November 22, 1998

Strategies for Complex Projects

Lincoln Stoller, Ph.D., Braided Matrix, Inc.

Managing Projects

This article is about managing the software development process. Software management refers to the particular pattern you employ for controlling your software development projects.

Software management is itself a problem that requires some management. There are, or should be, management alternatives that you can evaluate when problems or new situations arise. In this article we discuss concepts of complexity, effort, risk, cost, and benefit as applied to project management.

Management is any strategy for using limited resources to solve complex problems. Comparing different strategies requires some measurement of how well each handles situations of varying complexity. Toward this end we'll consider the nature of complexity.

Square Law of Computation

The square law of computation is stated as:

"Unless some simplification can be made, the amount of computation to solve a set of equations increases as fast as the square of the number of equations."—Gerald Weinberg, Quality Software Management, volume 1, p. 130.

This law also applies to the creation of computer programs according to complex specifications. We can say roughly that, barring other simplifications, the complexity of a software system grows as square of the number of its parts. It follows that the ability to manage projects must also grow as the square of the parts. If your resources are not able to grow quadratically with the complexity of your projects, then your methods will soon reach a complexity limit beyond which they will fail. So for those of us who work with limited resources, which is just about everyone, management must be an important issue.

Consider a simple measure of aptitude given by the size of the biggest jigsaw puzzle you can assemble in 5 minutes. We measure a puzzle by the number of pieces it contains. The law of squares implies that the complexity of the problem, and hence the brain power required to complete a jigsaw puzzle in the allowed time, increases with the square of the number of pieces. It actually increases much faster, but that only makes our point more credible. This complexity is shown as a function of the size of the puzzle in Figure 1. Because we're considering a test of aptitude, the ability to prevail over a level of complexity is also a measure of brain power.

According to the law of squares, if you had one person whose brain power was twice that of another, then they would only be able to assemble a jigsaw puzzle with 40% more pieces. So if Butthead could handle a

40-piece puzzle in 5 minutes, Albert Einstein might be able to handle one with 54 pieces, assuming that Albert had twice the brain power.

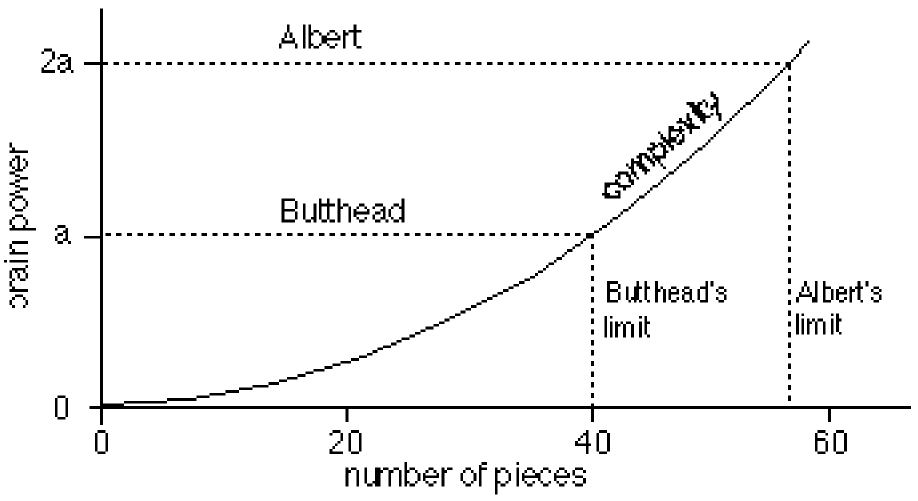

Figure #1

Figure 1: Brain power versus problem complexity: even having a lot of brains won’t get you very far.

The power of a software management method can be measured by the amount of effort required to implement it. This can be represented as a curve on an effort versus complexity graph.

Different methodologies can be represented by different effort/size curves. The curves give us a way to compare different methods and determine the better method. A bad method, for example, might be one that becomes difficult quickly. But even this might have its place for projects of a certain size, if you could be sure they would not grow over time.

Sticks and Stones

Let’s consider a simple example. The problem is how to get over cliffs of various sizes that you can not go around. We will consider two methods for solving the problem: the pile of stones method, and the ladder of sticks method.

The pile of stones method consists of building a big heap of stones on which to stand to get a boost over the cliff. The pile method is easy as long as you don’t need to build a large pile. Piling stones probably becomes impractical for cliffs over ten feet tall. The reason is that the quantity of stones needed to make a pile of any height increases in proportion to the square of the height.

The ladder of sticks method requires us to scavenge two long poles and a number of short rungs. We lash the rungs to the poles with rope, which we have plenty of, lean the ladder against the cliff and climb it. The ladder method takes a significant effort even for a short ladder. Lashing sticks to make a ladder takes some effort even if it only has one rung. However, doubling the number of rungs only requires doubling the effort. Consequently the effort to build a ladder increases linearly with its length. That works up to the length of the longest poles, which might be 20 feet. Building ladders longer than 20 feet would become so difficult as to become impractical. Figure 2 shows effort/height graphs for cliffs up to 20 feet tall; beyond this we’re stuck.

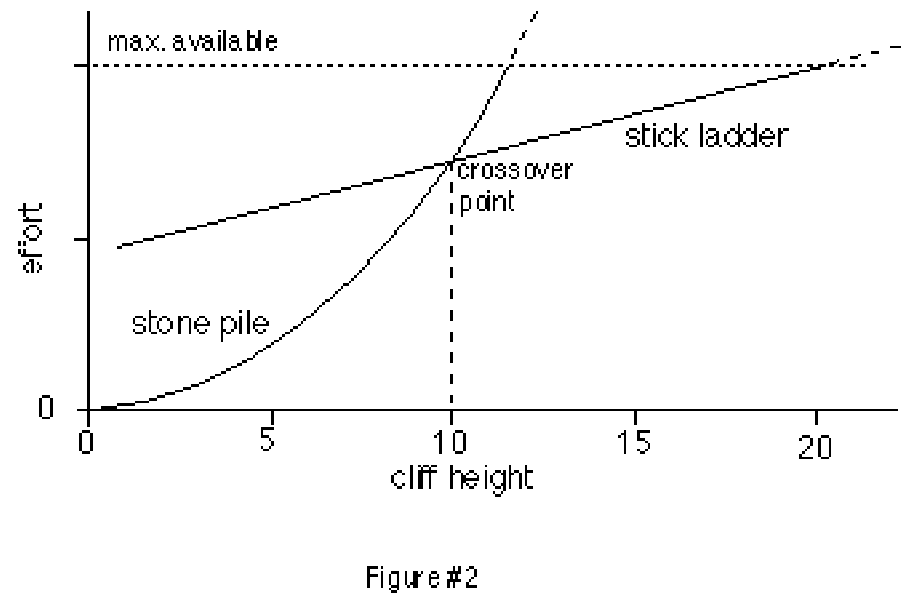

Figure 2: Effort versus height: comparing different ways to scale a cliff.

This graph illustrates that the pile method is easy for small piles but requires additional effort that grows quadratically with the height of the pile. The ladder method requires more effort even for short ladders, but the required effort grows more slowly: it grows linearly in fact. The graph shows that cliffs of about 9 feet or less are scaled most easily by standing on top of a pile of stones. Taller cliffs are more easily scaled using the ladder method.

This example shows that the important properties of methods can be discussed without describing the method’s details. The example is not meant to illustrate the law of squares, which explains why one of the lines in Figure 2 is straight.

Minimum of Effort

We can use the effort/complexity graph to compare different methods for managing software development. A project’s complexity can either mean the complexity of a project that changes over time, or the complexities of different projects that remain constant. We’ll begin by thinking of the effort required to manage different projects of differing complexities.

Software projects, unlike cliffs, become more complex as they get larger. The law of squares says that the effort required to manage a software project using any method will increase in proportion to the square of the project's complexity. Figure 3 shows the effort/complexity graphs for two methodologies labeled Method A and Method B.

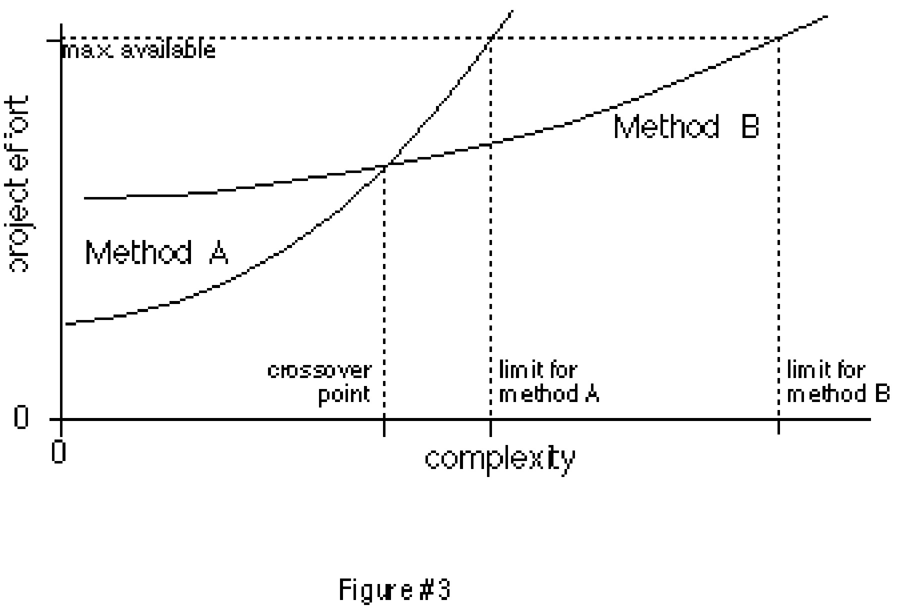

Figure 3: Project effort versus project complexity: different tools for different tasks.

Methods for managing software projects range from nonexistent (just start coding) to completely exhaustive (every step detailed from every angle). Using effort/complexity graphs allows us to compare methods abstractly, without discussing details. Comparing different alternatives from the same vantage point makes the underlying benefits and limitations more evident.

Some management methods

Some real-world software development strategies will help us apply these arguments to our own situations. Table 1 gives some candidates for Methods A and B as might be described by the curves in Figure 3.

Some sample methods: just for comparison

Method A	**Method B**
Prototyped design	Structured design methods
Single programmer/ consultant	Team of programmers and consultants
File Maker + Apple Events	4th Dimension
Build everything from scratch	Reuse preexisting code

Use whatever strategy worked last time	Develop new strategies to make use of new resources and achieve new objectives.

As in the cliff example, the maximum amount of effort available is set by the limits of our resources. Any project that requires more effort than can be applied to it is at serious risk of failure. Consider the alternatives of single programmer projects compared to team programmer projects. If you are only comfortable working with your own code, then you are using the “single programmer model.” This method may have worked for you in the past and you may have many reasons for believing this model will continue to work for you in the future.

Without knowing anything about your discipline or intelligence, the law of squares tells you that once you reach the limit in the size of the projects that you can handle with your current method, you will have little success in applying the method to more complex jobs. Even if you buy faster machines and more powerful tools, these resources will not be able to keep up with the increase in problem complexity. To handle more complex jobs you’ll need to find a different strategy.

Team efforts

We actually don’t know that the team approach is more effective at managing complex projects, so let’s consider some of its features. Not all groups act as teams, and not all projects lend themselves to teamwork. In order for a team approach to be effective, certain basic requirements must be satisfied.

Team members should be able to work on different jobs.

The project must factorize into separate tasks.

The difficulty of factoring the project should grow no faster than in proportion to the number of tasks it’s factored into.

If the project can be broken into parts to be assigned to different team members, and if each team member can complete their part independently, then the effort required to handle large projects could be controlled just by increasing the size of the team. This will work if the project tasks are sparsely related: that is if each task is related to only some other tasks and not all other tasks. Once each task becomes related to every other task the law of squares comes into effect. At some point the effort needed to manage the tasks will get out of hand and adding more team members will only make things worse.

The team approach works when carefully managed. It requires taking advantage of the structure of the project and planning ahead. For example, a project that requires a telecommunication link to a server from a remote site will factor into two tasks. One task is the establishment and support of data flow across the link. The other is the preparation of data to be passed through the user interface. One team member, the network expert, can write the code that carries generic information between the remote site to the server. The other team member can be put in charge of designing and coding the user interface. The only interaction required between the two is a set of conventions or specifications describing the tools the network expert will supply for use by the other team member.

Accounting systems are another case well suited to the team programming approach. Unless you’re a CPA you’re probably not an expert on accounting even though accounting is core to business operations. The reason that you don’t need to know accounting when working on other areas of business is because accounting has been designed largely to factor out of other business operations. This means that projects requiring accounting can usually be factored into two parts: accounting and business operations. Accounting is designed this way to support the team approach in business management. This is independent from any software development process. The law of squares has already been applied. This separation of tasks can be carried over to the programming of information systems.

Combination Methods

Just as no one tool is the best for all jobs, no one management strategy is the best for all projects. Having two tools at your disposal gives you the option of picking the best one for each project. A single programmer strategy for simple jobs, team management for complex ones, and a choice between the two in cases where either might work.

Figure 4 shows a combination of methodologies that enables the practitioner to handle a larger range of projects. Using this combination of methods, shown by the dark line, you would switch from Method A to Method B at the crossover point.

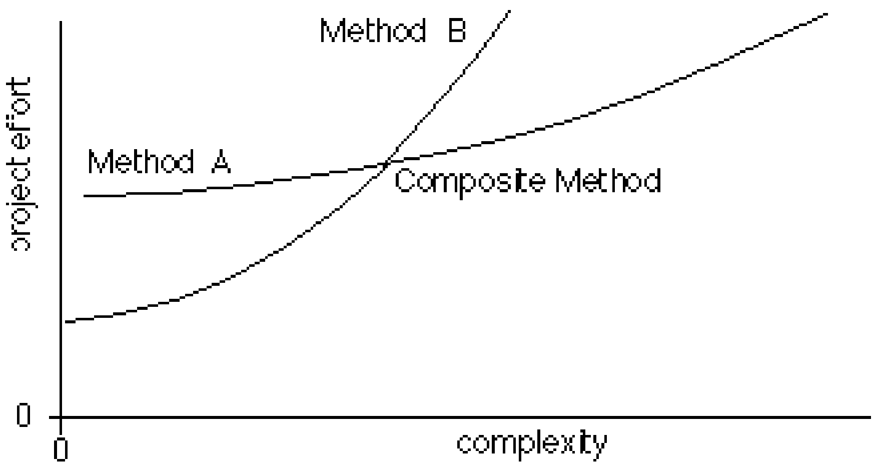

Figure #4

Figure 4: Combination methods: using what works best.

Not all changes in strategy require new methods. Some changes that open new opportunities can be managed using existing techniques. For example, 4D will soon run under both Windows and Mac operating systems. If you work according to the single programmer model you could learn the new tools required to support Windows platforms. Alternatively you could remain on the Mac side and develop new strategies for handling complex projects.

This is the choice between providing the same services to more customers, or more complex services in a field of fewer customers. Since 4th Dimension is expanding on both of these fronts the choice is yours. But remember the law of squares: if it takes an equal amount of effort to expand in either of these directions, then it will take four times the effort to expand in both at the same time.

Projects that keep on growing

Suppose you had an originally simple project on which you worked as a single programmer. Suppose that this project has grown, through the addition of new requirements, to an unmanageable size. You can no longer support the cascade of changes and new program requirements. You might be able to support the current effort if you could work 50% harder, but if the project continued to grow your progress would eventually stop.

Alternatively, if there are team members available, you could factor the project into separate tasks that you could delegate to team members. This would require more team management, but if communication is good and you've factored the project correctly, you might end up working 50% less. In this case a combined methodology could enable you to grow with an existing client whom you would otherwise have lost.

Alternatives from the start

A new client comes to you needing a fairly complex system. She'd like the system to become operational as soon as possible. She'd also like the system to be complete when it's first installed. It is then up to you to decide which of your two methods of project management is best for her. To make an informed decision you'll have to answer questions like:

- Is it more important to get the system running quickly, or is it more important to make sure it is complete? What are the costs and benefits in either case?
- It may be possible to cut a few corners in development, but will this approach cause problems later?
- The requirements may seem simple enough to begin coding now, without detailed specifications, but are the requirements really complete?
- Do you or the client have hidden expectations that may affect project management? Issues such as the system's growth potential, the client's management objectives, or possible changes in the end-user personnel may play a crucial role in your choice of an appropriate project management strategy.

If you understand and can communicate the benefits of different options, this may provide the client with valuable alternatives.

Minimum of Risk

Minimizing effort isn't the only basis on which to choose a design method. Another important factor is risk. For example, a method that can be applied with less effort may also have significantly less chance of success. Given an uncertain outcome the value of any method must be based on a combination of its cost and its risk. A method doomed to certain failure is not worth pursuing no matter how easy it is.

Every management approach has “corners” that can be cut to save effort in the short term. The trade-off is some increase in risk. Cutting corners is usually the first option considered when a project approaches the limits of manageability. Although some corners can always be cut, too much corner cutting may be a sign that the management method is inadequate.

Figure 5 shows the probability of success for two methods as a function of project complexity. Given a certain complexity the effort required to follow Method A may be acceptable, but the chance of success at this level may be unacceptable. Method A might be the most affordable for projects of a size marked X, but the likelihood that such projects fail half the time will make the method unappealing.

The risk associated with any method is strongly related to how successfully you’ve used it in the past. If you know how to succeed with a more difficult method because you’ve done something similar in the past, then the method’s outcome becomes more predictable. The more experienced you are with your tools, the less risky your methods become.

Remember that the client is also a source of risk. A full appraisal of any method will include some estimation of the client’s ability to manage her obligations. This is also strongly affected by whether or not the client has been successfully involved in previous software projects. As any developer who’s inherited another developer’s half-finished project knows, project failures are usually due to a combination of technical and managerial problems on both the developer and client side.

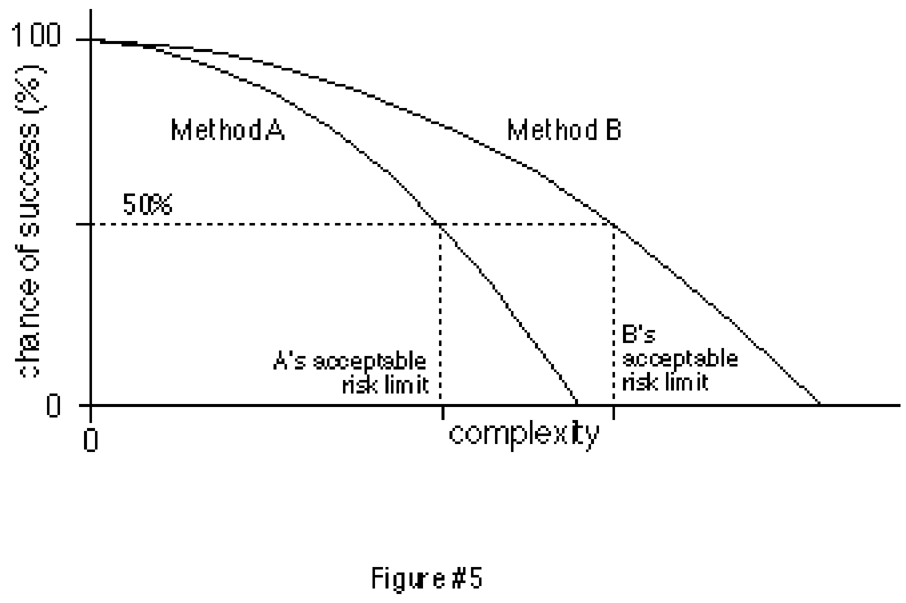

Figure 5: Probability of success versus project complexity: never underestimate the value of being safe.

Measure of Value

Economy of effort is a useful criterion only when all methods lead to products of equal benefit. A more general decision criterion is a measure of the project's final value. Value can be defined as benefit minus effort, or converted into monetary terms, profitability minus cost. To determine value we need estimates of profitability and cost.

The cost of building a system is proportional to the effort required to construct it. For simplicity we'll ignore the additional costs of installation, training and maintenance. You can add these back in yourself. We can create a cost/complexity graph by copying our previous effort/complexity graph with cost replacing effort on the vertical axis. In figure 6 the vertical scale has changed but the shape of the lines remains the same.

Now for the hard part: how do we measure benefit? A system's benefits are the rewards that come to its owners from it use. If a system enables a sales person to sell $220,000 worth of goods when they would only be able to sell $200,000 worth of goods without it, then the system yields $20,000 of benefits. If it enables a physician to serve 200 extra patients in a year without increasing office staff, then its benefit in dollars can be measured by the extra revenue generated by 200 patients. We'll use the terms "benefit" and "profitability" interchangeably because we're measuring them both in terms of money.

Risk also enters into the equation. It's certainly more valuable to construct a system that's 90% certain to satisfy the requirements than one designed to the same specs but that has only a 60% chance of success. However, adding additional risk variables makes the situation complex. We'll leave risk out of the equation by making the assumption, which may be inappropriate in some cases, that all methods are equally successful. In short, we'll assume you know what you're doing and you've succeeded at it before.

We need a sense of how profitability changes with complexity. We also need to know whether we're dealing with one system that might grow or shrink, or many systems both large and small. These two cases generate quite different results, as we'll now see.

Any given system usually has an optimal size determined by the function it serves. A more complex system would probably offer no additional benefit. A significantly less complex alternative might be useless. For example a multi-user system for a single user would be a waste of effort, while a 50-user system built on single-user technology would be useless.

Let's instead consider a range of possible systems, in keeping with the consideration of management strategies for different projects. Here the benefit of a system could be just about anything, so we'll have to be more specific. Let's consider an information system for some particular manufacturer. This system could consist of the following components:

- Clients
- Inventory
- Order tracking
- Manufacturing process
- Accounting

For simplicity let's say these processes are of equal complexity. In that case, adding any one of these compounds into the specifications to create a composite system would increase its complexity by an equal amount. A series of systems of increasing complexity could be drawn up to handle the following composite functions.

#	Project
1	Clients
2	Clients + Inventory
3	Clients + Inventory + Order tracking
4	Clients + Inventory + Order tracking + Manufacturing process
5	Clients + Inventory + Order tracking + Manufacturing process + Accounting

If we numbered these systems from one to five we'd expect some kind of increasing relationship between complexity and profitability. We represent the profitability of these five systems as five points on the graph in Figure 6. We will not connect the dots since drawing a line would imply that there are other alternatives whose value lies along the line, and that there is some kind of trend. Neither of these assertions needs to be true.

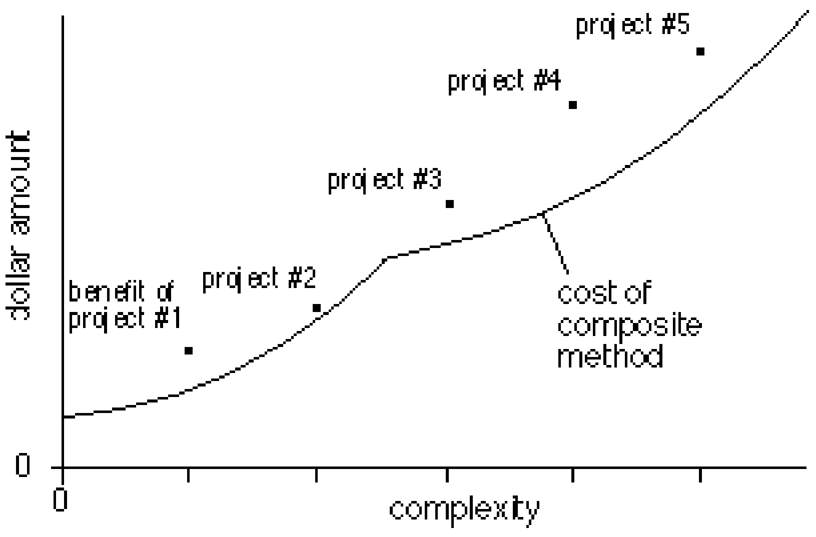

Figure #6

Figure 6: Cost and benefit versus project complexity: important things to know.

Our goal is to reach a description of value as a function of size. To reach this we need only to subtract costs from profitability. Do this by taking Figure 6 and measuring the cost and the profitability for systems of the various sizes. Measure the height of the profitability point above the X-axis and subtract from that the height of the cost curve at that point. Put a dot on the graph at a height given by this difference. Do this for the five systems and you'll get the dots shown in Figure 7.

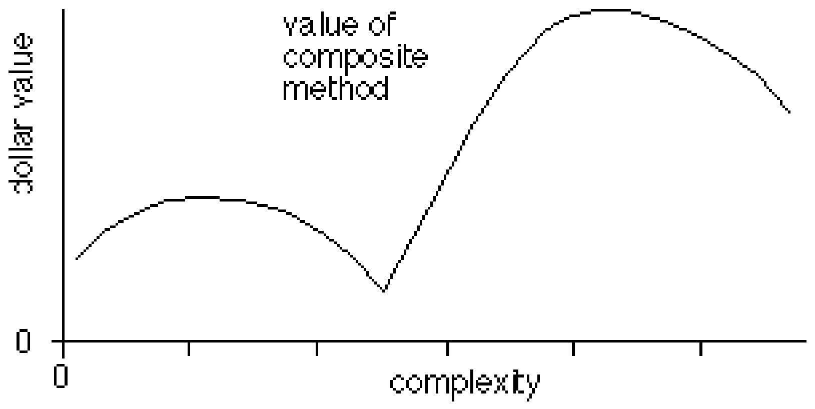

Figure #7

Figure 7: A general picture of value as a function of project complexity: the final measure of worth.

Figure 7 generalizes the information from the previous case. We've drawn a value/complexity line to better visualize the value/complexity concept, even though no real situation supports a continuous range of alternatives whose values are a smoothly varying function of complexity.

This is a simplified picture of an important relationship. A system's value is measured by its usefulness. It determines the client's satisfaction and the amount you get paid. While it's easier to talk about minimizing effort and risk, it's the maximizing of value that is the real objective.

However, this graph is only as good as the knowledge it's based on, and only as useful as the accuracy of these assumptions. The number and breadth of the assumptions that went into creating the value/complexity curves limit its utility. The lack of accurate knowledge explains why we usually work with the easier relationships of effort and risk. Still, value remains the ultimate criterion. Even when estimates are uncertain or wrong, that too can be an important consideration.

Relying on Alternatives

The graphs we've drawn show essential relationships between development methods and the complexity of the systems they're applied to. They tell us that those who rely on a single software development methodology shouldn't expect it to support them in any bid for more complex systems.

If your experience has been with small development projects, and you want to follow 4D's growth to provide solutions for larger and more complex systems, then you will have to learn new software management strategies. Identifying these alternatives is a topic for future articles.

Acknowledgments

This article is based on material discussed in chapters 9 and 10 of Gerald M. Weinberg's Quality Software Management, volume 1: System Thinking. (New York, Dorest House, 1992)