Familiar Metric Management: The Power of the Trade off

Our esteemed contemporary, the *Harvard Business Review*, in its July-August 1996 issue permitted Henry Mintzberg, a professor of management at McGill University in Canada and during part of the year at INSEAD in France, to take on "management fads, management education, and the worship of gurus."

"Measurement is fine for figuring out when to flip a hamburger," Mintzberg mused. "But when used . . . to assess the worth of a complicated professional service, measurement often goes awry."

As you might surmise from his reference to hamburgers, the professor may have had tongue in cheek, possibly searching for that overcooked (or undercooked) meat. Cheek or not, his view is not uncommon. In fact, we hasten to agree. Many aspects of software development do escape measurement. People don't necessarily "think" on schedule; they may have headaches instead!

We also hasten to [italics] disagree. [end italics] To conduct software development as a business activity, the managers of that business need some assurance that the development can be conducted within bounds of cost, schedule, and reliability. They need assurance that the schedule is long enough to enable the work to be done at all. They would like to see it short enough to avoid running into the next millennium—a possibility becoming a real threat. And here, of course, is where metrics rears its head. Without something of the sort, you are aloft on a wing and a prayer!

Sequential estimation

Many estimating systems employ two equations. The first equation estimates effort on the basis of some measure of size, usually lines of source code, but sometimes function points or other indicators of size. The impact of the size term is often magnified a bit by raising it to a small power. This power ranges from 1.05 to 1.26 in different estimating systems, but essentially it is not far from unity. Further, these systems modify the effort value so obtained by multiplying it by 15 or 20 process attributes or effort multipliers. Where 1.0 is nominal, these effort multipliers each may range from perhaps 0.65 to 1.35. In equations of this type the length of schedule planned does not appear; it has no effect on the determination of effort.

These systems assign the task of determining development time entirely to a second equation. This equation finds the schedule, basically, by taking the cube root of the effort. To be more precise, various estimating methods obtain months of development time by raising person months of effort to powers ranging from 0.25 to 0.38.

The point of concern is that these estimating methods, however they may differ in detail, first estimate effort; then, independently, estimate schedule length. Yet it is a matter of common intuition among practical software managers that one affects the other. It is a point that Fred Brooks immortalized in his book, *The Mythical ManMonth*. It is a point that the QSM software equation embodies in multiplicative form. The multiplicative relation implies that the solutions for estimated values of Time and Effort occur in pairs. It implies that Time and Effort trade off. It follows then that management can exploit this one-equation tradeoff relationship.

The Effort-Time tradeoff

Rearranging the software equation (we derive this relationship by algebraic manipulation from the software equation, presented in an earlier column):

Effort/Development Time^4 = Constant where Constant = B (Size/Process Productivity) ^3

When a particular project is under consideration, its size, process productivity, and special skills factor (B) are known quantities, and hence this Constant possesses a fixed value.

At this point in project planning (before the project starts), Effort divided by the fourth power of Development Time, must equal this constant value. An estimator may lengthen or shorten Development Time, providing he reduces or increases Effort according to this equation. This is tradeoff. It depends on the multiplicative relation between Time and Effort in a single equation.

For those whose dislike of algebra began in high school and has never lessened, Figure 1 presents the story line in graphic terms. There is some short Time below which it is “impossible” to complete the system with appropriate quality at all. “Impossible” simply means that no one in our 4000-project database has reported doing so at the corresponding system size and process productivity. Similarly, it is “impractical” to stretch out development Time much beyond 130 percent of this minimum Time. Beyond this point, further gains in reduced Effort trail off and become impractical. Moreover, as the schedule lengthens, you lose the benefit of “fast cycle time.”

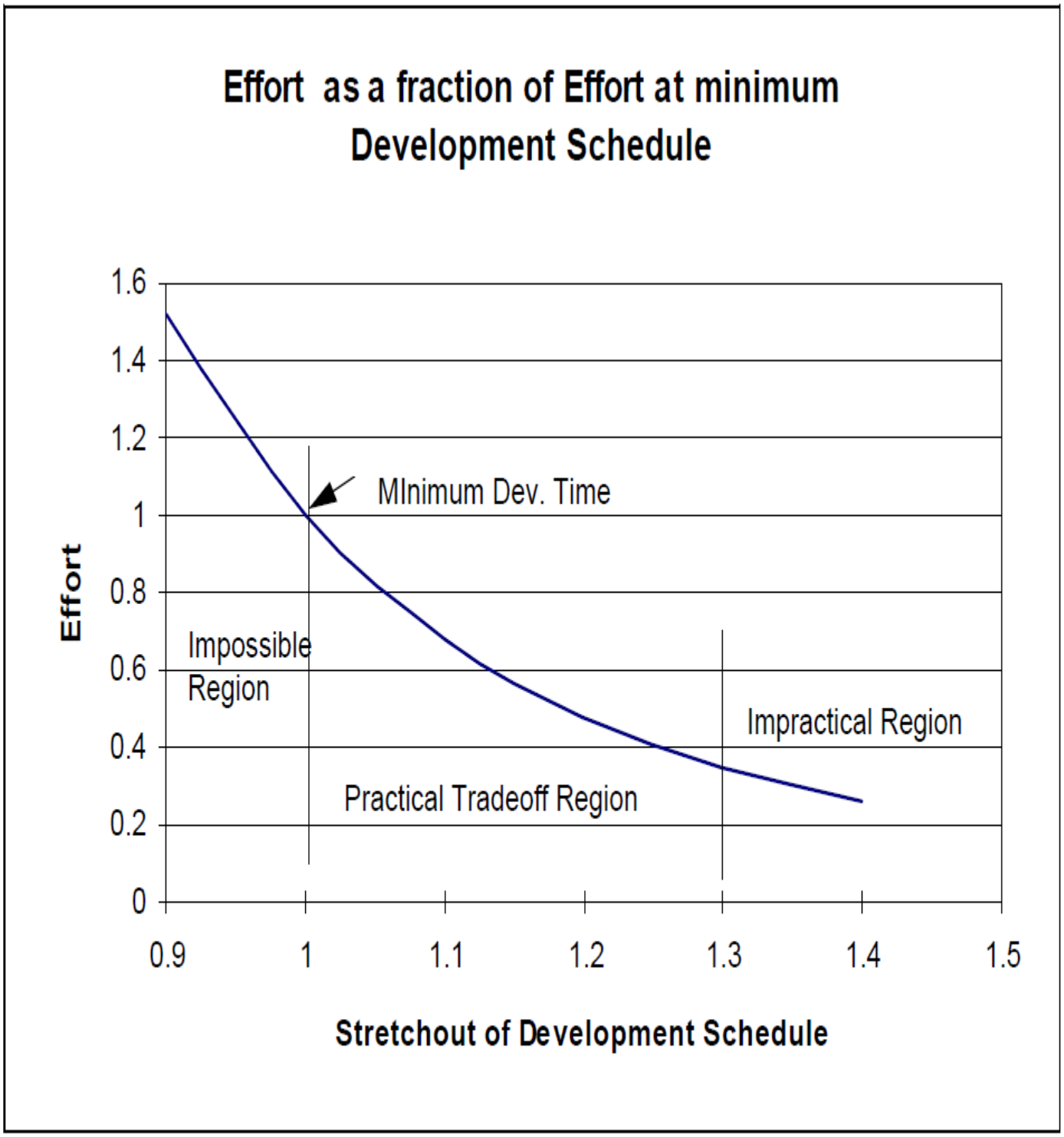

Figure 1. Between the impossibly short development time region on the left and the impractically long region on the right is a practical tradeoff region. Effort drops off according to the tradeoff law.

Not rocket science

That brings us to our next point. Albert Einstein found that mass and energy are related by his famous equation:

e = mc^2

That is a simple, precise equation. It would be nice if the software development relationship could be expressed this precisely. The reality is: it can't be. There are two sources of imprecision.

The first is that human beings are involved in the development process and we can't know for sure just what they are going to do the next time. (They do get headaches.) The developers of the effort-multiplier estimating systems try to accommodate this imprecision when they characterize it with 20 factors, each present in one of six levels. Of course, figuring out the effort multipliers is itself an imprecise and subjective art.

The second is that these human beings—developers, managers, and data collectors— provide project data that is itself fuzzy. On the one hand, our data comes from a wide variety of sources—an advantage in that it reflects practice worldwide. On the other hand, that breadth is a disadvantage in that every source does not define terms in the same way. For instance, many organizations seem to round off size to the nearest 5,000 source statements. They report development time in whole months. They seldom report effort in less than whole person months. Worse, there is a tendency to report development time at 12, 18, or 24 months.

In fact, we rejected one set of 104 systems for one of our studies of the tradeoff law because 28 of the development times had been reported as 12 months. Reporting more than a quarter of the systems at the same schedule indicates bias. This bias probably arose in the budget-planning process, where one year is a common planning horizon.

Well, when we have to work with imprecise data, we must expect imprecise equations to be the result. But some data is better than no data, and "four" appears to be approximately the correct ratio between Effort and Development Time. We can say this with assurance: having used this tradeoff ratio for nearly 20 years, we find that it gives very good results.

Working within limits

One observer reduced development time to 60 percent of his starting figure and found, according to the fourth-power law, that effort had increased by a factor of 7.7. He felt that was unrealistic. It is. This fellow was trying to operate in the Impossible region—planning a time shorter than the minimum possible time.

The laws of physics seem to operate throughout the universe, but the tradeoff law operates over a much narrower range, about 30 percent, as Figure 1 shows. Still, within that range you can obtain very substantial savings in effort and cost by a relatively small extension of planned development time.

Try it; see if it works. Many organizations have found that it does. They are saving a lot of money.