget, the labor costs were more than twice the budget.\textsuperscript{12}

\textsuperscript{12}And this does not count the added costs of any "marching army" -- i.e., "support" staff incurring more expenditures as such level-of-effort functions are continued for a longer time -- nor does it count any increased unit costs for another year of inflation.
Doing It

If we do know what to do, we certainly are not doing it. Having spent much space diagnosing why common countermeasures fail, let’s summarize what to do, and ask why we do not.

Do…

… Restrain the use of real overtime (reported or not) to short bursts — less than 25% for no more than two months.

… Hire and use only the most qualified people — let your competitors have the rest; yours will more than pay back the premium you pay.

… Schedule and plan the phases of work so as to achieve time separation between technically-dependent elements. Otherwise, undiscovered rework will thwart the attempted acceleration, and add cost.

… Resist pressures to build up staffing at the start of work or work phases faster than you can efficiently deploy them to ready tasks (you will save enormous rework effort later).

… Count on those tasks not being as ready as you think, if they are at all dependent on prior complex work or design activity — there is undiscovered rework lurking.

… Find and fix the undiscovered rework as a top priority — the earlier the better. This is one of the few purposes to which extra people and systems and tools and money can be put to universal benefit. Encourage rework discovery!

… Cut the links from schedule pressure (all projects have it) to productivity and quality: Set clear priorities among work quality, cost, and schedule (preferably in that order — all will benefit); Discourage taking work shortcuts and working out-of-sequence; Prohibit the release of incomplete work products to (and their use by) dependent work elements.
Use systems and procedures that acknowledge rework. Behave as though preventing rework is your top priority. It is. A close second is finding the inevitable rework that will still exist.\footnote{This is precisely why well-implemented “integrated product teams” are producing such good results. Representative toolers and builders who would normally wait to receive design product, for example, are actively engaged early in the design, helping to prevent, find and improve what would otherwise linger as undiscovered rework.}

These represent a cohesive and consistent set of good things to do in managing a project but they are \textit{not} common practice (despite being common sense.) Why not? There are just two general possibilities: (1) we know what to do, but lack the courage or incentive to do it; or (2) we have not learned what to do. We have seen both conditions among the many organizations we have analyzed. There are certainly some individuals who know what to do, operating in some companies and project settings which discourage taking the right managerial actions, and opt instead for short-term (perception of) progress. Such individuals are like candles in the night -- a windy night. It takes not too many being snuffed for others to get the message.\footnote{In this setting, managers can fear knowing “too much”. I have had project managers (happily a small minority) express their wish to use the rework cycle modeling capability, but explain that they could not because they would then “know too much” about the project’s future (and need to disclose it, and get fired...). The concept of \textit{foreknowledge} and the power of pre-emptive action eludes such dolts. They shall remain anonymous -- they get fired later anyway.} Even in this setting, however, the real failure is the organization management’s or customer’s lack of having learned the magnitude of gains achievable by taking the correct, even though short-term painful, actions.

So it is that by far the most common condition that prevails throughout all kinds and sizes of organizations and projects, among all levels of management, is the failure to learn. Without the failure to learn from past successes and past problems, the first three Failures would no longer exist.
4. FAILURE TO KNOW WHAT’S WHAT: Lessons Not Learned

The failure to learn is the most pervasive failure in all project management, even among organizations dependent on successful projects for income, new products and services, new processes and technology, and thus competitive advantage.

Four different conditions have contributed to and perpetuated this failure. *First* among these is the misguided belief that all projects are different, *that there is little to learn from others.* (A common version of this appears in the near-unanimous belief, "but *my project* is different.").) The roots of this misconception are in the genuine lack of understanding of the very common dynamics and phenomena shared by virtually all projects (Paul Revere, meet Wall Street).

*Second*, projects are by definition a transient phenomenon, with a start *and* an end. Each is time-pressured. We are rushed to start, rushed to end, and rushed to start again. Put the past behind us. Failure? No need to point fingers, unpleasant to re-live the disaster anyway. Success? Lucky. Easy customer. No challenge. Over-budgeted. Heroic individual effort. There are few companies (there are a few, and they are successful -- the software world is beginning to show how, such as at Microsoft) where there are organizations and money and systems and practices that span projects, for the very purpose of gleaning and improving upon transferable lessons of project management.

Even in the absence of organizational learning systems, there are individuals who learn, right? Great project managers might have three or four great projects that make up their career before they move to different responsibilities or retire. From this we should expect systematic assessment and learning of transferable lessons that get incorporated in subsequent projects? This limited span and career path of good project managers is the *third* contributor to the failure to learn.
The **fourth** and final contributing condition is the lingering lack of esteem in which project/program management has been held historically. (Some would assert this is an earned perception, but that is from lack of performance, not lack of importance.) There are no corporate-ego-level mergers or takeovers here, no big-ticket sales. Few business schools feature project management -- the exciting worlds of accounting and finance, organization theory, production management, marketing, international business, business law, statistics, yes -- but precious little on the discipline of project management, which requires all of the aforementioned subject skills and more.

Some organizations have begun to recognize that project management *is nothing less than the lifeblood of change*, the channel of innovation, the definitive source of competitive advantage from which all new refinements, products, services come to market to create income growth, shareholder value, and societal advancement. Again, look to the software industry, *built solely on projects* -- projects to define, develop, and market new items. Should one ever doubt the value of such projects, take note that Microsoft alone now has a market value that exceeds that of General Motors, Sears and Westinghouse Electric...*combined*.15

**SUMMARY**

We have discussed these four systemic project failures:

1. **Failure to Know What to Expect...**

   ... Excessive optimism and lack of discipline in defining the product, the cost budget, and the work schedule set the stage for disruptive change, counterproductive pressures, and management actions that worsen performance beyond the inaccuracy in the plans themselves.

15 *Business Week*, July 8, 1996.
2. **Failure to Know What to Watch...**

   ... Understanding the rework cycle in our projects is a fundamental requirement for accurate monitoring, progress assessment, and decision-making -- yet it is absent from all conventional project management systems.

3. **Failure to Know What to Do (and To Do It)...**

   ... Lack of understanding the multiple cause-effect paths through which our decisions cause "secondary" impacts on productivity, rework, and total project performance leads us to take common counterproductive management actions.

4. **Failure to Know What’s What...**

   ... This failure -- the failure to learn -- is most critical, as it perpetuates the other failures, and denies managers and organizations the understanding of the need for, content of, and impact achievable from systemic project management improvement.

Those organizations that do not overcome the Four Failures will themselves fail in a competitive world. Those that do will thrive and grow. To accomplish this, the winning organizations will develop and deploy learning systems for project management. These systems will be used to: cull lessons from past projects; disseminate those lessons via training current and prospective project managers; monitor ongoing projects for deviations from expectations; dispassionately and rigorously diagnose the deviations to determine positive and negative lessons and affirmation of prior lessons; feed back that diagnostic information for analysis and subsequent dissemination; plan and schedule new projects based on the rigorous past-project performance analysis; educate internal and external customers; and build a culture that incentivizes and rewards early problem detection, pre-emptive management, and ongoing lesson transfer and improvement. Organizations operating in this project management learning system will see success build upon success, and increasingly distance themselves from their failing competitors.