

THE REWORK CYCLE: Benchmarks for the Project Manager

Kenneth G. Cooper, Pugh-Roberts Associates/PA Consulting Group, Cambridge, Massachusetts

Editor's Note: The following article is the third in a three-pan series, the first two of which are in the February 1993 issue of *PMNETwork*. There the concept and workings of the rework cycle are introduced. Based on dozens of applications to major development projects, the structure portrays flows of project work in which there are multiple cycles of rework. Rework typically represents the bulk of development project expenditures and time. The absence of its treatment in conventional methods and systems is a critical omission which consistently leads to project overruns. This article translates experience using the rework cycle "model" into practical suggestions and guidelines for use by managers of development projects.

When gauged by initial expectations for cost, schedule, and planned product, the prevailing modes of performance on complex development projects are (1) surprise and (2) failure. Conventional methods and systems have contributed to the consistently poor track record of complex project performance. Critical-path-based methods lack any consideration of the need for reworking incomplete tasks. Earned-value systems, even those endorsed or required by the government, have been likened by our contractor clients to driving a car by

watching the rearview mirror. We have developed and applied extensively a structure designed to portray and anticipate the rework cycle in development projects.

In the diagram below, the boxes represent pools of work (drawings . . . lines of code . . . feet of cable . . .).

At the start of a project or project stage, all work resides in the pool of **work to be done**. As the project begins and progresses, changing levels of staff (**people**) working at varying **productivity** determine the pace of **work being done**. But unlike all other program/project analysis tools and systems, the rework cycle portrays the real-world phenomenon that work is "executed" at varying, but usually less than perfect, **quality**. Potentially ranging from 0 to 1, the value of quality (as well as that of productivity) depends on many variable conditions in the project and company. The fractional value of quality determines the portion of the work being done that will enter the pool of **work really done**, which will never again need re-doing. The rest will subsequently need some rework, but for a (sometimes substantial) period of time the rework remains in a pool of what we term **undiscovered rework**—work that contains as-yet-undetected errors, and is

therefore perceived as being done. Errors are detected by "downstream" efforts or testing; this **rework discovery** may occur months or even years later, during which time dependent work has incorporated these errors, or technical derivations thereof. Once discovered, the **known rework** demands the application of resources, beyond those needed for executing the original work. Executed rework enters the flow of work being done, subject to similar productivity and quality variations. Even some of the re-worked items may then flow through the rework cycle one or more subsequent times.

For several years my colleagues and I have been using simulation models based on the rework cycle, and adapting them to accurately recreate, forecast, and diagnose the performance of each of a variety of projects. Across the projects, and within certain groups of projects, have emerged patterns of behavior which 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.

THE QUALITY RANGE

First, 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 the display box, the effective values range from as high as 0.90 all the way down to below 0.10. The chart shows the range of "quality" values for each type of project², and the resulting number of rework iterations.

Of the projects examined, commercial software development projects exhibited the

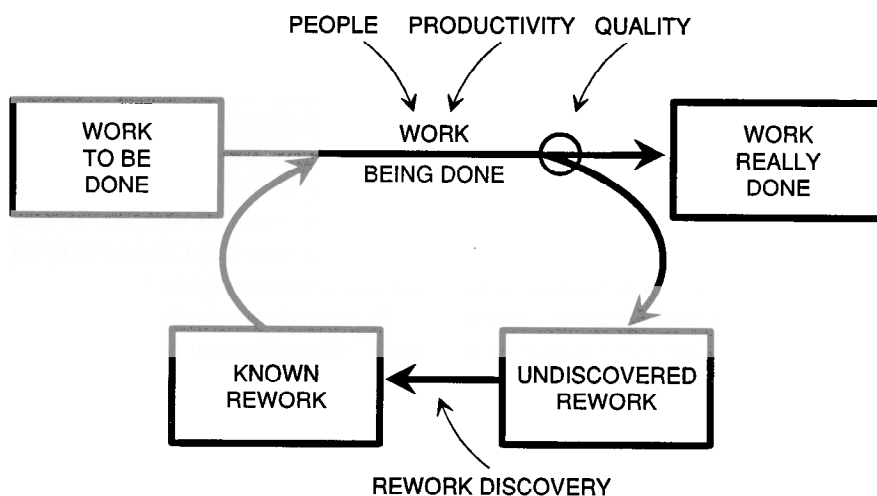
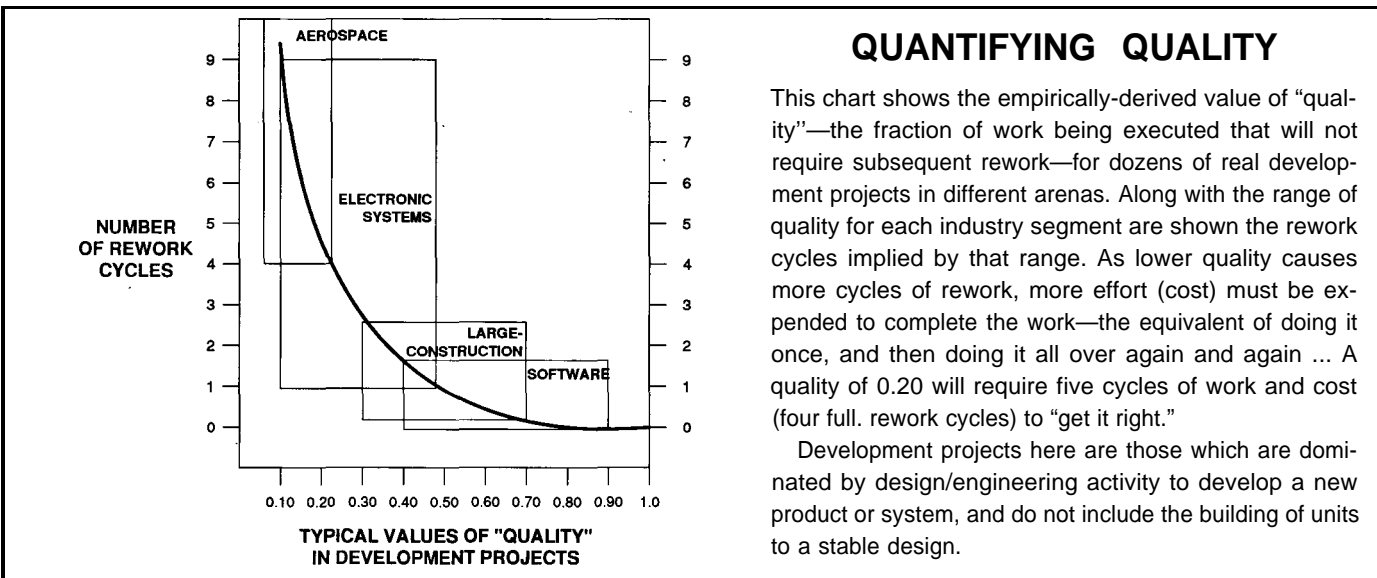


Figure 1. The Structure of the Rework Cycle.



lowest amount of rework (highest “quality”), ranging from little rework in some stages to “only” one and one-half full rework cycles in others. (Many such projects are actually adaptations of prior developments.) At the other extreme are aerospace development programs—all of these 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 one to nine full rework cycles, and the design of large-construction projects, with a range of about one-half to two and one-half rework cycles³.

Clearly the larger the technological leap being attempted by the overall project, the lower this measure of “quality” will be (the more rounds of rework). But 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 30-40 percent performance improvements are set forth as ambitious targets for organizations undergoing “change programs” and “process re-engineering,” the high y-leveraged impact of this kind of “quality improvement” must not be overlooked.

THE REWORK DISCOVERY TIME

Still, no matter what the quality improvement effort and impact, undetected errors and rework cycles are unavoidable in complex development projects. With 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 program-wide disruption, especially the development time impacts.

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 2 summarizes 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 2 would then be “number of years.”)

The chart shows four lines—for rework discovery times equal to one-quarter of the original design plan time (one-fourth year in our example), as well as discovery times of one-half, three-quarters, and one.

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 organization 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 isn’t 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.*

MEASURING WHERE YOU STAND

The prevailing rework discovery time on design development efforts is typically in the range of one-fourth to three-fourths 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 reissues of the work product in the stage of development being examined (historically, if available; otherwise, monitor an ongoing effort). Graph over time each subsequent round of revision as a line. Simplified, your chart should look comparable to that in Figure 3.

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.

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 “Revision B ‘s” on 500 “Revision A’s,” the prevailing quality during the work on Revision As was: $1 - (350/500) = 1 - 0.70 = 0.30$.⁴

ARE YOU LOST IN THE TRIANGLE?

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 the charts in Figures 4 and 5, 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, one 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 “percent complete,” versus (2) a retrospective computation of the percent **really** complete then (you should compute the percent really complete based on hours spent to that point, relative to the total hours eventually spent).

Perfectly accurate project progress monitoring would yield a straight 45° diagonal (hence the triangular ramp shape): at a perceived/reported condition of 20 percent complete, the actual percent complete would be 20 percent, 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 as shown in Figure 4.

Shown in Figure 5 for each of three typical combinations of quality and rework discovery time are the relations between perceived percent complete, as reported by traditional systems, and the real percent complete, when undiscovered rework is taken account of. In every case there is a gap between real

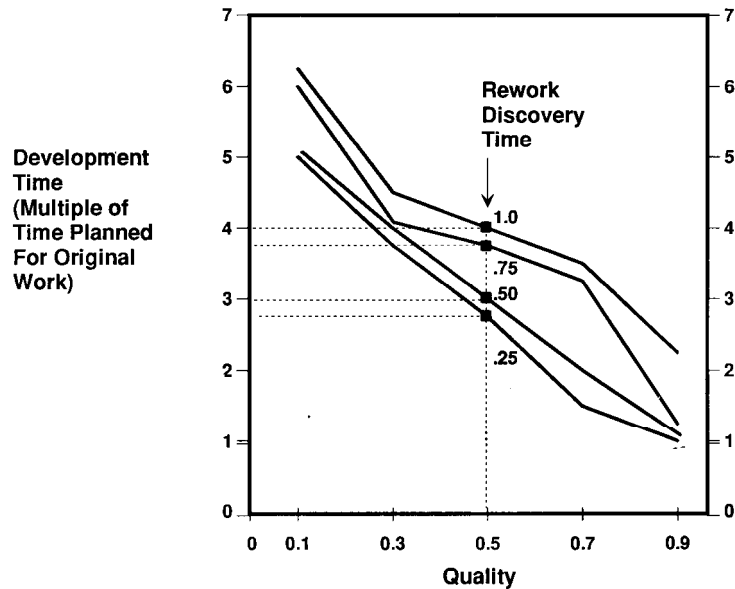


Figure 2. For most levels of work quality, improving the rework discovery time yields significant improvements in development schedules.

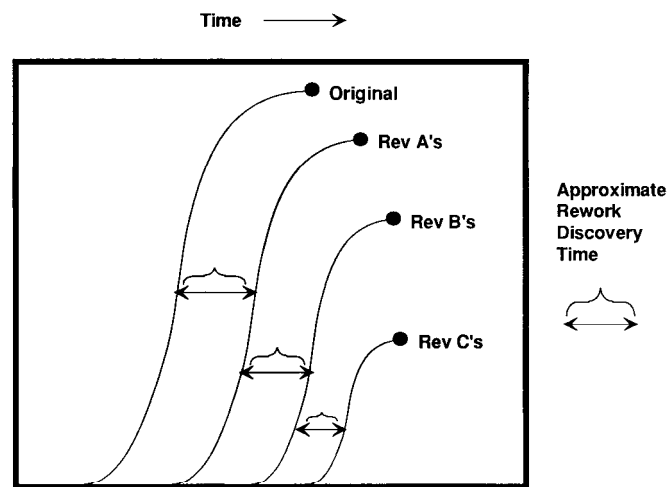


Figure 3. A critical addition to the set of closely monitored project performance measures should be the magnitude and timing of work revisions.

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; and
- the later the point of maximum uncertainty about real progress.

THE 90 PERCENT SYNDROME

The demonstrated uncertainty in real progress is responsible for several well-known troublesome project phenomena. One is the “90 Percent Syndrome,” in which for a prolonged time project managers report to executives or the customer that their effort is “90 percent” done. Examine the middle progress ramp in Figure 5 for an example of this. In these conditions, an earnest, responsible manager might well report 90 percent progress achieved, when 75 or 80 percent is really done. And so it goes until, after much distress and disappointment (and time and cost), 90 percent

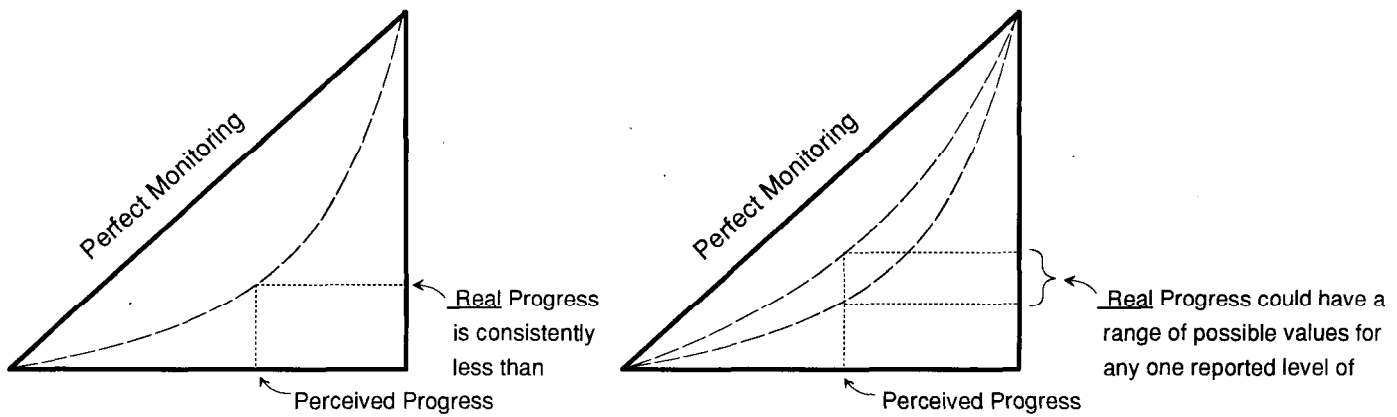


Figure 4. Progress Ramps help identify the true state of progress on a project or project phase.

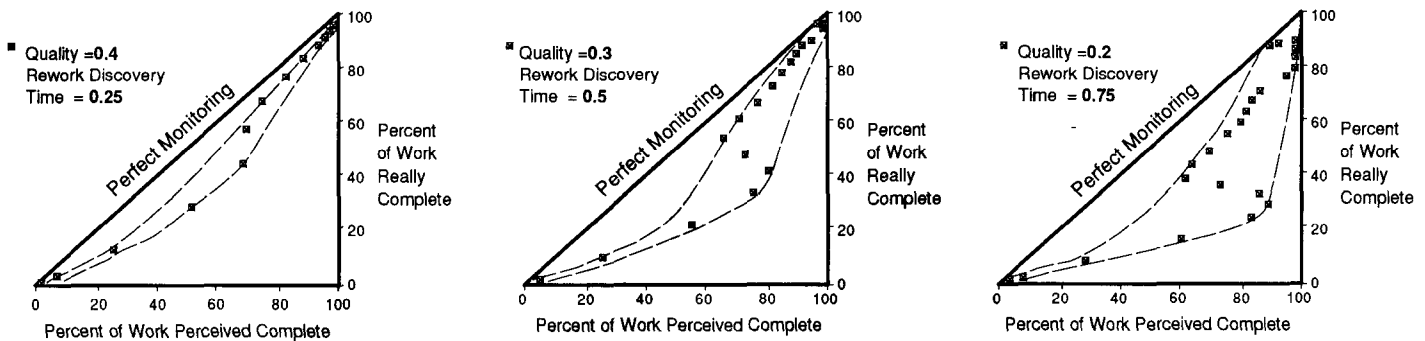


Figure 5. Lower “quality” and longer “rework discovery times” disguise low real progress, and increase the range of uncertainty in progress estimation.

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 percent syndrome.

THE LOST YEAR

One of our clients, managing a large new system development project described (not calmly) a related phenomenon, and requested a diagnosis. The 1000-person development effort had progressed to what seemed about two-thirds to three-quarters completion. **One year later**, after an additional thousand staff-years of effort, it seemed that they had made no progress, or had even lost ground, followed by a slow pace of progress toward 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 percent progress could be reported as “early” as at 30 percent real progress. Quite some time later (specifically, one year) having made 25-30 percent more real progress, “the system” can still be reporting 70 percent progress, or even less.

What happened? A lot of progress was made. A lot of rework was found and corrected (work that had been viewed as “done”). The “lost year” wasn’t lost; it obviously felt like “one step forward, 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 percent complete, was in fact just 60 percent complete, so what was seen as the remaining 30 percent felt like it took a long time to finish.

THE DELAYED PRODUCT INTRODUCTION

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 characterized by the third progress ramp (shown again in Figure 6). In this condition the true status of a development project remains highly uncertain until the very end. Combined with consistent overestimation 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 organization 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 testing earlier. While this increased testing costs, total project time and costs were reduced significantly. And although the resulting earlier rework discovery was a bit “scary” to the managers

at first, much more dependable introduction schedules were attained.

CONCLUSION

Despite the need for advances in the stagnated theory and practice of development project management, no new gimmicks, no new software, will significantly improve development projects' performance. Rather, we need to improve our fundamental understanding of how projects really work. A healthy start is understanding the critical role of the *rework cycle*. It must be recognized, monitored and minimized—and if it is, we can eventually achieve dramatic breakthroughs in project costs and schedules.

Documented herein are the range of quality and rework in different projects, how to monitor quality and the rework discovery time, and how to translate these newly-collected measures into more accurate project progress assessments. They are necessary, but only the most basic steps toward a more strategic and realistic view of projects.

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 internalize the associated lessons of the rework cycle. They must also learn the important differences between (a) the substantial *influence* the manager has over productivity, quality, and rework discovery (and hence, project costs and schedules) and (b) the relative lack of *control* the manager has over these same conditions. Understanding how the actions that managers can take influence their projects' outcomes requires extensions beyond the basic rework cycle structure. The very best of project managers know these intuitively. But just as important, managers' formulated action plans and their logic must be 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 behavior) won't 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

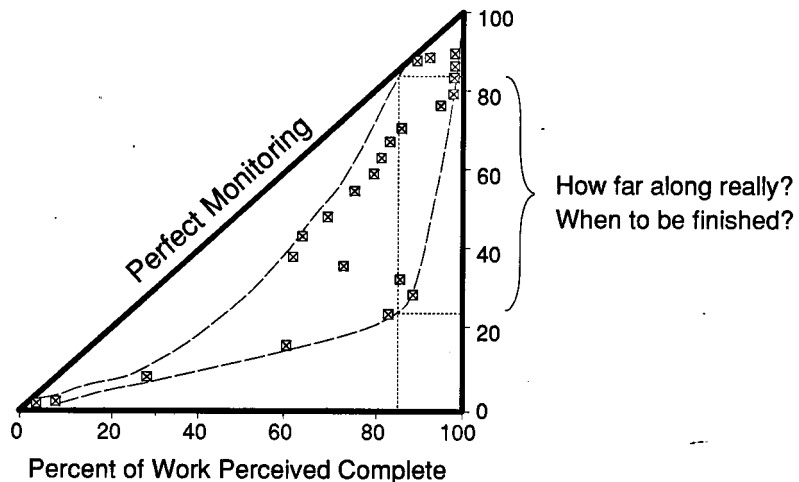


Figure 6. The end can appear to be near when real progress is as low as 30 percent, leading to overly-optimistic product introduction plans, destined to be delayed.

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.

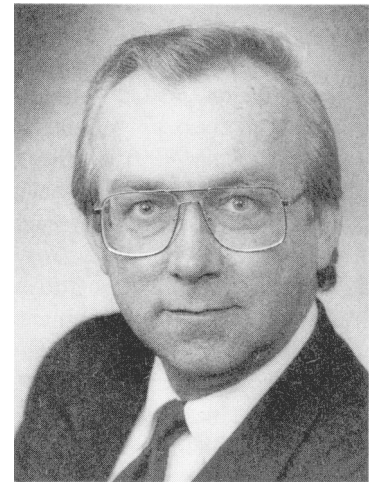
END NOTES

1. 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 Modeling with DYNAMO*. Richardson, George P. and Pugh III, Alexander L.

2. 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.

3. Take heart, home builders: the construction projects in the sample reported here—power plants, combatant ships—all significantly exceed \$1 billion.

4. Technical content and organizational 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 normalized to represent revision effort that is equal to original releases on a per unit basis.



Kenneth G. Cooper is director of the Management Simulation Group and senior vice president of Pugh-Roberts Associates, a division of PA Consulting Group. His management consulting career spans 20 years, specializing in the development and application of computer simulation models to a variety of strategic business issues. His clients include AT&T, Arizona Public Service, Hughes Aircraft, IBM, Litton, McDonnell-Douglas, Northrop, Rockwell, Transmanche-Link, and several law firms. Mr. Cooper has directed over 100 consulting engagements, among them analyses of 60 major commercial and defense development projects. He is an original author of the program management model introduced in this article. His group's offices are in Cambridge, Massachusetts, and Oxford, England. Mr. Cooper received his bachelor's and master's degrees from M.I.T and Boston University, respectively.

Continued from page 16

16. Lewis, Wayne M. and Jens, R. Micheal. 1987. Project Management Lessons from Past Decade of Mega-Projects. *Project Management Journal*, Volume XVII, No. 5 (December), pp.69-74.
17. Mason, Derek. 1982. Construction Control Schedule. 1982 *AACE Transactions*, pp. E.4.1-E.4.10.
18. Mueller, Frederick Wm. 1986. *Integrated Cost and Schedule Control for Construction Projects*, pp. 323. New York: Van Nostrand Reinhold.
19. Mueller, Frederick Wm. 1981. Simplified Integrated Comparative Methods of Cost and Schedule Control for Commercial and Industrial Building Construction. 1981 *AACE Transactions*, pp. A.O.1-A.O.10.
20. Popescu, Calin. 1977. CPM-Cost Control by Computer. *Journal of the Construction Division*, ASCE, Volume 103, No. CO4 (December), pp. 593-609.
21. Popescu, Calin. 1991. Quality Control and Quality Assurance in Planning and Scheduling. 1991 *PMI Proceedings*, pp. 57-61.
22. Powers, Jules. 1988. A Structured Approach to Schedule Development and Use. *Project Management Journal*, Volume XIX, No. 5 (November), pp. 39-48.
23. Rasdorf, William J. and Abudayyeh, Osama Y. 1991. Cost- and Schedule-Control Integration: Issues and Needs. *Journal of Construction Engineering and Management*, ASCE, Volume 117, No. 3 (September), pp. 486-501.
24. Raymond, Louis. 1987. Information Systems Design for Project Management: A Data Modeling Approach. *Project Management Journal*, Volume XVII, No. 4 (September), pp. 94-99.
25. Rounds, Jerald. 1985. Integrated Software Solutions to Cost Engineering Problems. 1985 *AACE Transactions*, pp. G.2.1-G.2.5.
26. Sancho, William M. 1982. A Critique of Integrated Cost/Schedule Systems. 1982 *AACE Transactions*, pp. A.3.1-A.3.4.
27. Sears, Glenn A. 1981. CPM/Cost: An Integrated Approach. *Journal of the Construction Division*, ASCE, Volume 107, No. CO2 (June), pp. 227-238.
28. Sheffield, Robert E. 1978. Construction Productivity and Progress Control. 1978 *AACE Transactions*, pp. 134-139.
29. Shiring, Paul B., Jr. 1982. Integration of Cost and Scheduling Systems. 1982 *AACE Transactions*, pp. H.3.1-H.3.5.
30. Thamhain, Hans J. 1987. The New Project Management Software and Its Impact on Management Style. *Project Management Journal*, Volume XVII, No. 3 (August), pp. 50-54.
31. Tiong, Robert L.K. 1990. Effective Controls for Large Scale Construction Projects. *Project Management Journal*, Volume XXI, No. 1 (March), pp. 32-42.
32. Tuman, John, Jr. 1988. Shaping Corporate Strategy with Information Technology. *Project Management Journal*, Volume XIX, No. 4 (September), pp. 35-42.
33. Ulrich, Walter E., Jr. 1977. Construction Labor Cost Measurement and Control. *Journal of the Construction Division*, ASCE, Volume 103, No. CO3 (September), pp. 329-341.



Chotchai Charoengnam is a Ph.D. candidate in the Construction Management program at the University of Texas at Austin. He earned a B.S. in civil engineering at King Mongkut's Institute of Technology, Bangkok, Thailand, in 1986 and an M.S. in civil engineering from the University of Kansas in 1989. He has one year of experience as a structural designer and two years as a field engineer. He is a member of the Project Management Institute, American Society of Civil Engineers and American Association of Cost Engineers.

Dr. Calin Popescu is Chotchai's sponsoring faculty member at the University of Texas at Austin.