A priority criterion for serial computer system development projects

J. Mende
Department of Accounting, University of the Witwatersrand, Johannesburg

When a firm's systems department has insufficient resources to carry out all requested development projects immediately, managers serving on the Data Processing Steering Comittee need to decide which project should be completed first, second, third, etc. This can be done by ranking the projects in sequence of a 'priority criterion'. In the past net present value (NPV) was used for this purpose. However, in a small systems department, where projects are carried out in series, NPV is inappropriate. A more accurate measure of priority is a project's 'net worth' divided by its duration.
S. Afr. J. Bus. Mgmt. 1985, 16: 55 – 60

Indien 'n firma se rekenaarstelsel-afdeling nie oor voldoende hulpbronne beskik om aangevraagde ontwikkelingsprojekte onmiddellik aan te pak nie, moet lede van die Dataverwerkingsbestuurskomitee die projekte in volgorde van belangrikheid rangskik. Netto teenwoordige waarde is in die verlede vir dié doel gebruik. 'n Meer gepaste metode om aangevraagde projekte in 'n klein rekenaarstelselafdeling volgens belangrikheid te rangskik, is die gebruik van 'netto waarde' gedeel deur die duur van die projek.
S.-Afr. Tydskr. Bedryfsl. 1985, 16: 55 – 60

J. Mende
Department of Accounting, University of the Witwatersrand, P.O. Box 1176, Johannesburg, 2000 Republic of South Africa

Accepted February 1985

Project priority

Sequence decision

Owing to environmental pressures and internal managerial striving for improvement, business firms continually change — and their information needs vary accordingly. At the same time, advances in data processing technology constantly afford new opportunities to reduce data processing costs and to produce information that was previously infeasible or not economically viable. Both to accomodate changing information needs and to take advantage of technological advances, a firm's systems development department therefore continually receives requests to undertake systems development projects.

Usually these requests are many, but the department's resources are few. Therefore some projects have to be deferred in favour of others. Then managers serving on the Data Processing (DP) Steering Comittee face the decision: Which project should be tackled first, which one next, etc.

Priority criterion

Several publications (e.g. Duffy & Assad, 1980: 204) suggest that this decision be made by ranking projects according to their net present value or internal rate of return. However, the mathematical reasoning leading to these indicators is primarily concerned with *viability* — the decision whether a project should be carried out *at all* — and does not specifically address the question of *priority* — i.e. the optimal *sequence* of projects.

This article offers a train of reasoning specifically aimed at the sequence decision. The article begins with a resume of the net worth concept — a refinement of net present value presented in the September 1984 issue of this journal under the title 'A viability criterion for computer system development projects', (Mende, 1984). The article then extends the net worth concept to a series of deferred projects, and shows that net worth is not an accurate indicator of priority. Finally a new criterion is established which can be employed in a small systems department to rank projects in the most advantageous sequence.

Net worth of a project

Monthly contribution

A development project is usually a costly affair, involving the purchase of expensive software and/or the salaries of computer professionals engaged for a period of months or even years. In return for this investment the newly developed system furnishes a stream of future benefits which should be large enough to compensate for the development cost.

To define these benefits suppose an existing system was abolished. The firm's gross profit would then be affected in two ways. Firstly, it would diminish as users of the system's informational outputs find alternative, less effective ways of doing business. This represents the system's *'operational value'*, v. Secondly, gross profit would increase owing to the clerical salaries, computer rental, etc. saved by eliminating the system. This constitutes the system's *'operating cost'*, u. In combination, these two effects should normally cause a net decrease in gross profit,

$$c = v - u$$

defined as the system's annual, or monthly, *contribution*.

Conversely, when a new system is implemented, its operational value v is the increase in gross profit resulting from the use of its informational output; the operational cost u is the decrease in gross profit resulting from the salaries, rental, etc. necessary to operate the system. So again the contribution will be the difference,

$$c = v - u.$$

Trends in contribution

At first glance one might regard c as constant – an invariant characteristic of the system. But this is not true. At least four factors — obsolescence, the cost of capital, organizational growth, and learning — influence a system's contribution: The first two reduce it as time goes on, while the last two tend to increase it.

As a system ages, it becomes increasingly difficult to adapt it to changes in users' needs and to incorporate technological advances. Accordingly operational value drops and maintenance — a component of operational cost — increases. Therefore contributions can be expected to decline with obsolescence. Similarly the possibility of depositing contributions in a bank and allowing them to accumulate interest makes a distant future contribution less valuable than an equal contribution received in the near future. Therefore the interest factor causes a further shrinkage in contributions.

On the other hand, if the firm expands the system will serve more users, which ought to increase both the operational value and the operational cost. However, operational value v exceeds operational cost u, therefore the difference $c = v - u$ should expand. Accordingly one would expect organizational growth to increase a system's contribution. Similarly, as people gain more experience in using a system, organizational learning should enhance the system's operational value and thereby its contribution.

Rate of decline

To quantify these four influences, firstly let c represent the monthly contribution estimated at the time the system is implemented, and secondly suppose obsolescence causes it to shrink — while growth and learning increase it — by constant monthly factors a, g and e respectively. In the first month of the system's operation these three factors will then change c to

$$c(1 - a + g + e);$$

in the second month they will change it to

$$c(1 - a + g + e)^2,$$

and so on.

Further, the Theory of Finance (Weston & Brigham, 1978) shows that the possibility of earning interest devalues successive contributions to

$$c[(1 - a + g + e) \div (1 + k)],\ c[(1 - a + g + e) \div (1 + k)]^2, \ldots$$

where k is the firm's 'cost of capital' — approximately the bank rate less the inflation rate.

In this age of rapid technological advance the obsolescence rate a is relatively large. Therefore the rates a, g, and e largely cancel each other in the factor $(1 - a + g + e)$, and so $(1 - a + g + e) \div (1 + k)$ typically affects c as an overall divisor rather than a multiplier. Therefore contributions are expressed more realistically as

$$c \div (1 + r)^1,\ c \div (1 + r)^2, \ldots$$

in terms of the system's 'monthly rate of decline'

$$r = (1 + k) \div (1 - a + g + e) - 1.$$

Net worth

These contributions might seem to continue forever. However, data processing technology advances so rapidly and users' information needs change so frequently that every system will eventually have to be replaced by a more modern version. Then its contributions will cease. Over an anticipated life span of n years, a system's contributions therefore add up to

$$c \div (1 + r)^1 + c \div (1 + r)^2 + \ldots + c \div (1 + r)^n.$$

But this series merely reflects one facet of the system's financial impact on the firm. In order to gain these contributions, sacrifices are necessary. Firstly there is the development cost, D. Secondly, if the newly developed system replaces an existing clerical or computer-based system, then the firm forfeits all future contributions still potentially available from the existing system. To quantify this loss, let f represent its last monthly contribution before being replaced. Then the total forfeiture will be

$$f \div (1 + r)^1 + f \div (1 + r)^2 + \ldots + f \div (1 + r)^n.$$

To combine both the gains and sacrifices in one concept, a system's 'net worth' is therefore expressed as

$$N = (c - f) \div (1 + r)^1 + \ldots + (c - f) \div (1 + r)^n - D$$

or, summing the embedded geometric progression:

$$N = (c - f)\,[1 - (1 + r)^{-n}] \div r - D.$$

This formula serves as a measure of economic viability. When a project is first proposed, N can be evaluated from estimates of c, f, n, r, and D: A positive value of N indicates a worthwhile project and a negative value warns that sacrifices are likely to exceed gains. However, the next section will show that N is not actually a reliable indicator of project *priority*: A project with large net worth should not necessarily be carried out before a project with smaller net worth.

Delayed projects

Project queue

At present the more advanced data processing technology is still so complicated that many man-years of effort are needed to develop the typical computer-based system. However, a firm's systems development department usually has very limited manpower resources. Therefore only a few projects can be carried out at any one time.

Yet in the present age of rapid technological progress it often happens that there are more systems in need of development than the few which can actually be developed. So a *queue* of projects tends to build up, awaiting the release of data processing specialists from projects currently in progress.

To ensure that these pending projects will be tackled in the sequence most advantageous to the firm, management should periodically decide the relative priority of each project in the queue. To this end they require estimates of each project's cost and contribution.

Implementation delay

However when such estimates are made one cannot know in advance when a project will actually be completed. Therefore one estimates development cost assuming the project will be carried out immediately, and the contribution as if the new system will be implemented tomorrow.

These estimates then need algebraic adjustment to reflect the delay before DP specialists are assigned to the project, plus the time taken to complete it. Consider a system whose development cost, new and old contribution are estimated today as D, c and f respectively. Suppose it will actually be implemented t months from today. Then at that time growth will increase c to $c(1 + g)^t$, while the interest factor will decrease the contribution to

$$c(1 + g)^t \div (1 + k)^t.$$

In the same way f will change to

$$f(1 + g)^t \div (1 + k)^t$$

and D will become

$$D(1 + g)^t \div (1 + k)^t.$$

Thus the implementation delay t requires each current estimate to be adjusted by a multiplier $(1 + g)^t$ and a divisor $(1 + k)^t$. Anticipating that the divisor will normally exceed the multiplier, let the 'queueing factor' be defined as

$$q = (1 + k) \div (1 + g).$$

Then the implementation delay changes current estimates of D, c, and f to $D \div q^t$, $c \div q^t$, and $f \div q^t$.

Total financial advantage

These changes in turn modify a project's net worth. The previous formula

$$N = (c - f)\,[1 - (1 + r)^{-n}] \div r - D$$

now transforms into

$$(c \div q^t - f \div q^t)\,[1 - (1 + r)^{-n}] \div r - D \div q^t.$$

Factoring, this becomes

$$\{(c - f)\,[1 - (1 + r)^{-n}] \div r - D\} \div q^t$$

or simply

$$N \div q^t.$$

In general, every pending project in the queue awaiting attention by the systems development department will involve an implementation delay. Therefore if their respective net worths are estimated today as N_1, N_2, N_3 . . . then their net worths become

$$N_1 \div q^{**}t_1,\ N_2 \div q^{**}t_2,\ N_3 \div q^{**}t_3 \ldots$$

where ** represents the mathematical operation of exponentiation. Assuming that q is the same for all projects, their 'total financial advantage' will then be

$$A = N_1 \div q^{**}t_1 + N_2 \div q^{**}t_2 + N_3 \div q^{**}t_3 + \ldots$$

Clearly if $q = 1$ this series is independent of t_1, t_2, etc. But if q is significantly greater (or less) than 1 the total advantage depends on the projects' respective implementation delays, and therefore on the order in which successive projects are completed. Then to ensure that the firm obtains the greatest possible advantage one is faced with a complicated 'optimization problem' — namely to find the particular project sequence which yields a maximum value of A.

Net worth ranking

In the past one employed a simple algorithm for determining the optimal sequence: Namely to rank projects in descending sequence of 'net present value' — or net worth if obsolescence, etc. is taken into account. Thus the project with largest estimated net worth would be tackled first, the project with second-largest net worth would be carried out next, etc.

To illustrate, consider a firm with queueing factor 1,01 per month whose one-man system development department faces a backlog of two projects. Suppose project

– 1 has net worth 75 000 and requires four months' work
– 2 has net worth 50 000 and requires twelve months' work.

Then the algorithm will predict the optimal sequence correctly as project 1 followed by 2. For the total advantage,

$$A_{12} = 75\,000 \div 1{,}01^{**}4 + 50\,000 \div 1{,}01^{**}16 = 115\,000$$

is 7 % greater than the advantage obtainable if project 2 was completed before project 1:

$$A_{21} = 50\,000 \div 1{,}01^{**}12 + 75\,000 \div 1{,}01^{**}16 = 108\,000.$$

However, suppose project 1 requires twelve instead of four months' work and project 2 requires four rather than twelve months' work. Then

$$A_{12} = 75\,000 \div 1{,}01^{**}12 + 50\,000 \div 1{,}01^{**}16 = 109\,000$$
$$A_{21} = 50\,000 \div 1{,}01^{**}4 + 75\,000 \div 1{,}01^{**}16 = 112\,000$$

and the optimal sequence is project 2 followed by 1. However, if the projects were ranked in descending sequence of net worth then project 1 should precede project 2. Thus the algorithm gives the wrong sequence.

Optimization problem

How then, can one determine the most advantageous order of projects? In principle, this problem can be solved by listing all possible project sequences, applying the formula to each one and then selecting the sequence which yields the largest value of A. However in the typical systems department there are so many pending projects, and alternative ways of allocating computer specialists to them, that this approach poses a difficult computational problem.

Therefore one seeks some alternative to net worth; a priority criterion which allows the optimal sequence to be determined correctly by a simple process of ranking. As q is unlikely to be significantly less than 1, and no optimization problem exists if $q = 1$, the search will be confined to situations in which $q > 1$.

Priority criterion

Proportional duration

The desired alternative criterion is particulary simple to establish when one can assume that the duration of a project is proportional to the number of computer professionals assigned to it, and that all available personnel can work on it simultaneously. Then straightforward relationships can be formulated between t_1, t_2, t_3 . . . involving estimates w_1, w_2, w_3 . . . of the man-months' effort required by the respective projects.

Consider a small systems development department which exclusively employs 'analyst programmers', each one capable of participating in all phases of system development — analysis, design, programming, and implementation. These people can be assigned to projects in two alternative ways — either 'in series' or 'in parallel'. Suppose all of them — z in number — participate in each project, and complete projects one after the other in series $1-2-3$. Then

$t_2 = t_1 + w_2 \div z$

$t_3 = t_1 + w_2 \div z + w_3 \div z$

$A_{123} = N_1 \div q^{**}t_1 + N_2 \div q^{**} (t_1 + w_2 \div z) + N_3 \div q^{**} (t_1 + w_2 \div z + w_3 \div z) + \ldots$

Similarly if they were to complete projects in series $1-3-2$,

$A_{132} = N_1 \div q^{**}t_1 + N_3 \div q^{**} (t_1 + w_3 \div z) + N_2 \div q^{**} (t_1 + w_3 \div z + w_2 \div z) + \ldots$

Alternatively, after completing project 1 the department's analyst programmers might be divided into two teams, of size x and y respectively, and allowed to tackle projects 2 and 3 in parallel before resuming the series approach with project 4. Then

$z = x + y$

$t_2 = t_1 + w_2 \div x$

$t_3 = t_1 + w_3 \div y$

$A = N_1 \div q^{**}t_1 + N_2 \div q^{**} (t_1 + w_2 \div x) + N_3 \div q^{**} (t_1 + w_3 \div y) + \ldots$

Series superiority

As the two teams are to join forces when they tackle project 4, projects 2 and 3 should end at the same time. Therefore the two team sizes should be selected in such a way that $w_3 \div y = w_2 \div x$.

Consequently

$$\begin{aligned} w_3.x &= w_2.y \\ w_3.x + w_3.y &= w_2.y + w_3.y \\ w_3.(x + y) &= (w_2 + w_3).y \\ w_3 \div y &= (w_2 + w_3) \div z \end{aligned}$$

and therefore

$N_3 \div q^{**} (t_1 + w_3 \div y) = N_3 \div q^{**} (t_1 + w_2 \div z + w_3 \div z)$.

Furthermore as $x < z$,

$w_2 \div x > w_2 \div z$

and therefore, q being assumed greater than 1,

$N_2 \div q^{**} (t_1 + w_2 \div x) < N_2 \div q^{**} (t_1 + w_2 \div z)$.

Consequently

$N_1 \div q^{**}t_1 + N_2 \div q^{**} (t_1 + w_2 \div x) + N_3 \div q^{**} (t_1 + w_3 \div y) <$

$N_1 \div q^{**}t_1 + N_2 \div q^{**} (t_1 + w_2 \div z) + N_3 \div q^{**} (t_1 + w_2 \div z + w_3 \div z)$

which means that

$A < A_{123}$.

In the same way it can be shown that even if projects 2 and 3 do not end at the same time:

$A < A_{123}$ if $w_3 \div y > w_2 \div x$

$A < A_{132}$ if $w_3 \div y < w_2 \div x$.

Thus parallel execution of these projects results in a lower total advantage than series development.

The above argument can be extended to any pair of consecutive projects to show that the advantage diminishes if they are carried out in parallel. Therefore in general the series approach is superior, and so the optimal succession of projects is a series in which each project is completed before the next one starts.

Series criterion

Now to establish a priority criterion for series development, suppose the series $1-2-3$ were more advantageous than the series $1-3-2$. Then —

$N_1 \div q^{**}t_1 + N_2 \div q^{**} (t_1 + w_2 \div z) + N_3 \div q^{**} (t_1 + w_2 \div z + w_3 \div z) >$

$N_1 \div q^{**}t_1 + N_3 \div q^{**} (t_1 + w_3 \div z) + N_2 \div q^{**} (t_1 + w_3 \div z + w_2 \div z)$.

Subtracting corresponding terms and factorizing,

$N_2 [q^{**} (-t_1 - w_2 \div z) - q^{**} (-t_1 - w_3 \div z - w_2 \div z)] >$
$N_3 [q^{**} (-t_1 - w_3 \div z) - q^{**} (-t_1 - w_2 \div z - w_3 \div z)]$.

Multiplying both sides of the inequality by $q^{**} (t_1 + w_2 \div z + w_3 \div z)$:

$N_2 [q^{**} (w_3 \div z) - 1] > N_3 [q^{**} (w_2 \div z) - 1]$.

Therefore $A_{123} > A_{132}$ implies that

$N_2 \div [q^{**} (w_2 \div z) - 1] > N_3 \div [q^{**} (w_3 \div z) - 1]$.

So one would suspect that a project's priority might be reflected by the function

$P = N \div [q^{**} (w \div z) - 1]$.

To prove this in general, consider two successive projects, i and $i + 1$. Suppose the function is greater for project i than for project $i + 1$. Then

$N_i \div [q^{**} (w_i \div z) - 1] > N_{i+1} \div [q^{**}(w_{i+1} \div z) - 1]$

$N_i [q^{**} (w_{i+1} \div z) - 1] > N_{i+1} [q^{**} (w_i \div z) - 1]$

$N_i q^{**} (w_{i+1} \div z) + N_{i+1} > N_{i+1} q^{**} (w_i \div z) + N_i$.

Dividing both sides by $q^{**} (t_1 + w_2 \div z + \ldots + w_i \div z + w_{i+1} \div z)$:

$N_i \div q^{**} (t_1 + w_2 \div z + \ldots + w_i \div z) + N_{i+1} \div q^{**} (t_1 + w_2 \div z + \ldots + w_i \div z + w_{i+1} \div z) >$

$N_{i+1} \div q^{**} (t_1 + w_2 \div z + \ldots + w_{i+1} \div z) + N_i \div q^{**} (t_1 + w_2 \div z + \ldots + w_{i+1} \div z + w_i \div z)$.

Now adding the series

$N_1 \div q^{**}t_1 + N_2 \div q^{**} (t_1 + w_2 \div z) + \ldots N_{i-1} \div q^{**} (t_1 + w_2 \div z + \ldots + w_{i-1} \div z)$

$+ N_{i+2} \div q^{**} (t_1 + w_2 \div z + \ldots + w_{i+2} \div z) + \ldots$

to both sides of the above inequality, it emerges that the project series $1,2, \ldots i-1, i, i+1, \ldots$ is more advantageous than the series $1,2, \ldots i-1, i+1, i, \ldots$ So the most advantageous series is that in which no project is preceeded by one with a greater value of P. In other words, the optimal project series is in declining sequence of P.

Differentiated tasks

Thus $P = N \div [q^{**} (w \div z) - 1]$ serves as a priority criterion for projects to be completed by a systems department in which duration is proportional to manpower and all available personnel can be assigned to one project *at the same time.*

The first of these two premises applies in virtually any small systems department, but the second is only tenable in a department employing one or two people. However, even if small, the typical systems department normally *differentiates* the tasks of system development, employing people either as system analysts or as programmers. Consequently the programmers cannot participate in a project until the analysts have completed their preparatory work.

But this difficulty can be circumvented. The trick is to focus the argument on the programmers, assuming the analysts are appropriately scheduled so that programmers, having completed one project, can immediately proceed to the next. Letting w_1, w_2, w_3, etc. now represent the man-hours programming effort required by each project, and z the total number of programmers, then if projects are programmed in series $1-2-3$,

$$t_2 = t_1 + w_2 \div z$$
$$t_3 = t_1 + w_2 \div z + w_3 \div z.$$

Alternatively if the programmers were to complete projects 2 and 3 in parallel after project 1,

$$t_2 = t_1 + w_2 \div x$$
$$t_3 = t_1 + w_3 \div y.$$

Therefore the same algebraic manipulations as above could be performed to prove the superiority of the series approach and to show that the optimal series is in declining sequence of the priority criterion

$$P = N \div [q^{**} (w \div z) - 1]$$

where w represents the total programming effort.

Approximate criterion

Therefore the same function serves as a priority criterion in both kinds of department — the very small one which only employs analyst programmers and the small department which employs separate analysts and programmers. Only the meanings of z, w_1, w_2, w_3, etc. are different. However, even this discrepancy can be eliminated.

Firstly, returning to the definition of the queueing factor:

$$q = (1 + k) \div (1 + g)$$
$$= (1 + g + k - g) \div (1 + g)$$
$$= 1 + (k - g) \div (1 + g).$$

So $q = 1 + h$ where $h = (k - g) \div (1 + g)$ is very small, and the 'Binomial Theorem' (Ferrar, 1945: 133) is applicable. This predicts that (approximately)

$$q^{**} (w \div z) = (1 + h)^{**} (w \div z) = 1 + h.w \div z.$$

Now if $N_1 \div w_1 > N_2 \div w_2$, then

$$N_1 \div [(1 + h.w_1 \div z) - 1] > N_2 \div [(1 + h.w_2 \div z) - 1],$$
$$N_1 \div [q^{**} (w_1 \div z) - 1] > N_2 \div [q^{**} (w_2 \div z) - 1],$$
$$P_1 > P_2.$$

And in general if $N_i \div w_i > N_{i+1} \div w_{i+1}$ then $P_i > P_{i+1}$. So if projects are in declining sequence of $N \div w$ they will also be in declining sequence of P. Consequently the simpler function $N \div w$ serves as an approximate priority criterion.

Secondly, one may be able to assume that the proportion of programming effort, w', to total effort, w, is approximately the same for all projects. In that case

$N_1 \div w_1' > N_2 \div w_2'$ implies $N_1 \div w_1 > N_2 \div w_2$.

Therefore w can be interpreted quite generally as the total number of man months' effort required by the project.

Numerical proof

To verify the foregoing algebraic manipulations arithmetically, suppose a systems development department employing four programmers faces a queue of three pending projects:

— 1 with net worth 240 000 requiring 48 man-months effort
— 2 with net worth 210 000 requiring 60 man-months effort
— 3 with net worth 144 000 requiring 24 man-months effort.

Then the projects' respective priority criteria will be

5 000 , 3 500 , 6 000

and the optimal sequence should be

3 , 1 , 2.

To check, suppose $q = 1{,}01$ and each project involves two-thirds programming effort, so that they could be completed in eight, ten, and four months respectively. Then:

$$A_{123} = (240 \div 1{,}01^{8} + 210 \div 1{,}01^{18} + 144 \div 1{,}01^{22}).1000$$
$$= 512\ 889$$
$$A_{132} = (240 \div 1{,}01^{8} + 144 \div 1{,}01^{12} + 210 \div 1{,}01^{22}).1000$$
$$= 518\ 142$$
$$A_{213} = (210 \div 1{,}01^{10} + 240 \div 1{,}01^{18} + 144 \div 1{,}01^{22}).1000$$
$$= 506\ 443$$
$$A_{231} = (210 \div 1{,}01^{10} + 144 \div 1{,}01^{14} + 240 \div 1{,}01^{22}).1000$$
$$= 508\ 200$$
$$A_{312} = (144 \div 1{,}01^{4} + 240 \div 1{,}01^{12} + 210 \div 1{,}01^{22}).1000$$
$$= 520\ 082$$
$$A_{321} = (144 \div 1{,}01^{4} + 210 \div 1{,}01^{14} + 240 \div 1{,}01^{22}).1000$$
$$= 513\ 888.$$

Further, if all four programmers complete project 3 and then two programmers carry out project 2 while the other two work on project 1,

$$A = (144 \div 1{,}01^{4} + 210 \div 1{,}01^{24} + 240 \div 1{,}01^{20}).1000$$
$$= 500\ 461.$$

Therefore the project series $3-1-2$ indeed affords maximum advantage!

Application

Priority decision

So $N \div w$ indicates the relative priority of projects to be carried out by a small systems development department. Whenever the Data Processing Steering Committee receives a project request, it enables the data processing manager, the financial manager and other committee members to determine its appropriate position in the queue of pending projects.

To apply the criterion, they need current estimates of the following parameters for each project:

- a, the rate of obsolescence
- g, the rate of growth in contributions
- e, the rate of organizational learning
- k, the cost of capital
- n, the system's life expectancy
- c, the new system's expected contribution
- f, the old system's actual contribution
- D, the development cost
- w, the man-months' effort required.

Each project's priority can then be determined from these parameters by applying the formulae

$$r = (1 + k) \div (1 - a + g + e) - 1$$

$$N = (c - f)\,[1 - (1 + r)^{-n}] \div r - D$$

$$P = N \div w$$

Finally, the list of projects can be sorted in descending order of P to establish the optimal sequence.

Numerical example

Table 1 illustrates this procedure for the Systems Development Department of the previous example. The first nine rows of the table contain revised parameters for the three existing projects in columns 1 – 3, and the parameters of a newly requested project in column 4. The last three rows contain revised calculations of the priority criteria $N \div w$ for projects 1 – 3, and a new calculation for project 4. Consequently the optimal sequence of projects is 4 – 3 – 1 – 2.

Table 1 Project priority calculation

Project	1	2	3	4
a	0,002	0,004	0,002	0,002
g	0,001	0,001	0,001	0,001
e	0	0,001	0	0
k	0,005	0,005	0,005	0,005
n	120	72	120	120
c	4 200	5 400	3 000	6 000
f	500	0	700	1 000
D	80 000	100 000	50 000	70 000
w	48	60	24	36
r	0,006	0,007	0,006	0,006
N	236 000	205 000	146 000	357 000
$N \div w$	4 900	3 400	6 100	9 900

Limitations

This procedure affords a rational, quantitative approach to the project-sequencing decision in a small systems development department. It promises a more reliable decision outcome than mere 'gut feel', but one should beware of placing undue reliance on the results of the calculations. For they are subject to two sources of potentially major error.

Firstly, some or even all of the parameters a, g, c, etc. may be impossible to estimate precisely. So there is always a risk that the values submitted to the Steering Committee may contain gross inaccuracies. And these would correspondingly falsify the priority criteria.

Secondly, the formulae are based upon premises which at best only mirror reality approximately, and at worst distort it badly. For instance, rates of obsolescence, growth, etc. which were assumed to be constant are actually liable to vary from year to year, and if these variations are large, substantial errors may ensue. Also, the queueing factor, supposedly a constant, is unlikely to be exactly the same for every system at all times: Large differences would again result in appreciable discrepancies. And then if z is large, the Law of Diminishing Returns predicts a degree of disproportionality between project duration and manpower allocation which would cause further errors.

In view of these limitations, the numbers cannot be trusted blindly. Consequently one should not hesitate to override them if qualitative factors such as motivation, corporate image or competitive advantage indicate a different priority than predicted quantitatively.

Advantages

Yet despite its limitations, the above procedure is an improvement on the older method of ranking in declining sequence of net present value. Firstly the formula of net present value,

$$\text{NPV} = (c - f)\,[1 - (1 + k)^{-n}] \div k - D,$$

ignores the impact of obsolescence, growth, and learning. Secondly it can be shown that NPV only serves as a priority criterion when all projects are carried out in parallel — a premise which is quite atypical of systems development.

Therefore the new method using $N \div w$ is more likely to identify the optimal sequence of projects than the old net present value method.

References

Duffy, N. & Assad, M. 1980. *Information Management.* Cape Town: Oxford University Press, 243 p.

Ferrar, W.L. 1945. *Higher Algebra for Schools.* London: Oxford University Press, 210 p.

Mende, J. 1984. A Viability Criterion for Computer System Development Projects. *S. Afr. J. Bus. Mgmt.*, vol. 15, 144 – 149.

Weston, J. & Brigham, E. 1978. *Managerial Finance.* Hinsdale Illinois: Dryden Press, 1020 p. Chapter 9 and Chapter 11 Appendix F.