Development Inefficiencies - Managing The "Rework Cycle" In Complex Projects

- Peter Lancy, PA Consulting Group.

The aim of the project was to design and build an exciting first-of-a-kind product, but during the development effort, unexpected problems emerged - changes in the design specifications, shortages of qualified people, and delays in materials supply. 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 more than double the original budget. If this tale of development project woes sounds all too familiar, perhaps from personal experience, or because others like it are lamented so frequently in company boardrooms, and reported in business publications with disturbing regularity.

It might be a construction firm's experience in building a power plant, or a software company's troubles in bringing new systems to market, or the most recently reported defence programme problems, or the automobile industry's struggle to cut product development time. In fact, the product sponsor in this tale 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 USS Constitution inaugurated the US Navy and was, in 1794, a new nation's introduction to the challenge of managing large development projects. Today the challenge, in the US and throughout the rest of the world, remains largely unanswered.

The simple fact is that the traditional art/science of the management of complex development projects is a failure. The consequences of this failure are enormous:

- contractors and other businesses lose vast sums of money on development efforts;
- legal disputes over contracted responsibility for project costs drag on for years;
- millions of dollars are capitalised unnecessarily on company balance sheets;
- other projects are denied scarce resources;
- products are late to market, at higher prices and with fewer features than planned and promised;
- market shares, jobs and profits are lost.

For decades, there has been stagnation in the prevailing theory and practice of large project management. "*Advances*" have been largely confined to the computerisation of fundamentally inadequate methods. Nothing short of a fundamental change in the way managers view development projects is required. To avoid the persistent cost and schedule performance problems so closely associated with complex developments, we must take a more strategic and realistic view of project work content and processes. We must recognise that while the products and technical steps may indeed be unique, there *are* common structures and processes, and common causes of problems. From these, it is possible to extract lessons and to implement changes that achieve radical improvements in project performance and business success.

TRADITIONAL METHODS OF MANAGING LARGE PROJECTS ARE FUNDAMENTALLY INADEQUATE

Imagine that you have contracted for your house to be built, and that the standard tape measure used by all the contractors has long sections - half the tape, in fact - uncalibrated. We are, in effect, equipping project managers with half-blank tape measures as the standard tools for planning, monitoring, and managing large projects. At PA Consulting, we have analysed over 60 major development projects and programmes in aerospace, large construction, software systems, shipbuilding, defence electronics, and telecommunications. When we began doing so over ten years ago, we were asked by our contractor client to build a computer-based model capable of simulating the performance of a large project accurately from the start of design to the completion of construction. Large portions of the project effort simply could not be described or explained by applying the standard available tools.

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. If one element on the path were to be delayed, it would create a delay in the entire project. It is an accepted, often required, technique for planning projects and for testing impacts on schedules. 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, is an inadequate model for managing complex development projects.

What is missing, as any experienced project or programme 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 products, or the amount of rework that 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 revisiting 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, for example, in the release of initial engineering drawings, "A" revisions, "B" revisions, "C" revisions and so on (for those companies that actually monitor such information).

Companies experienced in complex projects know to expect this, and have developed rules of thumb to count on two (or three or four) revisions per engineering product. Even so, this expectation is rarely incorporated explicitly in work planning and management systems, because the techniques do not 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 are reading it with their eyes closed. These are not unintelligent people nor project-naive companies; on the contrary, they include the most technically sophisticated people conducting and managing complex developments in firms whose very existence depends on successful project performance.

THE ADDITION OF THE REWORK CYCLE TO THE TRADITIONAL VIEW OF PROJECT WORK PROVIDES A POWERFUL MANAGEMENT CAPABILITY

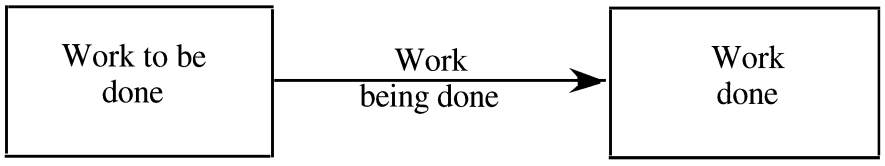
What we need, then, is a different view of development projects - one that recognises the rework cycle, plans for it, monitors it, and helps managers reduce its magnitude and duration. We need a method that reflects a more *strategic* view of projects - one that accounts for the *quality* of work done 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 in this way 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 a project's man-hours spent, tasks/items performed, elapsed time, and much more. Not only can the Rework Cycle model 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 Litton[1], the Rework Cycle model has since been applied accurately and successfully to over 60 projects - software system developments at AT&T, defence electronics systems at Hughes Aircraft, the cross-Channel tunnel, electric utilities' power plant engineering and construction, and dozens of other programmes and projects. 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 enormous benefits to the businesses that have adopted it.

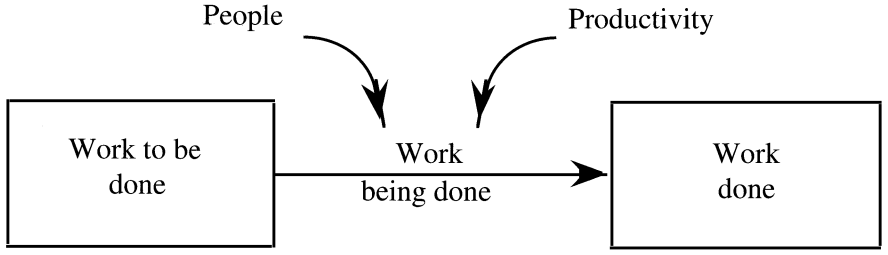
The traditional view of a project as stages of "*work to be done*", "*work in process*", and "*work done*" may be seen as a more continuous stream of work in which the pool of tasks in work to be done is depleted over the course of time, such that at the end of the project, nothing is left there, and all the tasks fill the pool of work done (Figure 1).

Figure 1 A project may be seen as a flow of work



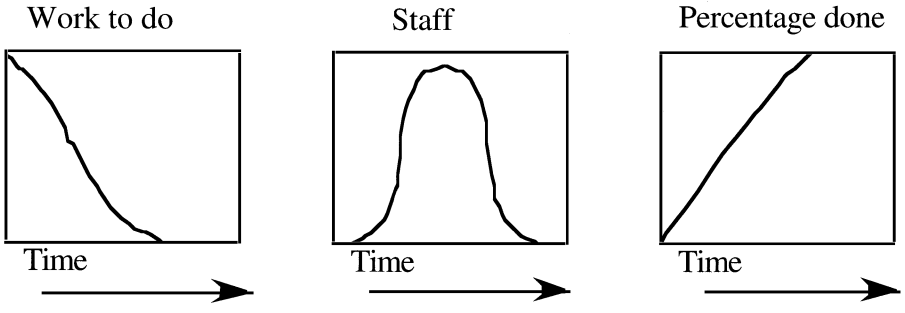
Since it is people working at some (varying) level of productivity who cause the work to get done, changing the number of people along the way, or somehow influencing their productivity, alters the pace of work getting done (see Figure 2).

Figure 2 Productivity affects the pace of work flow



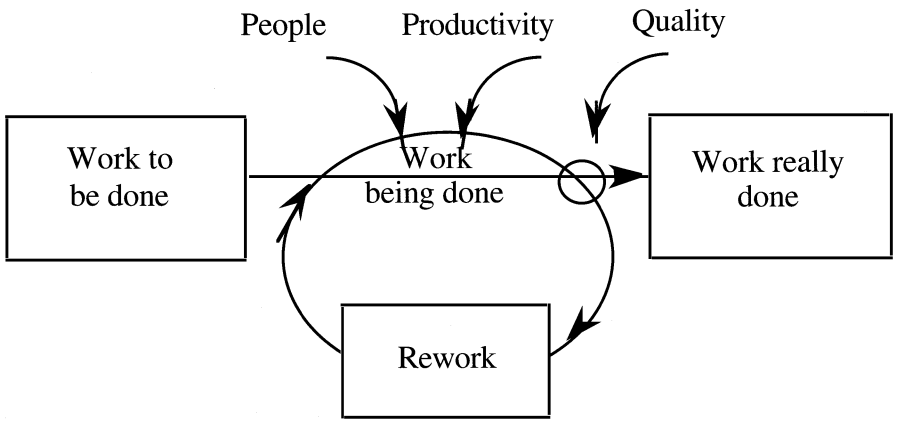
Under this traditional view, key measures of project performance - work to be done, project (stage) staff, and percentage complete - as computed by a simulation model, might look like those illustrated in Figure 3. These may show the way we *plan* the effort to go; this may be the way we *hope* things will go. It is rarely, however, that even a remotely ambitious development project actually performs this way.

Figure 3 There are "idealised" shapes to some of the key performance measures



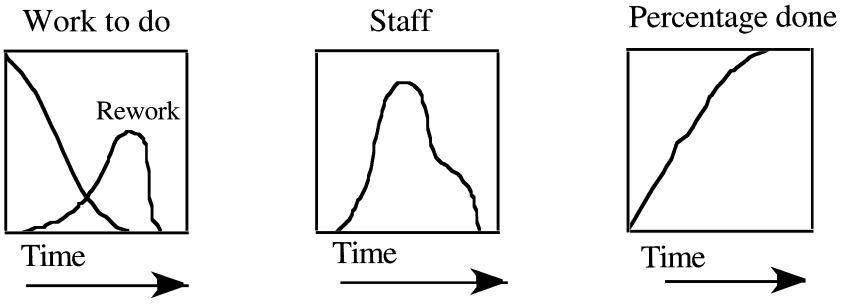
If we view the process as physical *pools* in which work resides and *pipes* through which work flows, it is easy to see that a "*quality*" measure, acting as a *valve*, and varying (over time) in the range of 0 to 1.0, diverts more or less of the work being done into the rework cycle (see Figure 4). 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.

Figure 4 A quality measure diverts work into the rework cycle



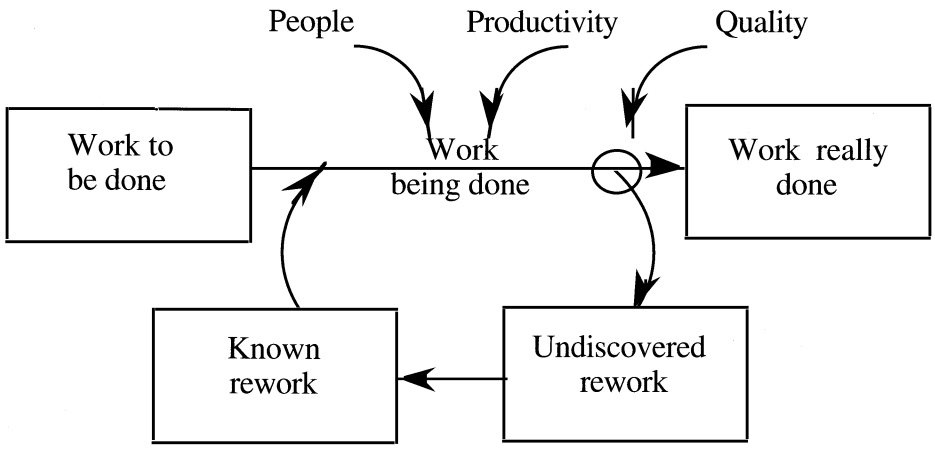
The distinction between productivity and quality is important. Staff may exhibit high productivity, but be producing work of low quality that requires later reworking. In such a case, the net throughput to the pool of work really done is low[2]. The pool of rework requires staff to expend effort to execute it - to alter/correct/complete the work items needing revision. With this addition to the model, project performance would be different. As shown in Figure 5, somewhat more familiar and realistic patterns are simulated when rework is generated and executed. Recognising the allocation of additional staff effort to execute rework, and the resulting slowdown in the pace of final completion, provides a more accurate description of work on a project.

Figure 5 More realistic patterns of performance measures are simulated when rework is generated and executed



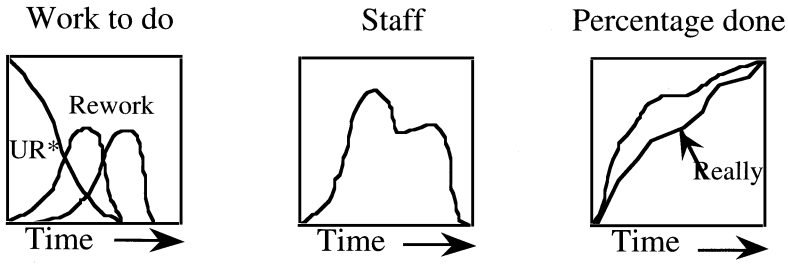
In fact, however, there is a critical "*way station*" in which elements of rework linger until identified as *needing* rework. We have termed this way station "*undiscovered rework*" (see Figure 6). Undiscovered rework consists of those tasks or work products that contain as-yet-undetected errors of commission or omission, and are therefore perceived, and reported by all traditional systems, as being done.

Figure 6 With undiscovered rework taken into account, the rework cycle is complete



The completed model of the rework cycle yields simulation-generated behaviour that is characteristic of all development projects. The precise quantities and timing obviously differ among projects, but the behaviour 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 down as the magnitude of recognised rework grows, and an extended completion effort ensues as the last elements of undiscovered rework emerge (see Figure 7).

Figure 7 With the full rework cycle acknowledged, simulated performance is characteristic of all development projects



* Undiscovered rework

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 simultaneous the project tasks, the more of a multiplier effect there will be 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 a design miscalculation, or a 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. Even more important, most projects seem to be planned and managed as though it does not exist.

The specific technical content of undiscovered rework is, by definition, unknown at any point in the programme, but it is crucial to programme management success that it be:

- acknowledged (and plans and schedules set accordingly so as to reduce the disruption of the "*surprise*");
- actively sought out (while earlier discovery may feel unpleasant at the time, it is important not to "*kill the messenger*" but rather, to encourage early identification of technical problems;
- prevented as much as possible – that is, improving "*quality*" as defined here (easier said than done, since managers cannot mandate by edict the achievement of higher quality).

THERE ARE BENCHMARKS FOR THOSE WHO SEEK TO MANAGE THE REWORK CYCLE

Across the variety of projects that we have examined[3], and within certain groups of projects, patterns of behaviour have emerged that can be useful as managerial rules of thumb. While each project is, indeed, different, the empirical observations and measurements reported here are valid benchmarks for all those who undertake to manage the rework cycle in their own projects. These relate to quality and to the timing of the discovery of rework. With these measures, it is possible for a project manager to assess the true status of a project.

Moderate improvements in quality have a dramatic effect on performance

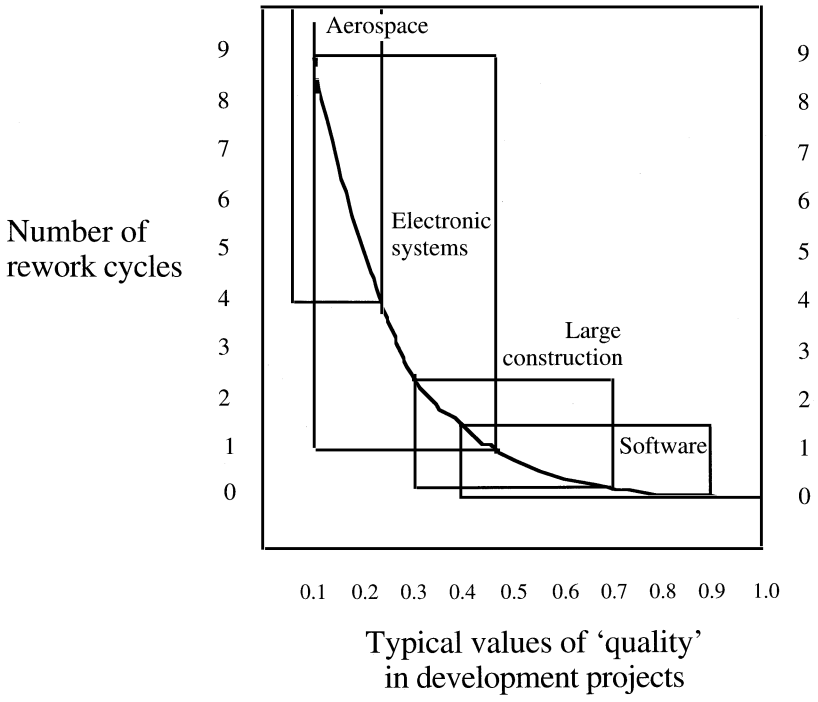
To underscore the magnitude of the issue and the appropriate degree of managerial concern, consider the range of values exhibited for "*quality*"- the fraction of work being executed that will not require subsequent rework.

As indicated in Figure 8, the effective values range from 0.90 to below 0.10. The figure shows the range of quality values for each type of project[4], and the resulting number of rework iterations.

Of the projects examined, commercial software development projects exhibited the lowest amount of rework (i.e. the highest quality), ranging from little rework in some stages to $1\frac{1}{2}$ full rework cycles in others. Many such projects are actually adaptations of prior developments.

At the other extreme are aerospace development programmes - all of which are advanced military developments, usually with substantial research efforts - which typically have at least four (and usually more) rework cycles in the design effort. Between these two extremes are electronic systems development projects (typically designing and integrating systems of new hardware and new software) which exhibit between one and nine full rework cycles, and the design of large-construction projects (with a range of about - to $2\frac{1}{2}$ rework cycles[6]).

Figure 8 There is a wide range of values for "quality" - the fraction of work being executed that will not require rework



This figure shows the empirically derived value of quality for dozens of real development projects in different arenas. Along with the range of quality for each industry segment are shown the rework cycles implied by that range. As lower quality causes more cycles of rework, more effort and cost must be expended to complete the work - the equivalent of doing it once, and then doing it all over again and again. A quality of 0.20 will require five cycles of work and cost (four rework cycles) to get it right. Development projects here are those that are dominated by design/ engineering activity to develop a new product or system, and do not include the building of units to a stable design.

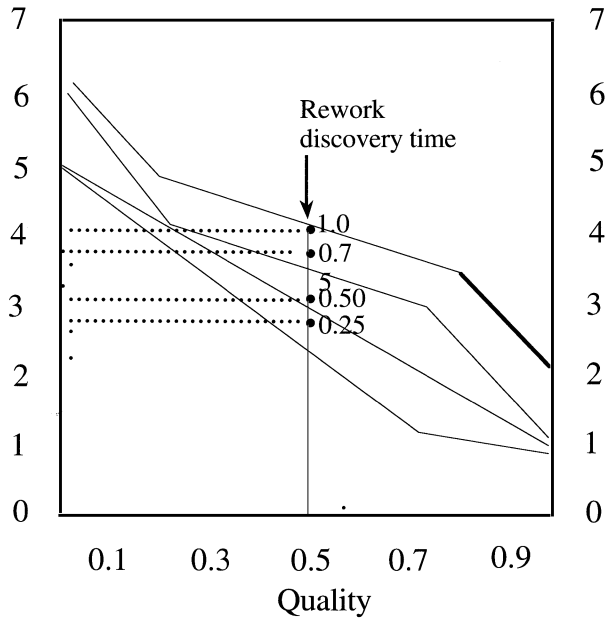
Clearly, the larger the technological leap being attempted by the overall project, the lower this measure of quality will be and consequently the more rounds of rework required. However, even within the range of each project type, there is considerable variability in work quality, and therefore in the cost and time required in a development project. Consider the project performance improvement achievable with even moderate improvements in quality. In an era when 30 to 40 per cent performance improvements are set forth as ambitious targets for organisations undergoing change programmes and process re-engineering, the highly-leveraged impact of *this* kind of quality improvement must not be overlooked.

Discovering rework early has a significant impact on development time

Regardless of the effort and impact of the quality improvement, undetected errors and rework cycles are unavoidable in complex development projects. Whatever rework is generated, it has its most destructive effect on the whole project when it is in the state of "*undiscovered rework*". Discovering the rework earlier and faster removes much of the programme-wide disruption, especially the impacts on development time.

To illustrate this, we used our model of the rework cycle to simulate the development time required for a project under varying levels of quality *and* the speed of rework discovery. Figure 9 summarises the results in terms of the multiple of the original work schedule. (For convenience, think of a development effort planned to achieve initial work completion in one year – the results shown in Figure 9 would then be "*number of years*"). The figure shows four lines - for rework discovery times equal to one-quarter of the original design plan time (one-quarter year in our example), as well as discovery times of half, three-quarters, and one.

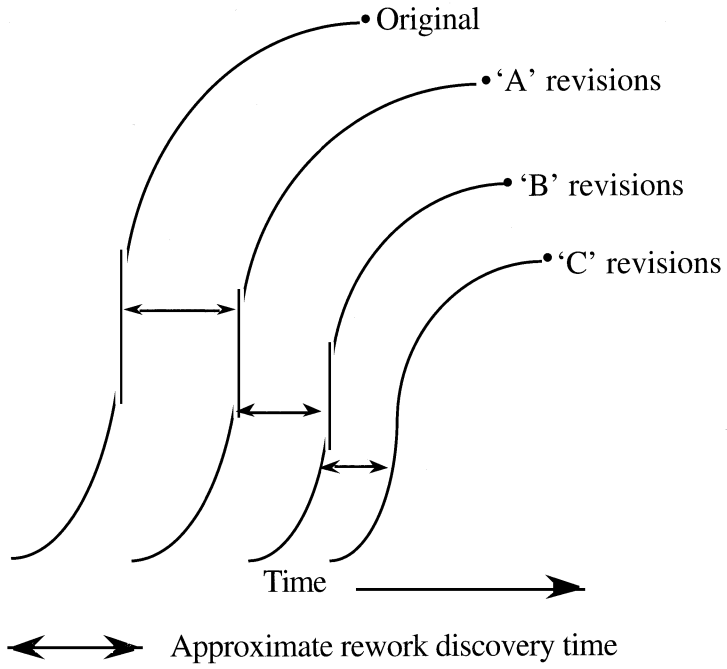
Figure 9 For most levels of work quality, improving rework discovery time yields significant improvements in development schedules



As is now obvious, improvement either in quality or rework discovery yields much better schedule performance. It is noteworthy from the results charted that lowering the rework discovery time in an organisation or project is most leveraged in improving schedule performance when quality is *not* at extremely low or extremely high levels. At extremely high quality levels, there simply is not as much room for improvement of schedule performance, and in the stages of a development when extremely low quality prevails, rapid rework discovery ends up subjecting the execution of the discovered rework to the same low-quality conditions that caused it to cycle in the first place. In such conditions, it is best to work first on quality enhancement practices and systems, then to accelerate the benefit with rework discovery enhancements, such as earlier or improved reviews or testing.

The prevailing rework discovery time on design development efforts is typically in the range of one-quarter to three-quarters the scheduled length of the original design effort. In order to derive a good approximation of rework discovery times, construct a graph of the issues and re-issues of work product in the stage of development being examined (historically, if available; otherwise, monitor an ongoing effort). Plot, over time, each subsequent round of revision as a line. Simplified, your chart should look like Figure 10. By measuring the typical horizontal "*distance*" (time) between each adjacent pair of curves, good estimates of the rework discovery time (and how it changes over the project/stage) may be obtained.

Figure 10 A critical addition to the set of closely monitored performance measures should be the magnitude and timing of work revisions



Now, you can also compute a good approximation of the time-varying quality of a project stage's work product. Calculate the ratio of (a) the number of revisions to (b) the number of releases/revisions in the prior round; then subtract each computed ratio from 1.0. For example, if there were 350 "B" revisions on 500 "A" revisions[5], the prevailing quality during the work on "A" revisions was 1 – (350/500) = 1 – 0.70 = 0.30

With measures available, it is possible to assess the true status of a project

Armed with this new information, you will be able to assess where your project stands relative to other development projects. Further, you will be able to determine much more accurately where your project really stands as it proceeds.

We used our simulation model of the rework cycle to construct Figures 11 and 12, which are necessary for more correct mid-project assessments of progress and the magnitude of remaining effort. Using your estimates of project quality and rework discovery time to select the appropriate chart, you may "*look up*" the true (range of) *real* progress. In fact, with the benefit of data on a completed project, one may construct one's own "*progress ramp*" chart by plotting, for the completed project, (1) the historically reported "*percentage complete*" versus (2) a retrospective computation of the percentage really complete. (You should compute the percentage really complete based on hours spent up to that point, relative to the total hours eventually spent.) Perfectly accurate project progress monitoring would yield a straight 45-degree diagonal (hence the triangular ramp shape): at a perceived/reported condition of 20 per cent complete, the *actual* percentage complete would be 20 per cent, and so on. Instead, real progress is typically less than reported progress. Further, a given level of reported progress might mean any of a range of values for real progress.

Figure 11 Progress ramps help to identify the true state of progress on a project or a project phase

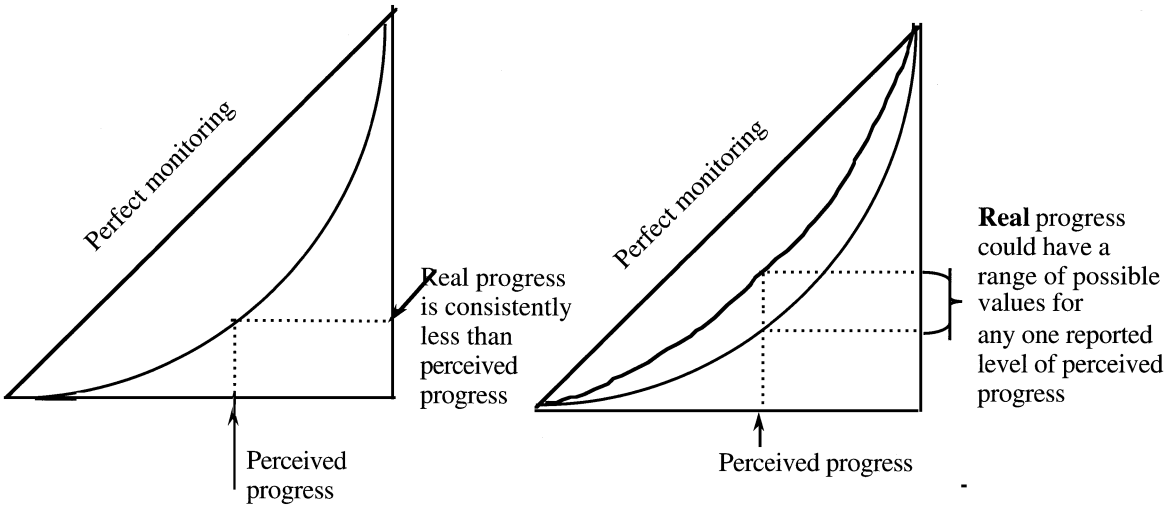
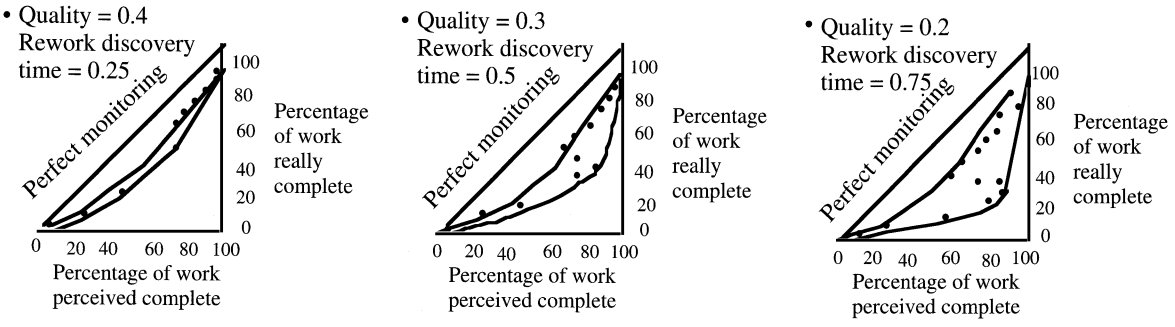


Figure 12 shows, for each of three typical combinations of quality and rework discovery time, the relations between *perceived* percentage complete, as reported by traditional systems, and the *real* percentage complete, when undiscovered rework is taken into account.

Figure 12 Lower quality and longer rework discovery times disguise low real progress, and increase the range of uncertainty in estimating progress



In every case, there is a gap between real progress and that which is perceived. Looking across the three charts, it becomes clear that the lower the quality and the longer the rework discovery times:

- the larger the gap between real progress and that which is perceived, and the longer-lasting the gap;
- the later in the project/stage that a significant gap persists;
- the greater and longer-lasting the uncertainty in the size of the gap;
- the later the point of maximum uncertainty abut real progress.

Uncertainty in real progress is responsible for several troublesome project phenomena

The demonstrated uncertainty in real progress is responsible for several well known troublesome project phenomena - the 90 per cent syndrome, the lost year, and the delayed introduction of the product.

The 90 per cent syndrome

The "*90 per cent syndrome*" occurs when, for a prolonged time, project managers report to executives or the customer that their effort is "*90 per cent*" done. Examine the middle progress ramp for an example of this. In these conditions, an earnest, responsible manager might well report 90 per cent progress achieved when, in fact, the real progress level is as low as 70 per cent. Some more work gets done, some rework is discovered, and later, the manager still perceives and reports that about 90 per cent of the work is done, when 75 or 80 per cent is really done. And so it goes on, until, after much distress and disappointment (and time and cost), 90 per cent is really achieved and the project moves on to completion. Quite apart from any lily-gilding inclinations of engineers, there is a systemic cause for the 90 per cent syndrome.

The lost year

One of our clients, managing a large new system development project, described a related phenomenon, and requested a diagnosis. The 1,000-person development effort had progressed to what seemed about two-thirds to three-quarters completion. One year later, after an additional 1,000 staff-years of effort, it seemed that they had made no progress, or had even lost ground, followed by a slow pace of progress towards completion. He called it the "*lost year*". His question: "*What happened?*" The abbreviated answer is clear, once again, from the middle progress ramp. In these conditions, 70 per cent progress could be reported as "*early*" as at 30 per cent real progress. Quite some time later (specifically, one year), having made 25 to 30 per cent more real progress, "*the system*" can still be reporting 70 per cent progress, or even less.

A lot of progress was made. A lot of rework was found and corrected (work that *had* been viewed as done). The "*lost year*" was not lost; it obviously felt like one step forward and one step back, but it was just a large example of the rework cycle thwarting conventional monitoring systems. Even after the "*lost year*", the project, still being reported at about 70 per cent complete, was in fact just 60 per cent complete, so what was seen as the remaining 30 per cent felt as if it took a long time to finish.

The delayed introduction of the product

Not so very uncommon was the dilemma faced by another client firm. They wanted to introduce their new system product at the earliest responsible time in a technologically competitive market, but they had a history of undependable announcements on product release schedules, followed by dashed expectations. When could they really promise the market that this new system product would be available?

The diagnosis revealed a somewhat higher quality than in the prior charts, but with a longer rework discovery time, as characterised by the third progress ramp (shown in Figure 12). In this condition, the true status of a development project remains highly uncertain until the very end. Combined with consistent over-estimating of progress are wide (vertical) bands of uncertainty within which *perceived* progress may become unexpectedly "*stuck*". While real progress *is* being made, the late and prolonged appearance of previously undiscovered rework results in ever-sliding, undependable estimates of completion time - particularly at the later stages, when companies are making announcements of new product introduction schedules.

This client's organisation and practices had been set so as to encourage high levels of "*completion*" before subjecting to testing the integrated prototype. The analysis led the client to implement a new structure and set of procedures that included much more earlier testing. While this increased testing costs, total project time and costs were reduced significantly. Although the resulting earlier rework discovery was a bit "*scary*" to the managers at first, much more dependable introduction schedules were attained.

SUCCESSFUL PROJECT MANAGERS WILL BE BOTH ADVOCATES AND INSTRUMENTS OF CHANGE

Even with "*a better model*", there will remain a managerial inertia forged from years, even decades, of misunderstanding. It will not be enough for project managers to internalise the associated lessons of the rework cycle. They must also learn the important differences between:

- the substantial influence that the manager has over productivity, quality, and rework discovery, and hence project costs and schedules;
- the relative lack of control that the manager has over these same conditions.

The very best of project managers know this intuitively, but just as important they must ensure that their formulated action plans and associated logic are effectively and persuasively communicated to (and implemented with the aid of) many other players - senior company managers and colleagues, the project staff, partners and suppliers, and customers themselves. The passive manager (or worse, those who encourage obfuscating or ostrich-like behaviour) will not make waves - nor will they make any contribution. Their projects will continue to fail. Successful project managers will take on the roles of thought leader and action leader - the advocate and instrument of change - in the systems and practices that surround them. The alternative - the unmitigated machination of the rework cycle - will mean continuing the familiar pattern of development projects - unforeseen costs, unexpected delays, and unfulfilled promises.

Notes

1. Naval ship production: a claim settled and a framework built. Cooper, KG. *Interfaces*, a publication of the Institute of Management Sciences. December, 1980.
2. The 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, and so on) can be measured over time much as in traditional systems, and compared with the number of hours spent in the same timeframes, 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.
3. These project models were built using the dynamic continuous simulation language, DYNAMO. See *DYNAMO user's manual*. Pugh-Roberts Associates, and *Introduction to systems dynamics modelling with DYNAMO*, Richardson, George P and Pugh III, Alexander L. Pugh-Roberts Associates is a US subsiduary of PA Consulting Group
4. We acknowledge the bias in the data caused by the fact that easy, smoothly running projects rarely command the attention of consultants brought in by management. Therefore, the true range of "quality" values is likely to be broader "to the right" for each class if one includes less difficult projects.
5. Take heart, home builders. The construction projects in the sample reported here - power plants, combatant ships - all significantly exceed $1 billion.
6. Technical content and organisational practice in executing and reporting revisions vary widely. Account should be taken of the fact that the amount of effort on successive rounds of revisions will change (usually decrease). All measures of "quality" reported herein have been normalised to represent revision effort that is equal to original releases on a per unit basis.