COMPUTER EVALUATION

This article examines computer evaluation from the point of view of someone wishing to create a computer system for some particular purpose, looking at the considerations and methodology required to ensure the chosen system is the most appropriate.

In order to consider the evaluation process it is first necessary to establish what it is that requires evaluation, that is, what is meant by a "computer." The definition of a computer or computing engine may be taken from the Oxford dictionary (1), which states that a computer is an "electronic apparatus for analysing or storing data, making calculations, or controlling operations." This definition implies that the computer include, in addition to the machine, the software, the power supply, and at least one human user/customer/maintainer. This fits well with other evaluations that we might carry out: evaluation of a car might well include measures such as top speed and gas consumption, but all the measurements we made would be made in the context of the use of the vehicle. So it is with computer evaluations. Different computer uses will of course necessitate different evaluations, but all evaluation will inevitably require the inclusion of elements other than the computer machine.

Similarly, a computer might be taken as a network of smaller machines (and the associated software) rather than a single machine. The Sun Corporation (2) has made this identification for many years—referring to the network as "the computer." This is particularly appropriate when making evaluative comparisons between mainframe and client–server solutions in a business environment.

In this article a "computer" is considered to be the whole of a computer system, that is, the hardware and software for one or more machines connected in a computer network. Furthermore, in its evaluation, consideration will be given not just to the computer but also the staff employed to develop and operate the computer.

This article describes the considerations and methodologies used in the evaluation of computer systems, and these are then illustrated in a number of examples and case studies taken from a company studied in depth by the authors—an engineering company with just over one thousand employees responsible for the design and development of mechanical systems. Its use of computers is not untypical, however, the principal use being for document production and electronic mail, a major concern for virtually all modern companies.

TYPES OF EVALUATION

Computer evaluation is required in two distinct situations:

Budgetary Evaluation. This is the application of the technique when an indication of total cost distribution is required for the life of the system, to ensure that the system is affordable. Whole-life costing can be used to choose between alternatives at every step of the life cycle to give the minimum cost of ownership for the remaining life of the system.

Design Choice Evaluation. This is used where there is a need to determine which of a number of competing system designs is the more effective over the life of the computer system. A *design* in this context may refer to the design of the electronics or software, but it can equally apply to a system built from off-the-shelf hardware loaded with standard software packages.

Conventional wisdom suggests that system design methodologies provide quality software and hardware solutions and that qualitative judgements would be concerned with the "rightness" of the delivered software and hardware in matching the customer requirements. However, it is all too easy to become concerned only with the technical excellence of the product, the costs of development and installation, or delivery to schedule. These issues inevitably press down on project managers, as they are often the immediate measure of success. Nevertheless, evaluation of a computer system must include an assessment of all the products used in the system for each of the phases of the life cycle.

THE ULTIMATE EVALUATION MEASURE—WHOLE-LIFE COST

The two reasons for evaluation, budgetary and design choice, map to two financial measuring methods, cost–benefit analysis and whole-life costing analysis. Whole-life cost analysis is the broader of the two methods, in that cost–benefit analysis tends to refer to the costing of a single option to examine its feasability which may not require consideration of all the costs and benefits of the full life cycle. The difference between the two is that whole-life costing is generally used to direct the design process on the grounds of cost, whereas cost–benefit analysis is used to make a single decision as to whether or not a particular option is affordable. If a cost–benefit analysis considers all the costs and benefits of every system design alternative, then it becomes the equivalent of whole life costing.

A draft international standard issued for comment in January 1995 by the British Standards Institution (3) provided a definition of whole-life cost as *the cumulative cost of a product over its life cycle.* Significantly for the computer industry, although software is not explicitly covered by the draft standard, the panel for the standard does include representation from the United Kingdom's National Computing Centre Limited. Software systems should and can be dealt with in the same way as physical systems from a whole-life cost point of view.

Alternative descriptions of whole-life cost are available, for example, from Dhillon (4) as *the sum of all costs incurred during the lifetime of an item.*

Major users of whole-life cost techniques to date have been the civil engineering fraternity and the defense industry. The industries that apply the techniques are those that face major competition to sales, or budgetary constraints to purchases. UK defense contracts are now including whole-life cost requirements at the bid stage. Other industries employing whole-life cost arguments to gain business include printer (Kyocera) and car (Daewoo) manufacturers. From a purchaser's perspective the objectives are usually simple: guaranteed affordability, coupled with a design that demonstrably has the least whole-life cost for a specified design life. Demanding whole-life cost at the bid or design phases forces potential contractors to think in terms of the most cost-effective set of design solutions. Once adopted by an industry, whole-life cost becomes an integral part of the design decision process.

THE APPLICATION OF WHOLE-LIFE COSTING TO COMPUTER EVALUATION

Although the technique may be common practice in other industries, its application in the Information Technology (IT) industry is less well established. Computer hardware and software engineers need to be concerned with the benefits of the whole-life cost approach in application to the computer system design process.

Best *software* practice currently concerns itself with the application of an accepted structured design method. In the recent past there appears to Sommerville (5) not to have been a definitive way of establishing what is "good design." Conventional wisdom might suggest that quality judgements would be concerned with the quality of the delivered code or algorithms, in matching the customer requirements. These qualitative judgments can be influenced by:

- Cost of the delivered software
- Timeliness of the delivery
- Uncorrected bugs
- Functionality
- Ease of use

Similarly, qualitative judgements on the suitability of the hardware of a computer-based system might be based on:

- Cost of the delivered hardware
- Timeliness of the delivery
- Ease of installation
- Ease of maintenance or the cost of a maintenance contract
- A number of technical specification details, such as Super VGA (SVGA), fast CD drive, or large/fast disk drive.

To apply the technique of whole-life costing, each of the above are directly quantified in monetary terms in a whole life-cycle cost profile. This profile, by definition, incorporates all possible costs, which together contribute to a single cash figure for comparison purposes.

It is important that *all* the costs of ownership of the system are included, not merely the costs associated with the software development life cycle or hardware purchase. Existing cost estimating methods for software, such as COCOMO (6) or Putnam (7), are largely concerned with development costs. Other significant costs are not dealt with. These might typically include the cost of:

- Hardware
- Hardware maintenance
- Software operation
- Network installation and support
- Software license
- Database administration
- Data validation
- Training

In application, the costs included in the life cycle cost can be categorized as:

- Analysis and design
- Production and construction
- Operation and support
- Retirement and disposal

The sum of these costs for a product can then be used for the two ends of establishing afford ability and comparing system alternatives.

TIMING OF THE APPLICATION OF WHOLE LIFE COSTING

The timing of the use of the whole-life costing method of evaluation is important. Fabrycky and Blanchard (8) have shown that the principal decisions, with the greatest influence on cost of ownership, are taken right at the beginning of the design process. By the end of the concept design process typically two-thirds of the life cycle cost has been committed. To have any financial effect on the system, life cycle cost techniques must be in place that assist the design decision-making process.

THE PROCESS OF EVALUATION

The process shown in Fig. 1, taken from Bradley (9), provides a model of the process of system design to whole-life cost for an organization using formal design reviews. Although not specifically designed for software or computer systems, the model is found to be equally applicable to both hardware and software systems. Here feasible alternatives are identified, typically using brainstorming techniques, and these are then costed individually to provide cost profiles for each of the competing design options. A subsequent design review meeting decides which system design option is to be taken, taking a range of qualitative data into account as well as the whole-life cost comparisons. The process is iterative, reflecting both the need for the design review's right to veto and the need to design to whole-life cost in a top-down manner, in greater detail at each successive level in the system design hierarchy.

The complete process has nine steps, centered on the *design review meeting*. The process, in summary, is shown in Fig. 1, with amplification of the individual steps as follows:

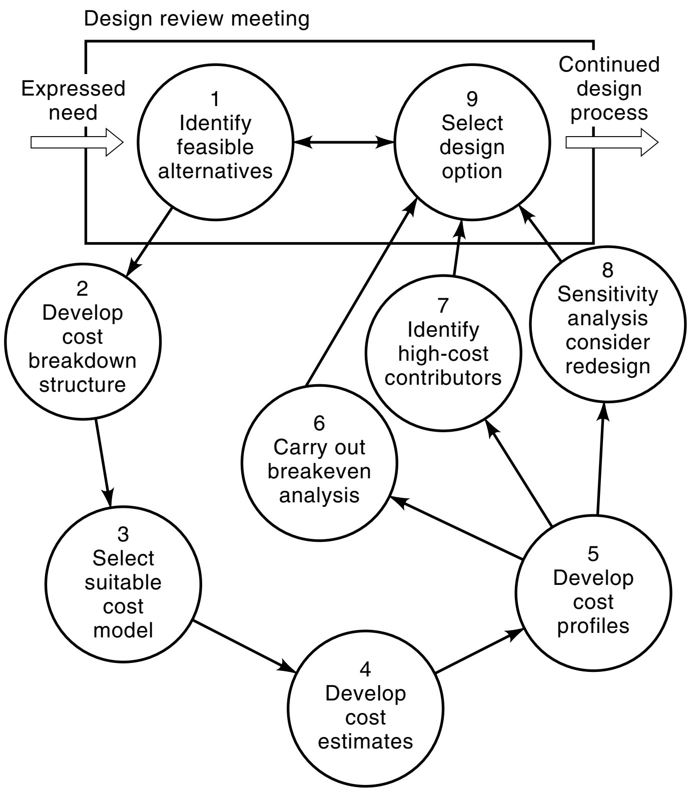
Design review meeting
Expressed need
1 Identify feasible alternatives
9 Select design option
Continued design process
2 Develop cost breakdown structure
3 Select suitable cost model
4 Develop cost estimates
5 Develop cost profiles
6 Carry out breakeven analysis
7 Identify high-cost contributors
8 Sensitivity analysis consider redesign

Figure 1. Design to minimum cost.

1. *Initial Design Review to Establish the Feasible Alternatives for a Design Solution.* This initial step provides the analyst with guidance on the costing task, alternatives to be costed, the overall life span, financial discounting, the need for an accurate estimate or comparative study, and the boundaries of the costs to be included.
2. *Development of a Suitable Cost Breakdown Structure.* All elements of the project need to be cost-accounted for. The cost breakdown structure used must also be one that will be supported by the subsequent system management tasks. This enables the estimate to be compared with the actual costs incurred during the system life.
3. *Selection of a Suitable Cost Model for the Analysis.* For each element of the cost breakdown structure created in step 2, costs for acquisition, operation, and disposal need to be separately included. An audit trail is an essential feature of any model used, to allow identification of the source of any cost data used. The model must cater for the time span of the product life, allowing cost data entry for each year of life for the cost elements included.
4. *Development of Costs Estimates.* Cost estimates are attached to the cost elements in the model. These can be based on actual data from historical records, quotations, or estimates obtained from the many estimating techniques available (e.g. parametric, comparative, detailed). This article does not provide guidance on cost-estimating algorithms and methods. Bradley and Dawson (10) provide algorithms for estimating the development timescales of INGRES applications.
5. *Development of Cost Profiles.* Cost profiles are created by the summation of all costs for each year. These are generally presented as a histogram. The cost profile may be adjusted to take account of the time value of money using the following formula:

$$P = F\,\frac{1}{(1+I)^n} \tag{1}$$

where

F = future value
P = present value
I = discount rate
n = numbers of years

If present values are used to generate a cost profile, then the whole-life cost practitioner must take care to use a discount rate and a range of years appropriate for the computer system being studied. For example, in relational database applications a typical value for n would be between 3 and 10 years. Discount rates (I) vary with the industry, but are generally greater than 5%. Rates are often fixed centrally in an organization and may even be fixed by government agencies. Different combinations of I and n can lead to radically different system design decisions; indeed, a high discount rate can lead to design choices where the early project costs are dominant, leading to delivered projects that will have high actual costs in later life. It can therefore be helpful to have profiles of both present value and undiscounted costs and benefits.

Profiles of discounted values are normally expressed as a single value, the widely used accountancy/spreadsheet term *net present value* (NPV), which is the sum of the present values of annual costs for the lives of the items.

6. *Breakeven Analysis (Optional).* A combination of cost profiles and net present value will normally provide a sound basis for comparison. In some cases, however, where competing designs have very similar cost profiles, it can be useful to compare the accumulating costs rather than the year-on-year costs in the profile. As the name implies, the break-even analysis reveals the year(s) in which the competing designs have identical whole-life costs, i.e., where the accumulating cost curves for competing designs cross. This identifies the point where a design option with higher initial costs will start to show an overall cost saving.
7. *Identification of High-Cost Contributors.* A review of all cost elements in the structure will identify
   a. The high-cost drivers
   b. The high-cost periods in the whole life cycle
   Once identified, these can be candidates for redesign with a view to whole-life cost reduction, causing an iteration.
8. *Sensitivity and Uncertainty Analysis.* Sensitivity analysis, or the determination of the effect of changing estimates, can be useful in identifying risks to the project. Software is available to carry out sensitivity analysis automatically in a spreadsheet. Similarly, uncertainty can be treated to provide decision makers with guidance on the range of whole-life cost for each system design option.
9. *Selection of the Design Option.* Ultimately, decision makers have to balance all the information being provided from the design review. Generally all the influences on a system design can be converted to a cost and incorporated into the whole-life cost evaluation. However, there are occasions where there are inputs to the decision that defy a cost analysis. This usually happens where such inputs are outside the boundary of the evaluation. Invariably, in such situations, the boundary of the evaluation needs to be reassessed and the method repeated.

THE DIFFICULTIES OF APPLYING THE WHOLE-LIFE COST EVALUATION

It is necessary to understand the costs that will be generated for each of the following:

- Category of cost
- Component in the component breakdown for the computer system
- Time period in the design life

The very size of this task can be a problem, as it represents a significantly large three-dimensional array of data for even a small system. The need for data for the task can be insurmountable, particularly for the operational period of a computer system. How, for example, is it possible to quantify the cost of a bad impression on a customer caused by the poor appearance of inferior printer output? Historically, organizations have collected data on initial hardware purchase and software development and neglected data on the operation of the system. Fortunately, in comparative evaluations it is only necessary to include the costs where there is a difference between competing options. In such situations the output of the evaluation is therefore the cost difference between options rather than an absolute cost.

BENCHMARKING VERSES THE COLLECTION OF LOCAL IN-SERVICE DATA

No evaluation is possible without data. Generally, the collection of data is driven by the need for greater understanding and control of processes and products. This is true of the whole-life cost method of evaluation. Local data can come from two sources: firstly from benchmarking experiments to measure performance of some aspect of the software or hardware, and secondly from measurements of systems currently in service.

Benchmarking of computer hardware is used mainly to check the manufacturer's own published figures in the context where the hardware will operate. It may be useful for a company to check, for example, how many pages per minute can be printed on different printers on a sample of the company's usual output. Software may also need to be benchmarked, particularly when large volumes of data are to be used. A database query, for example, may give a near-instant response while it is being developed using low volumes of sample test data, but when the system is loaded with the volumes of data it must handle in service, the delay caused may lead to a significant overhead in wasted operator time.

When a system is put together involving hardware and software for individual machines and across a network, it becomes much more difficult to perform meaningful benchmark experiments, as it can be difficult to simulate the exact conditions under which the system will be used. In such conditions the examination of existing systems being used within a company's own environment can give more accurate and reliable data. This is particularly important when determining reliability data. To be able to predict the rate of hardware failure or software error by benchmarking experiments would be prohibitively expensive, as the tests would need to be run on a significant number of machines for an extended period. By the time the test was complete, it could easily be out of date, as the manufacturers would probably have released an upgraded version of one or another of the components of the system.

System designers need to be able to collect information on the performance of existing system configurations as a basis for providing new and improved system designs. This has resulted in the implementation of computer-based systems for the collection of local in-service data. Traditional, step-by-step implementation of systems both supports and is supported by this data collection activity. Very small changes in the system configuration are seen to be desirable and are enabled by the act of observing the performance of the current system design. The discrete changes that result generally improve the system design with a minimum of risk. The desired results are cost reductions, including the benefits derived from improved availability, typically from changed maintenance regimes or small configuration changes. In this environment gradual improvement, the Japanese *kaizen,* can be seen to be desirable. Such small changes maximize the relevance of collected data, as any deterioration of a particular product can be detected by continuous data collection and monitoring.

Problems occur in computer evaluation when a step change is promoted using new technology. Here, even if an organization collects data, the newness of the incoming technology means that no collected data are available and cost data must then come from external organizations. This in turn can be problematic, as no two organizations are identical. Adjustment of cost profiles may be necessary to match the expected differences between the organization providing the data and the one carrying out an evaluation.

THE DISTRIBUTION OF COSTS OVER THE SYSTEM LIFE

Research carried out by Bradley and Dawson (10) has demonstrated that the highest costs in a computer system are likely to be in the system use, software design, and hardware maintenance/replacement. Figure 2 demonstrates the Pareto-like nature of the costs of a particular system. Much has been written on the software design process, and so this facet of evaluation is ignored here. Instead the following sections concentrate on reducing the costs associated the use of computers.

DATA AND COSTS CONCERNED WITH COMPUTER SYSTEM USE

Human–computer interaction is critical to productivity and hence to the cost of the system that is being used. End users may be less productive at their desktop than they could be because of poor application skills or poor IT performance. Managing the cost of this lost productivity to the business is the most important way of reducing the whole-life cost. Understanding exactly how PC technology is being used is therefore a significant first step in introducing change to improve cost-effectiveness.

Although largely ignored, the end-user cost of PC usage is by far the largest single element of a cost model. But, given the fact that the PC exists solely to improve the efficiency of a business process or task driven by the user, it is fundamental to measure the cost of end-user time spent running and serving PC applications. Just two hours a day of PC use for someone with an annual salary of £20,000 ($32,000) equates to £5000 ($8,000) per annum in PC-related costs. It has been shown by Green (11) that users will typically use their PC for between 2 and 4 hours per day, so this assumed cost of £5000 is a conservative estimate.

There will always be some elements of cost that cannot be directly measured. However, there are at two important measurable items of cost that most models have failed to include. These are, firstly, the amount of time wasted by users waiting for PCs to perform their operations, and secondly, the amount of extra time that it takes users to perform application tasks because they are not as proficient in using the functions and capabilities as they could be.

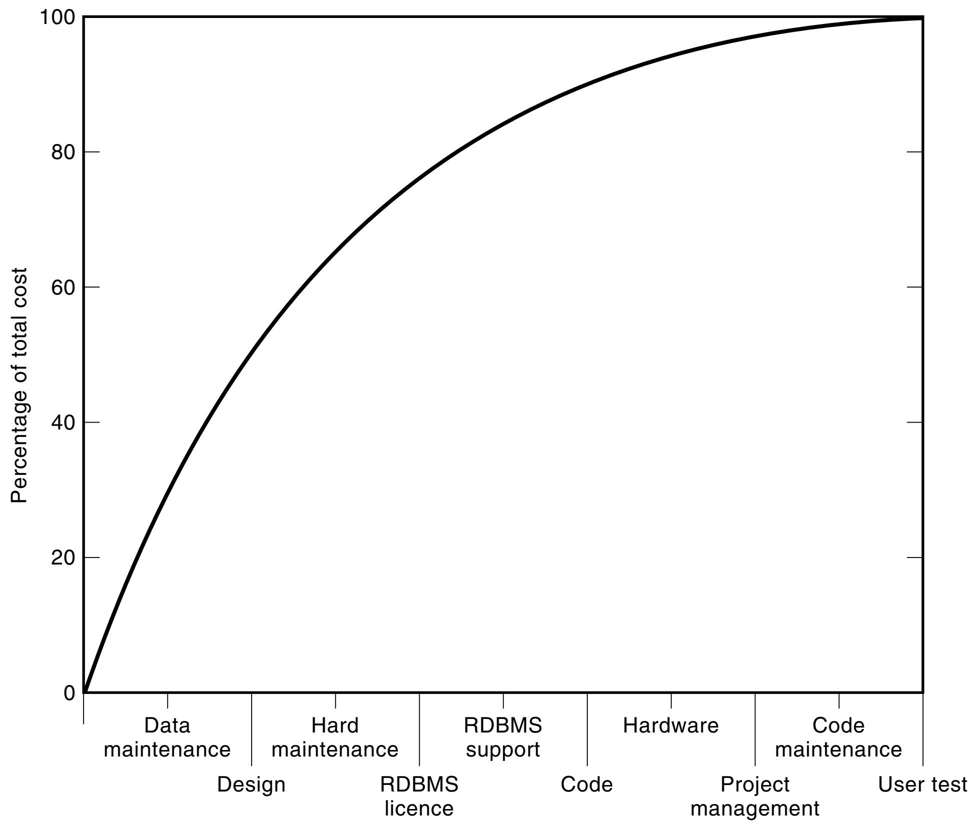

Figure 2. The costs of database systems studied by Bradley and Dawson.

Software now exists that will continuously monitor the use of a computer and record the extent of use of the machine, the software that is used, and the effectiveness of individual users in executing a particular software function. The following case study illustrates how such software was used at the engineering company studied by the authors to learn about the company's use of computers and make decisions to reduce the overall cost of ownership of their computers.

CASE STUDY: MONITORING COMPUTER USE

In the studied company, the Deskwatch software (Platinum Technology, Inc.) was used to monitor the company's computer network. The software was originally purchased to provide data for hard disk procurement decisions, but has proved invaluable in support of whole-life cost reduction decisions. It provides detailed PC-system performance and configuration data using a combination of Windows executables, Windows function libraries, and DOS executables on both networked and nonnetworked PCs (11). It consists of the *capture agent* component, which is installed onto each PC and performs the monitoring function, and the *agent manager,* which provides administrative control of the agent community. This allows decisions to be made on the scope of collected data. The capture agent continuously monitors the PC's activity and configuration details (the *capture* process), and records them to *event record files* on the PC's hard drive.

The Deskwatch analysis showed, not surprisingly, that a user's productivity is directly affected by the speed of their PC.

The analysis identified that 1.5% of all PCs have a *good* performance level, 25% have an *acceptable* performance level, 42% have a *poor* performance level, and 31% have a *critical* performance level. Effectively this shows that at the time of the analysis approximately 73% of PCs were failing to meet acceptable service levels. This illustrated a significant target area within which large potential productivity increases could be obtained.

Following the analysis, a program was initiated to:

1. Swap little-used high-specification machines for high-use low-specification machines
2. Buy in new machines on a rolling program

In addition the software identified, through the results shown in Figs. 3 to 6, that:

- WordPerfect (Corel Corporation Limited) and Groupwise (Novell Inc.) were the top applications, 50% of active machine time being used in these two applications.
- Active machine use was largely for only 1 to 2 hours per day. Only a very small number of machines saw 6 or more hours of use per day.
- Applications' demands on processor and disk are variable, and need to be taken account of in system design.
- Cost of ownership of specific PC models is affected by the applications used and their extent of use, giving rise to a range of annual costs of £1000 to £22,000 ($1600 to $35,200).

THE USE OF THE HELPDESK FOR OPERATIONAL DATA

Many companies operate a helpdesk to monitor problems encountered by users of their computer systems. Helpdesk soft-

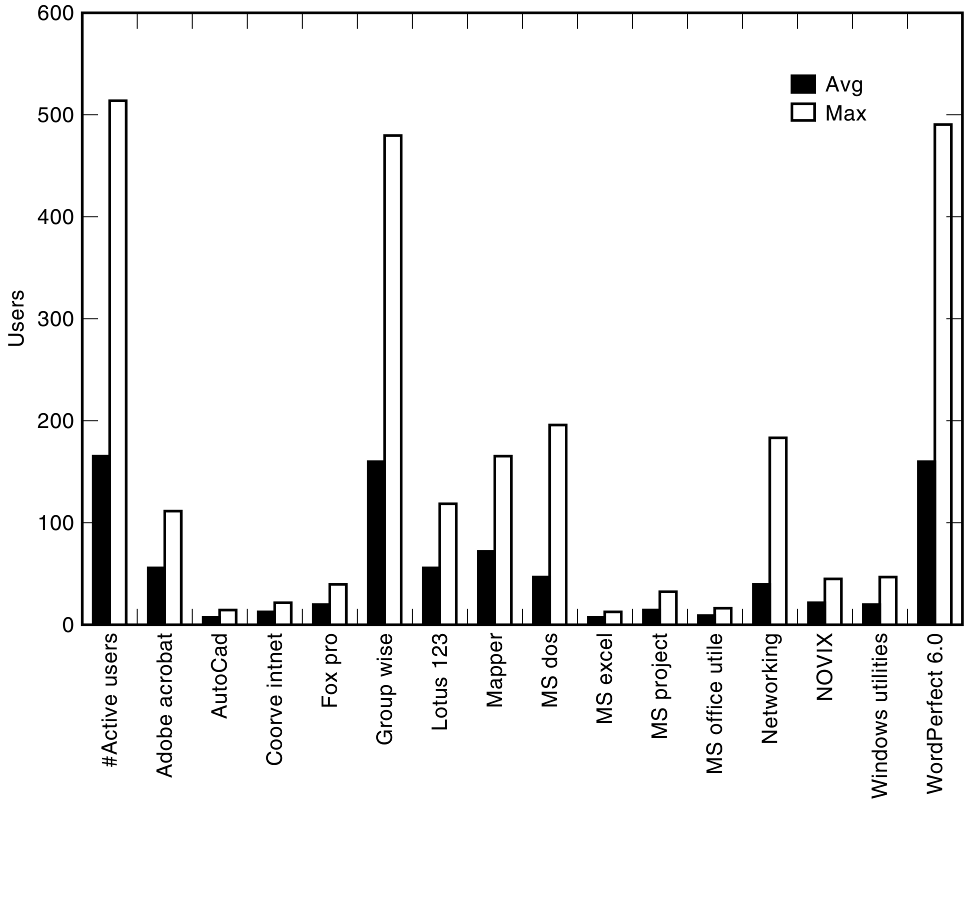

Figure 3. Application user counts.

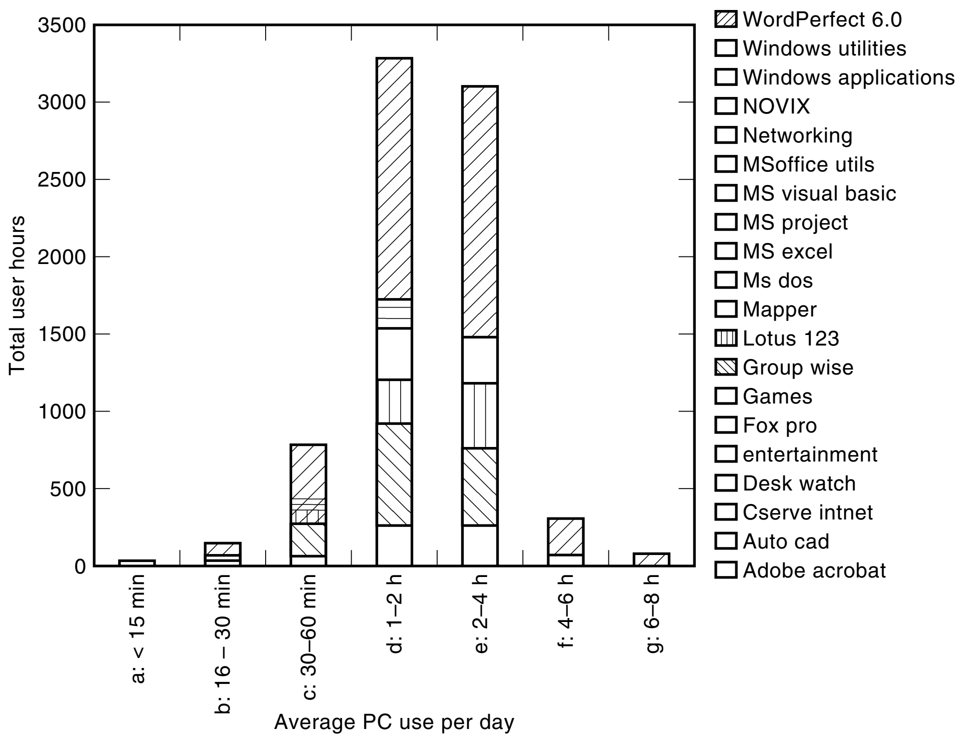

Figure 4. PC use per day.

ware is available to track the reporting of faults and the action taken to correct them. The traditional use of this software has generally been to improve the helpdesk operation. However, as reported in Bradley and Dawson (12), the helpdesk records can provide invaluable data on system use, reliability, and maintenance. With regard to computer system use the helpdesk can provide data on which software gives problems, and on whether the problems are due to failure in the software or to failure in the user's training in its use. With regard to the hardware the helpdesk can be used to record and supply data on:

- Mean time to failure of individual components and of the computer as a whole
- Software, machine, or network down time
- Repair action and the active repair time

These data can then be used in evaluations of the network, groups of components, or processes. An evaluation that demonstrates the use of helpdesk data is included in the section below.

CASE STUDY: USING THE HELPDESK TO MINIMIZE MAINTENANCE COSTS USING DETERMINISTIC METHODS

An analysis of maintenance options for the PCs at the engineering company studied by the authors was carried out to

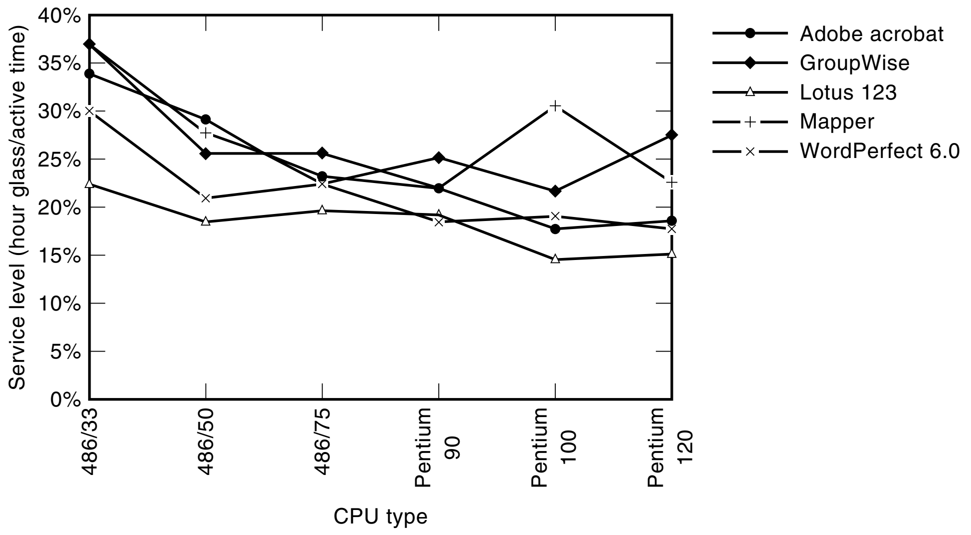

Figure 5. Performance level by application.

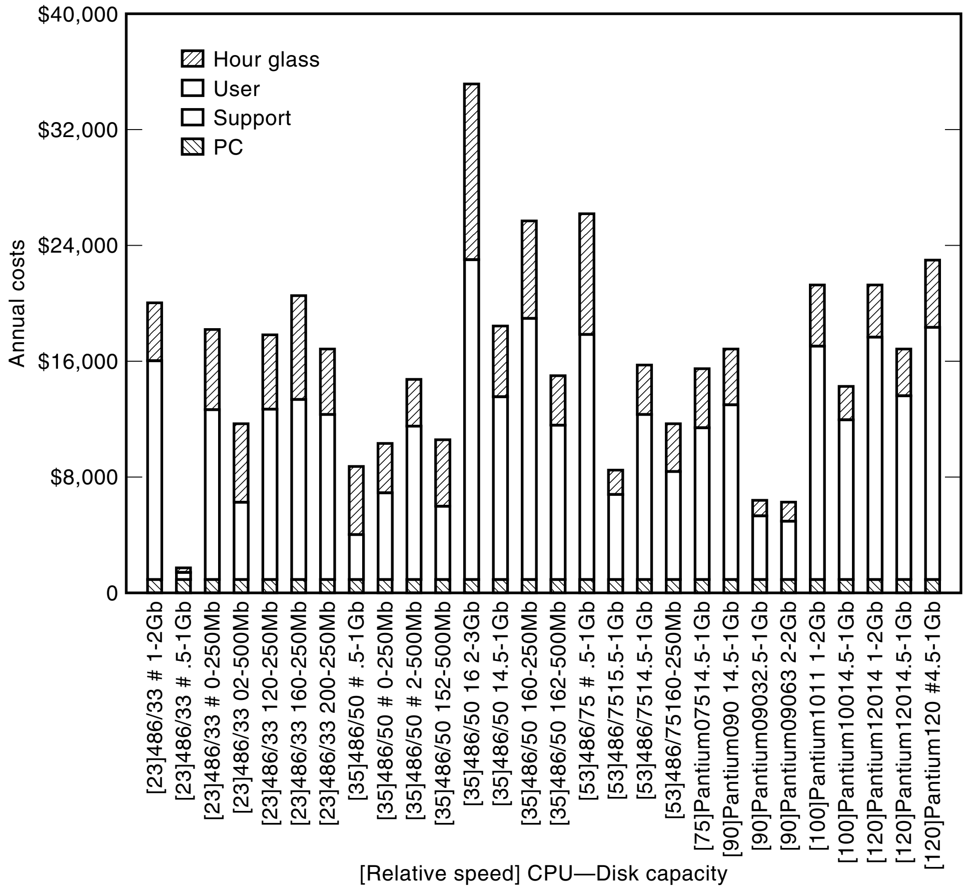

Figure 6. PC cost by specification.

determine the least-cost option. The data used were derived from three sources:

1. The helpdesk operated by the company
2. The output of the Deskwatch network monitoring software
3. The register of electrical equipment owned by the company

Data concerned with network events was extracted from archived helpdesk files. Analysis was initially concerned with categorization of events into hardware and nonhardware failures. Later analysis attempted a lower-level understanding of failure by ascribing the cause of hardware failure at the subcomponent level.

The company's implementation of the helpdesk system was only partial. For example, no record was maintained of the actual corrective action taken following a reported event, the spares consumed in the corrective action, or the time taken to effect a repair. This made the analysis particularly difficult. The taxonomy of events was only achieved by inspection of all the text statements for the telephone reporting of the event. Some sixteen thousand records were created in the first year of operation.

Data concerned with the use of hardware and software was taken from the Deskwatch system used throughout the company. Purchase dates of computing equipment were obtained from the company's register of electrical equipment, as a full inventory of computing equipment was not available from the helpdesk system.

The 341 PCs in the study were supplied by a number of different manufacturers. To test the consistency of data across all manufacturers, a comparison of days to failure was performed for each major component. This showed only minor differences between manufacturers that were not significant in the population considered, so for the purposes of the analysis the differences between manufacturers were disregarded when calculating MTTFs.

To assist in subsequent cost estimation a corrected set of component failure rates was created for the full population of 341 PCs. Table 1 presents failure characteristics as MTTF in

Table 1. Corrected Failure Rates for PC Subcomponents

Description	Failures	MTTF (h)	Replacement Cost (£)	Replacement Cost ($)
A drive	19	20,456	20	32
Battery	6	64,778	5	8
Fan	2	194,333	6	7.2
HD	28	13,881	179	286
Keyboard	15	25,911	20	32
Monitor	18	21,593	169	270
Motherboard	4	97,166	150	240
Mouse	24	16,194	15	24
Netcard	2	194,333	23	37
Other	14	27,762	—	—
Complete PC	132	2,944	—	—

hours. For the calculations in Table 1 the working day was taken to be 7.5 h and the number of days in use was taken as $\frac{5}{7}$ of the number of days elapsed since the computer was installed. Finally the calculation takes account of the time that PCs are in use. The Deskwatch software shows that PCs are typically powered for 90% of core time (the 7.5 h used above).

This gives the MTTF calculation as:

$$\text{MTTF} = \sum_{i=1}^{341} \frac{(\text{Endate} - \text{Pdate}_i) \times 7.5 \times 0.9 \times \dfrac{5}{7}}{N_i} \tag{2}$$

where

i = number of the PC in the data set
Endate = March 5, 1996
Pdate_i = purchase date for the ith PC
N_i = number of recorded failures for the ith PC

The cost data included in Table 1 are typical values taken from the September 1996 UK trade magazines for the components listed.

THE MAINTENANCE OPTIONS CONSIDERED IN THE CASE STUDY

Five alternative maintenance options were considered:

1. The previously existing practice of taking out a maintenance contract extension to provide on-site hardware maintenance for the first three years of life, the fourth and any subsequent year being supported separately to the maintenance contract (referred to as "As is" in Table 2).
2. Purchase of spares support only for the whole of life (years 1 to 4), diagnosis and support being provided by the company's own staff (referred to as "Spares only" in Table 2).
3. Purchase of support for the first three months only (to cover infantile failures), providing support from spares held at the company for the rest of the computer life ("In house").
4. Bulk purchase of PCs complete with spares for life ("Bulk purchase").
5. Support of the PCs using company staff with spares procurement as follows:
   a. A bulk buy of the high-obsolescence items (motherboard, hard drive, and fan), which are likely to be obsolete within the life of the PC
   b. Spares-only support for the low-obsolescence items after the first three months

   ("High obsolescence"). This provides protection from the possibility that a replacement will be unavailable due to obsolescence, but minimizes the initial expenditure on through-life spares.

Table 2. Net Present Values for Maintenance Options

Maintenance Option	NPV (£)	NPV ($)
Spares only	71,592	115,000
In house	88,312	141,300
Bulk purchase	108,955	174,300
High obsolescence	88,437	141,500
As is	61,630	98,600

A number of assumptions were made in deriving the cost curves for the five options:

- The labor costs used were the without-profit rate for the grade of staff used for maintenance.
- Hardware costs are typical values for single purchases from the September 1996 trade press, but reduced by 15% and 10% respectively for bulk purchases in excess of £40,000 ($64,000) and £25,000 ($40,000).
- Maintenance times are assumed to be the average figures of 20 min for a visit to either fix or establish the cause of a fault.

The net present values (NPVs) for these options are tabulated in Table 2. The discount rate used for NPV calculations was 6%.

From the results shown in Table 2 it can be seen that the current option, that of subcontracting the maintenance of PC hardware, appears to be the most cost-effective of the straightforward options available to the company. However, this is not the complete picture. Whole-life costing requires *all* costs to be taken into consideration, and when the costs of unavailability are included the results are substantially altered.

APPLYING AVAILABILITY PRINCIPLES IN THE WHOLE-LIFE COST CALCULATION

Availability is defined by Nicholas (13) as

$$A_0 = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR} + \text{MTTS}} \tag{3}$$

where

A_0 = operational availability
MTTF = mean time to failure
MTTR = mean time to repair
MTTS = mean time to support

It can be seen that for a given repair scenario A_0 will have a maximum or inherent value when MTTS = 0, that is, when the time to provide the support actions is zero. MTTF is an expression of reliability, normally quoted in hours.

To simplify the illustration the costs of availability are shown for only a small subset of the 341 PCs considered, namely, the computing system used by a group of eight engineers working in one particular room. The engineers had access to seven PCs, of various manufacture and specification, and of various ages. From the Deskwatch software it was known that:

- For around 90% of the normal working day the seven PCs are powered up.

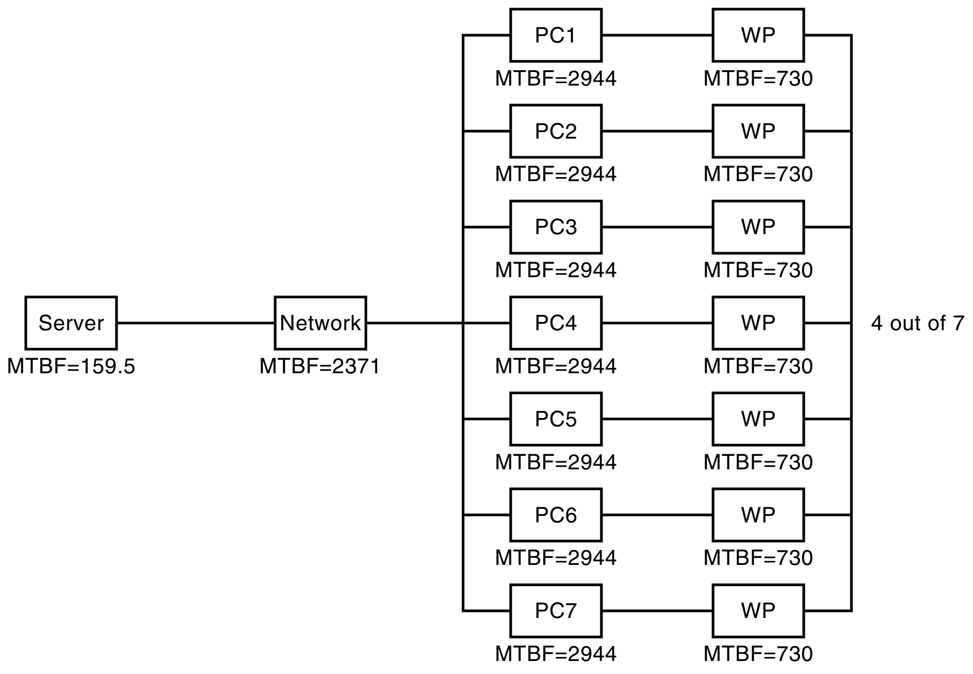

Figure 7. Reliability block diagram for a network element.

- The principal software package, WordPerfect, is in use for 50% of the active time.

In addition it is clear from Deskwatch that in this office the active PC use was less than 50%. Taking a judgement on these figures, we assessed that the minimum availability requirement of the PCs is probably four out of seven for the office use of the word processor to be available. That is, three of the seven PCs could suffer reduced function before the local computer system was considered unavailable to the engineers.

Knowing the inherent or actual availability of a system is only of value if the cost of achieving that availability is also known, and likewise the cost of any unavailability. Recent research by the Gartner Group (14) gives an indication of the cost of ownership of a networked PC as £8460 ($13,500) per year. Since no company that wants to stay in business will be likely to have assets that are providing a return that is less than the cost, the quoted cost has been taken to be the cost and/or benefit of the PC's function in assessing the availability of the system for a full year of operation. The section "Case Study: Monitoring Computer Use" earlier in this article records our assessment of PC cost of ownership as varying between £1000 ($1,600) and £20000 ($35,200) per year depending on the use and the software application. For simplicity the Gartner figure was used in this case study.

All of the intelligence in the previous paragraphs is incorporated in Fig. 7, a reliability block diagram (RBD) for the minisystem under consideration. RBDs are described in BS5760 (12) as a tool in the assessment of software reliability. As the principal software package is WordPerfect, only this software function is included in the diagram. Reliability figures (MTBF) are those derived for the full set of 341 PCs.

Word Perfect failures include any software-based query that has been placed with the helpdesk. It is necessary at this point to state that the failures described here are almost never associated with any defect in the WordPerfect software as such, but are usually due to lack of understanding of particular features of the software on the part of an individual user and are therefore, strictly speaking, training failures. Finally, the installation of WordPerfect is as network software, that is, the executable on a PC is dependent on the continuing existence of a session protocol between the PC and the network server. This relationship is described implicitly by the RBD.

From the reliability figures (MTBF) shown on the diagram, the total system reliability can be determined by use of the following formulas from Knezevic (15):

For series configurations:

$$R_s(t) = \prod_{i=1}^{NCI} \exp(-t/A_i) \tag{4}$$

where

R_s = system reliability
t = required time for the system to be functional
A_i = scale parameter of the exponential distribution
NCI = number of consisting items

For r out of n parallel configurations:

$$R_s(t) = \sum_{i=1}^{\text{NCI}} C_x^{\text{NCI}} R_i(t) \times [1 - R_i(t)^{\text{NCI}-x}] \tag{5}$$

where

$R_i(t)$ = item reliability
C_x^{NCI} = total number of combinations of NCI consisting items taken x at a time

Using Eqs. (2) and (3), the maximum availability that can be achieved (*inherent availability*) is 99.99%, and the availability achieved with the existing support options was 92.58%. Inherent availability is defined in Eq. (3) where MTTS = 0. In monetary terms this leads to losses to the organization approximately equal to 100 h/year for the seven-PC subsystem shown in the RBD. After adjustment for the 50% usage of WordPerfect determined from the Deskwatch software, this equates to approximately £59,000 ($95,000) per year for the population of 341 PCs, or about £172 ($275) per PC.

The figures for loss of availability provide access to an additional cost consideration for PC maintenance. The cost of availability was missing from the cost profiles previously generated and shown in Table 2. After adjusting the values in Table 2 to include the cost of unavailability, the lowest-cost option is the "Spares only" one. After inclusion of lost availability, costs this option has a NPV of £71,592 ($115,000), compared to the new figure of £89,964 ($144,000) for the "As is" maintenance arrangements. The company has now changed its maintenance contracting arrangements.

FAILURE DATA COLLECTION, MANIPULATION, AND USE

Data used in whole-life cost evaluation are usually best represented as a range of values or a probability density function rather than a single value. Purchase prices may be an exception, but data on computer use, failure, repair, and obsolescence are best represented by a probability distribution. The data analysis and the subsequent data manipulation in a cost model require specific tools for the statistical analysis to determine the fit of data sets to a range of standard probability density functions such as Weibull, lognormal, or exponential. In cost model manipulation, functions are required to generate likely cost values from probability density functions, and also to generate more complex cost probability density functions from the sum or product of probability density functions of modeled cost elements.

The maintenance case study above provides a useful demonstration of the use of whole-life cost and availability as evaluation techniques for IT systems. However, the adoption of a probabilistic approach to financial and reliability decisions is seen as desirable, as it provides more meaningful cost ranges over the life cycle. A pragmatic view of cost of ownership can be established, based on the cost of replacing physical components and the cost of unavailability (during the repair process). In the generally accepted life of a PC, three years or less, the annual costs drop in successive years as the incidence of failure declines. This is demonstrated in the next case study, concerned with cost of failure.

CASE STUDY: DERIVING FAILURE PROBABILITIES AND ESTABLISHING COST OF OWNERSHIP

A PC can be regarded as a modular assembly, and one that can be constructed from the assembled component parts in a matter of fifteen minutes or less.

The failure probability of a component or system can be estimated experimentally from field data, provided the population studied is reasonably large and the rate of occurrence of failure is sufficiently high to provide a high confidence level

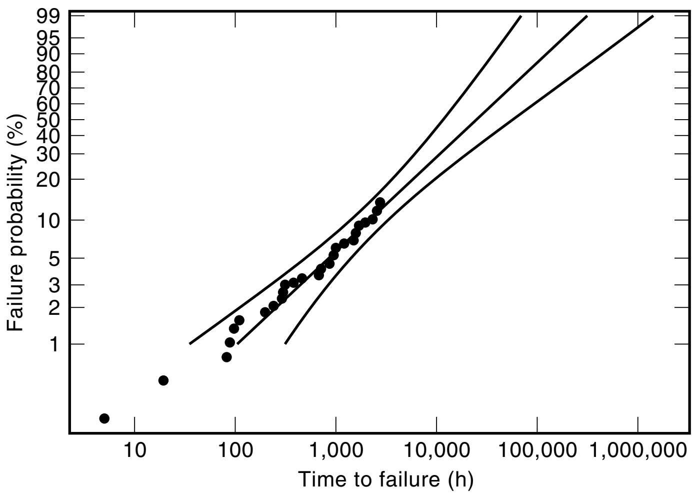

Figure 8. Typical failure probability data.

in the derived failure probabilities. For this article a large body of computers was studied for a period a little short of two years. During that time the majority of subcomponent design categories experienced more than thirty failures each. A probability plot for one subcomponent is shown in Fig. 8. The plot is for the population of keyboards in the 341 PCs studied. The distribution is a Weibull with a shape parameter of 0.76. The Weibull distribution is fully described in the context of engineering reliability by Knezevic (15); a Weibull distribution with a shape parameter approaching unity is consistent with an exponential distribution, as can be shown by the following taken from the Microsoft Excel Function Reference:

$$f(x;\, \alpha,\, \beta) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta} \tag{6}$$

where α is the shape parameter and β is the scale parameter or the mean. When $\alpha = 1$, $f(x;\, \alpha,\, \beta) = \beta e^{-x\beta}$. An exponential failure distribution exhibits a constant failure rate, and it is therefore legitimate in such cases to use the MTTF as the measure of reliability. A shape parameter α that is less than unity implies that the failure rate is not constant, with a tendency to more early failures. Similarly, a shape parameter above unity implies a tendency to more later failures. The importance to computer evaluation is that it shows the failure characteristics of the majority of PC components tend toward early failure.

Only one subcomponent within the PC population under study had a shape parameter approaching unity. This was the motherboard, with a shape parameter of 0.99. This sample of the total population suggests a MTTF of around 49,000 h for a motherboard. The motherboard, with a near-exponential distribution, could be expected to yield failures at an approximately constant rate. The remaining subcomponents would be expected to yield more failures in the early part of the computer's life.

Using the statistical analysis software tool MINITAB (Minitab Inc.), the results given in Table 3 have been obtained from source data given by the company helpdesk. Data for this study has been collected over a period of approximately two years, giving a total PC usage of over 600 PC years.

Table 3. Experimentally Determined Values of Weibull Parameters—PC Subcomponents

Equipment	α	β
Mouse	0.86	22,440
Motherboard	0.99	49,171
Keyboard	0.76	41,919
Hard disk	0.51	136,752
A drive	0.55	196,426
Monitors	0.76	58,395

The data from Fig. 8 are redrawn in Fig. 9 in the more convenient representation of the cumulative density function. The steeper curve during early life shows a probability of failure that reduces over time. The full study has resulted in similar outputs for the major PC components with the exception of the motherboard. The Weibull parameters for each component are provided in Table 3.

Calculation of the Maintenance Element of Ownership Costs

The calculation of the post-installation maintenance cost of computer hardware ownership can be summarized as follows:

$$C_o = \text{cost of component replacement} + \text{cost of lost availability}$$
$$= \sum_{i=1}^{n} P_t N(\mathrm{Cc}_i + \mathrm{Cr}_i + \mathrm{Ca}_i) \qquad (7)$$

where

Cc_i = cost of replaced component i
Cr_i = labor cost of replacing component i
Ca_i = cost of lost availability of component i to the organization
P_t = cumulative probability of component failure at time t after installation
N = number of computers
n = number of components in the computer

Calculation of Replacement Costs

The collected reliability profiles from all the major PC components have been assembled and used to establish the direct cost of unreliability of the complete PC, and the indirect cost of unavailability. Figure 10 shows the costs accrued over a period of 7000 h (approximately four years) in equal time slices of 500 h for the full population of 1000 computers in the network. For each time slice the number of failures for each component design that could be expected was calculated in a Microsoft Excel spreadsheet. This was achieved by substituting the Weibull shape and scale parameters from Table 3, together with the time to date in hours, into the EXCEL Weibull function. The replacement cost calculated in each case was then the sum of labor and material costs for the individual component. The labor costs used are those applicable in the company prior to outsourcing. The labor times used are those derived for the individual components from interviews with staff and contractors on the proportion of the time that is related to the active repair.

In our model each PC component is treated independently, so that if it is replaced it will count as new, with the failure probability of a new component, while all other components in the same PC continue their aging. The additional replacement components therefore create new sample populations for each new time slice, giving a new but smaller population for each time slice. This process continues for each subsequent time slice until there are too few components remaining in any population to perform meaningful calculations.

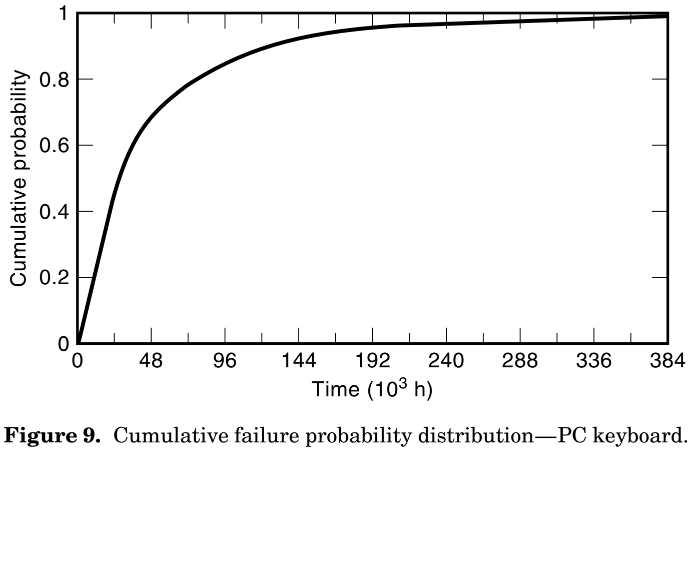

Figure 9. Cumulative failure probability distribution—PC keyboard.

Calculation of Cost of Unavailability

The derivation of the cost of unavailability used in the financial arguments was as described earlier in this article. The cost of the unavailability to the organization was calculated using the research by the Gartner Group (14) as £8460 ($13,536) per year, or approximately £5 ($8) per hour.

Because PC component failures appear to be skewed to the early part of life, the cost of unavailability and unreliability is similarly skewed into the early life of the machines. This is important in any computer evaluation. Where large numbers of machines are replaced at the same time in an organization, the immediate effect is likely to be a significant number of early failures—damaging to the organization's operational availability, creating a busy period for its maintainer, and possibly damaging to its reputation. This suggests that a program of rolling replacement for PCs would be advantageous for many organizations.

Helpdesk data can be used to provide the costs essential to the planning of a quality maintenance program as shown in the earlier case study. This is a use for help desk information not normally considered (16).

USING GENETIC ALGORITHMS TO SELECT THE LEAST-COST OPTION

Genetic algorithms provide a mechanism for optimizing the design of systems to minimize the whole-life cost. This is particularly useful when there are many possible values of the system variables. The method employed with genetic algorithms needs to recognize that the full data set must be built into a model. The model effectively has control of the selection process.

Genetic algorithms are appropriate when:

1. The number of variables in the problem is large. Different authors have different views on what is large, but

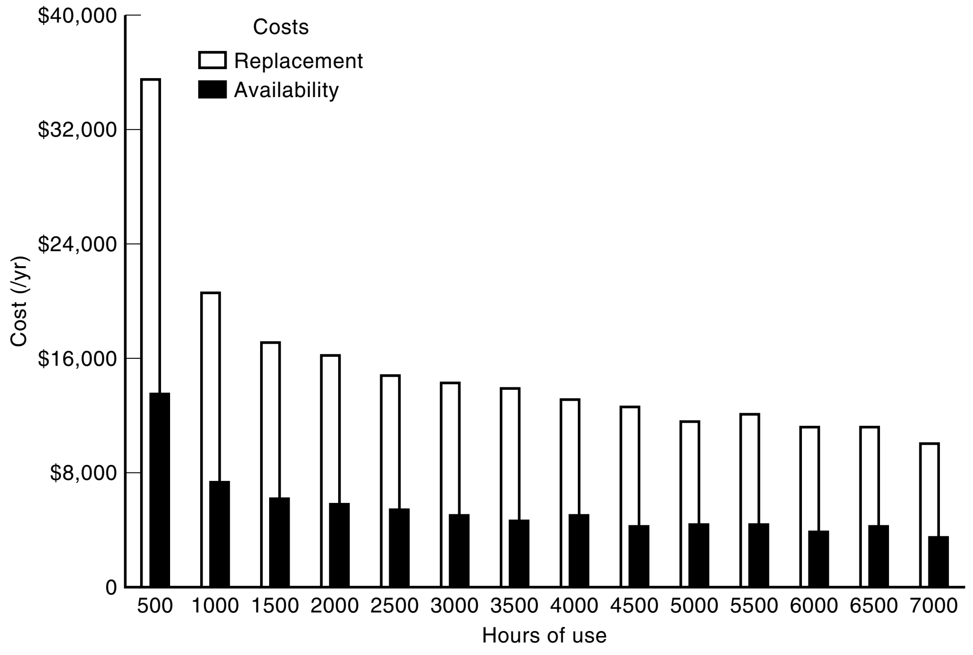

Figure 10. PC cost of ownership based on reliability and availability.

15 appears to be so regarded by most genetic algorithm designers. For smaller numbers of variables a deterministic, linear search is probably quicker.

2. All aspects of the problem are understood, that is, all the separate cost relationships are defined.

When a genetic algorithm is used in problems that fit these criteria, they perform generally better than alternative optimizing techniques.

CASE STUDY: USING A GENETIC ALGORITHM TO MINIMIZE THE COST OF LETTER PRODUCTION

The system design for typing letters for a three- year period were considered in the example company and reported by Bradley and Dawson (17). A balance was required between the PC specification, the word-processor version, and the decision for the authors to either type their documents themselves or write them by hand and then pass them to a typing pool. The whole-life costs must be calculated given a number of fixed parameters, such as typing speed and accuracy, document throughput, and labor rates.

The following steps employ genetic algorithms as an appropriate method to derive an optimum design to minimize whole-life cost, so that steps 2 to 5 of the whole-life-cost design method shown in Fig. 1 become:

2. Derive, through brainstorming techniques, a causal network of the relationships affecting the whole-life cost of an information system incorporating a relational database application.
3. Create a list of relations from the causal network in step 1.
4. Derive by experiment a set of heuristics and constraints for each relation.
5. Using the heuristics and constraints, create a mathematical model incorporating a genetic algorithm to establish the minimum-whole-life-cost design.

A causal network for the word-processing system, shown in Fig. 11, demonstrates the extreme complexity of the relationships and the number of interactions that can affect the total cost of a system. This very complexity is the main justification for the use of historical data and genetic algorithms. The search time for optimized solutions on large problems is reduced by the use of the genetic algorithm, whilst the use of historical data increases the probability of an accurate solution. Table 4 shows the cost of the word-processing system over an expected life of three years for each option.

This very simple example shows that it was the earliest version and an inexpensive computer that constituted the lowest-cost option through life. Clearly this example is out of date, but the method could be extended to cover the use of later operating systems and Pentium PCs. This would then

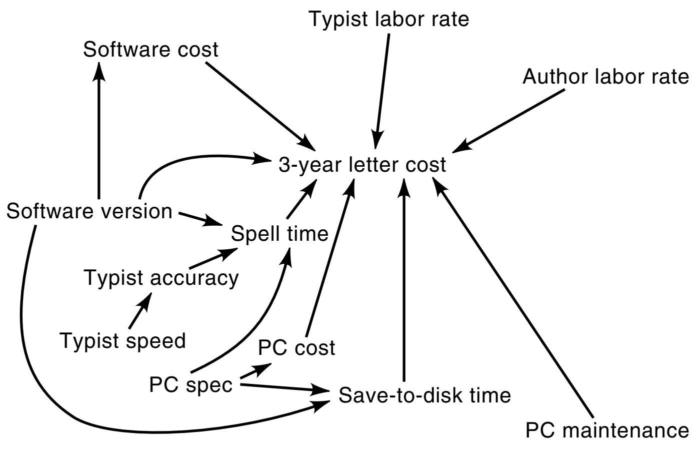

Figure 11. Causal network—the typing system.

Table 4. Typical Three-Year Costs of Producing Letters with a Number of Competing Options

			Cost (£)			Cost ($)		
PC	Spec.	Procedure	W.P. Version 1 (DOS)	2 (DOS)	3 (WIN 3.*x*)	W.P. Version 1 (DOS)	2 (DOS)	3 (WIN 3.*x*)
1	(80386)	Handwritten for typing	5,882	6,174	6,355	9,411	9,878	10,168
		Author types own	7,397	8,272	8,816	11,835	13,235	14,105
2	(80486)	Handwritten for typing	6,403	6,551	6,631	10,244	10,481	10,009
		Author types own	8,334	8,779	9,018	13,334	14,046	14,428

justify the use of genetic algorithms, which only truly become necessary as problems become large and complicated. Nevertheless, this example has shown that at the time of this analysis choosing other available options could cost over 50% more than necessary, yet the solution offered would be unlikely to be intuitive to the manager responsible for the purchasing decisions.

THE LIFE OF A COMPUTER SYSTEM

Determining whole-life costs for a computer system leads to some difficult questions of how long a life the system is expected to have and, if one component of the system becomes obsolete, whether that affects the life of the remainder. The intuitive response to the first question for most items of equipment is that it lasts until it is "worn out," meaning that it breaks down too frequently and starts to become too costly to repair. A car, for example, is usually scrapped when it reaches this point. A computer, however, does not fit in with this concept. Indeed, most office-based employees will know of computers that are no longer used despite still being capable of delivering the service they provided when first purchased. How many 80286 computers running MS-DOS word processors are in use today despite the fact they may still be capable of operation? Clearly, computer obsolescence can be caused by factors other than component failure. The next subsections therefore highlight some considerations reported by Bradley and Dawson (18) that should be taken into account when planning for the life of a computer.

Types of Obsolescence

There are two principle forms of obsolescence, technological and functional, and these affect the provider and user of products and services in different ways.

Technological obsolescence is a problem that afflicts all delivered products or services to some degree, and it has the greatest affect in rapidly changing technologies such as the computer industry. An example is the punched-card reader, which has now been superseded by much more sophisticated and user-friendly forms of input. The effect of technological change and obsolescence is to force new purchases on the user, often providing opportunity and competitive advantage that would mean commercial disaster if ignored.

Functional obsolescence is a secondary problem, generally caused by technological advance elsewhere. The card trays and cabinets used for storing the punched cards exemplify this. While this office furniture may still be the most up-to-date way of storing punched cards, it is clearly no longer required when the cards themselves are no longer used.

In addition to the typing of obsolescence, there is also the issue of the standpoint from which obsolescence is viewed. There are a great number of articles in the literature written from the position of a manufacturer (19–23), providing advice on strategies to retain or gain market leadership. There is also advice available from the literature for organizations in the middle of the supply chain (23), where it is clearly important to monitor component obsolescence to avoid both lost sales and obsolete stock.

This section is concerned with both forms of obsolescence described above in the context of common computing applications, the hardware required, and the consequent implications for cost of ownership of IT systems. The context is that of the end user.

The Root Causes of Obsolescence

Obsolescence affects a computer and the associated applications in a number of ways:

- Software may be subject to an upgrade by the software supplier, creating technological obsolescence of the earlier software version and possible consequent obsolescence of the computing hardware.
- The demands made on a computer may be increased within the capability of the software, but beyond the capability of the hardware, creating functional obsolescence of the computer.
- The computer may no longer be supported in that some component is no longer in production or is no longer in demand—an example of technological obsolescence.
- The business may change, so that the software must change in line with the new business, causing functional obsolescence of the old software (version), again with a possible consequent obsolescence of the hardware.

An understanding of the likely occurrence of obsolescence is important in understanding the cost of ownership of a computer in the organization. The first three of these categories are dealt with below; the fourth category is covered only briefly, as it is business-specific and is best dealt with by carrying out a risk analysis of the business future.

The Software Upgrade Problem. This form of obsolescence occurs where the software being used is subject to an upgrade by the software supplier and where the upgrade requires an increased-specification computer, or where the software is no longer supported at all (withdrawal).

An indication of the volatility of software can be gained from the time between full releases of packaged business soft-

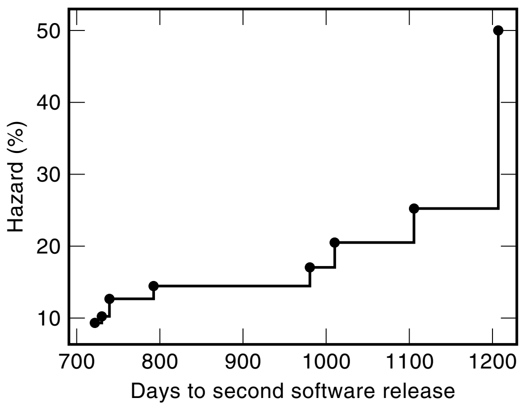

Figure 12. Software releases, based on elapsed time for two releases.

ware. Clearly, not all releases cause an improved specification to be purchased, but usually the pressure is increased on the business to upgrade. A review of releases for some popular Microsoft Windows-based software carried out in May 1997 revealed the following: Software vendors generally provide support for software for the current and previous releases only. Some may provide informal support for earlier releases, but may not formally enter into support agreements based on old software versions. This implies that the time between successive releases is significant. It has also been quite normal for three contiguous PC software releases to move the software from DOS through Windows 3.*X* to Windows 95, with significant changes to hardware specification at each change. Based on a total of eight popular software packages with a total of 25 releases, the time between successive software releases is estimated to follow a lognormal probability distribution, with a hazard plot as shown in Fig. 12. The hazard function provides an indicator of the change in instantaneous renewal (or obsolescence) rate with respect to time t over the life of a component. It is normally used to present the *bathtub curve* used by many texts to show the normally expected failure characteristics of engineering components.

Figure 12 indicates that the instantaneous risk of a second rerelease of a single PC software package increases sharply with time, and that the expected time to rerelease of any one piece of software is 1160 calendar days. This effectively means that if an organization was fortunate enough to buy a software licence for a business application on the first day that the version was available, the chances are that it would be considered obsolete around 1160 days later, or approximately three years. Unfortunately, the situation is not normally as clear cut. Most organizations operate a small number of key PC software packages, but also a larger number of less-used ones. A large organization may have as many as twenty commercial off-the-shelf packages in use, with an approximately random spread of release dates. If two commercial application packages are running on one PC, either of the packages may become obsolete, and this may force a change in the hardware or operating system that will, in turn, render the other package obsolete. If we assume that one of the software licences is purchased in mid version, the expected time to obsolescence drops to 870 days. As the number of key software packages increases, the expected time to a second rerelease decreases to around half the expected time for two releases (580 days, or 21 months). What happens in practice is that the dominant packages in an organization are the ones that decide if the organization upgrades its hardware. For the majority of machines, software technological obsolescence causes expenditure on both software and hardware between 580 and 870 days from the joint purchase of hardware and software, despite the fact that the hardware in isolation has a low probability of failure as described by Bradley and Dawson (12).

This particular form of software obsolescence also has an effect on the software development fraternity: as compilers and their versions change, so to must the developer. Developer skill obsolescence is therefore exactly paralleled by the obsolescence characteristics of the software that the developers use. All this means that there is a likelihood that at two- to three-year intervals the software or software version will change, with a corresponding risk to hardware and training.

When Demand Outstrips Performance. Obsolescence can occur where the demands made on a computer are increased within the capability of the software but beyond that of the hardware. An example of the phenomenon is an initial use of a word-processing package to produce simple letters and documents, followed by ever more sophisticated and larger documents as users grow in confidence and awareness of the software.

Bradley and Dawson (17) showed that for a word processor package the learning period was around two years, during which time the PCs being used had been stretched in terms of processor speed and memory. This was also demonstrated by Green (11), who recommended in his 1997 study of computer use for a medium-sized engineering company that ten Pentium 90 machines less than 2 years old should be replaced with large-memory Pentium 200s on the grounds of inadequate performance.

The brunt of this problem is borne by the hardware; the organization is effectively pressured into the purchase of the latest technology that will run the older software more effectively. If Green's experience is typical, then the need to upgrade hardware might be expected to occur at around two years.

A Break in the Spares Supply Chain. Effective obsolescence occurs when the PC is no longer supported because a component is no longer in production, or has been overtaken by new technology. In either situation spares to support in-service machines may become unavailable.

The clearest example of this problem, and the one that to date has probably determined PC obsolescence more than any other, is the processor installed in the PC. Data from Intel (24) show that the obsolescence effect of the chip has been accelerating with each successive design improvement. Figure 13 shows the days between significant design improvements of the 80*x*86 chip from the original 8086 to the Pentium II. Strictly, Fig. 13 shows virtual obsolescence, as the chips can still generally be purchased from the manufacturer. However, the supply chain tends to support the current production and/ or the hardware that is covered by warranty. In the event of a motherboard failure that requires a replacement chip, this effectively puts a limit on the spares availability and leads to machine obsolescence in around three years. Chip improvements, though, have been occurring more frequently recently,

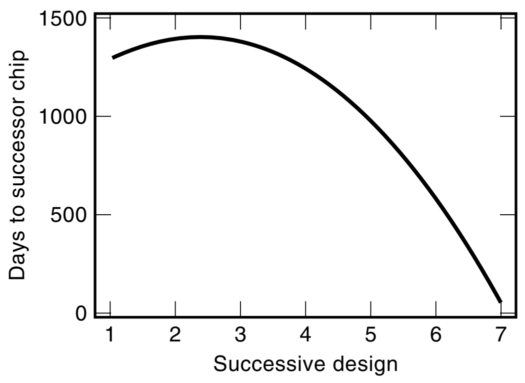

Figure 13. Days between significant Intel chip improvements.

and the average time has fallen to less than two years for the most recent PC sales.

The Problem of the Business Moving On. The final cause of obsolescence is found where the business moves on and the software in use has to change. This will often be to meet the demands of a (new) prime customer, or because part of the core business has changed. It is an issue that needs careful attention from both the IT management and the main board of an organization. As it will be different for different organizations, one may generalize only on policy for handling the risks, as methods for handling these are likely to be the same for all organizations.

The Need to Identify and Manage Risk. Organizations investing in IT have a need to understand the various risks that can affect the cost of the investment. It follows that companies investing in IT need a well-developed risk management method, largely in accord with the principles described by Pressman (25). Risk management can be expensive, however, Bradley and Dawson (18) reported the experience of one large UK organization that usually used risk management methods only on projects with a value greater than £10M ($16M).

The importance of risk management and continuous risk review is illustrated by Sillitoe (26). In this case the company was requested informally by the main customer to move from WordPerfect to Word. The change, if carried out, would have involved a move not only to a new word processor, but also to a new operating system, Windows 95, which at the time would not run on 70% of the company's PCs. The cost of a change of the full PC population (around 700 were involved) would be in excess of £1M ($1.6M) for hardware alone. The risk that might have been anticipated in this case was the obsolescence of the operating software for the extant PC population. Continuous review of the risk would have identified the need for an earlier and gradual migration to a new operating system. Thus, risk management for IT needs to include obsolescence of software and hardware.

The data sources for establishing the obsolescence risk in a particular organization's computing systems will, of course, vary. The sources for this view of obsolescence were taken from the supplier's web page, from measuring the varying patterns of file creation, and by looking at the pattern of software upgrades for the main software packages in the organization. In short, the data are not on the shelf. They have to be inferred from data collected for other reasons. The best data in this situation are those that are always collected as a matter of routine because the organization requires them for other reasons. Legislative, audit, or contractual reasons are good examples. An exceptionally useful source of data is the helpdesk run by large organizations to support IT. This is an example of the audit type of data source. Helpdesks are invariably faced with a service-level agreement that ensures that all calls are logged with a date-and-time stamp and, importantly, against a configurable item of software or hardware.

CONCLUSION

This article started with the premise that evaluation of a computer is likely to require more than the evaluation of the hardware of one machine. The term "computer" is likely to involve a number of machines connected in a network, a range of software running on the network, and even the associated infrastructure of operating personal, procedures and processes, maintenance contracts, and licenses.

In order to evaluate this complex collection, a technique known as whole-life costing is proposed. This involves a rigorous assessment of all the costs in a system over the system life to ensure the system is affordable and is the least-cost option. Whether parts of the system are to be developed in house or the system is built entirely from off-the-shelf components, the complete computer system must be carefully designed to achieve the minimum whole-life cost, and a methodology based on nine iterative steps is suggested to achieve this aim. Some costs associated with a computer may be difficult to measure and can only be the subject of the vaguest estimates, but even in these cases whole-life costing can be a valuable exercise when different options are being compared.

The chapter has shown the importance of collecting data to support the costing process. While some costs can be deduced from benchmark tests, other costs can only be derived from data collected from existing systems in service. Examples have shown that software to monitor computer use and helpdesk records can be particularly useful in providing such data. The analysis of the data can then be complex, and may require a probabilistic approach. Other spreadsheet add-ins can provide genetic algorithm software for deriving minimum-cost solutions when there is a large number of variables to consider.

Is computer evaluation worthwhile? Although it may be a complex activity to collect and analyze the data required to fully evaluate a computer system, the examples given in this article have shown there are significant gains to be made, particularly when the savings can be applied across a large organization.

Professionalism, paradoxically, can stand in the way of achieving the best computer system configuration. Designers of any discipline will generally hold the view that their design method takes account of all the necessary parameters and that the final design approaches the best possible. This leads to resistance to design by whole-life costing, as adopting that approach requires new rigor in the design process. At every stage of the top-down design process the question must be asked, "Which configuration will give the lowest through-life cost?" However, whole-life cost tradeoffs in the system design process take time and effort, and the evaluation process itself will inevitably cost more.

Full whole-life costing of a computer or software system is expensive. The process carried out in 1994 to select a new generation of document production hardware and software cost in excess of £50,000 ($80,000). It therefore makes sense to understand which equipment, processes, and software generate the highest costs, enabling an organization to focus the data collection effort on those areas that will yield the biggest return. An organization must understand how it is spending its money in great detail before collecting low-level cost data. Only by obtaining this understanding can an organization be sure it will avoid spending more money collecting data than can be saved by applying the lessons provided by the data. Breaking down the resistance of IT staff to whole-life costing is really only achieved by proving that solid savings can be made by careful application of the technique. Success breeds success.

BIBLIOGRAPHY

1. *The Little Oxford Dictionary of Current English,* Oxford: Clarendon, 1986.
2. Sun Microsystems Computers, *Stop the Technology Madness,* Bagshot, U.K. 1998.
3. British Standards Institute, *Draft British Standard IEC 300-3-3,* 1995.
4. B. S. Dhillon, *Life Cycle Costing,* New York: Gordon and Breach, 1989.
5. I. Sommerville, *Software Engineering,* Reading, MA: Addison-Wesley, 1992.
6. B. W. Boehm, *Software Engineering Economics,* Englewood Cliffs, NJ: Prentice-Hall, 1989.
7. L. H. Putnam, A general empirical solution to the macro software sizing and estimating problem, *IEEE Trans. Softw. Eng.,* **SE-4**: 345–361, 1978.
8. W J. Fabrycky and B. S. Blanchard, *Life Cycle Cost and Economic Analysis,* Englewood Cliffs, NJ: Prentice-Hall, 1991.
9. M. Bradley, Life cycle costing, in J. C. Newton (ed.), *Engineering through Life Support for Profit,* Derby: Rolls-Royce, 1994.
10. M. Bradley and R. J. Dawson, Software life cycle metrics applied to 4GL applications, *Proc. Software Quality Conf.,* University of Abertay, 1995.
11. D. Green, Using total cost of ownership techniques to determine hardware and software replacement strategies at Rolls-Royce and Associates Limited, M.Sc. Dissertation, Loughborough University, 1997.
12. M. Bradley and R. J. Dawson, Reducing the cost of IT ownership, *Softw. Quality J.,* **6** (2): 113–125, 1997.
13. C. S. Nicholas, Supportability engineering—the economic dimension, *Proc. Soc. Logistics Eng.—Annu. Int. Symp.,* 1996, pp. 151–156.
14. Gartner Group, Intel signals expensive PC management, *Computing,* October 3, 1996, p. 14.
15. J. Knezevic, *Reliability, Maintainability and Supportability—A Probabilistic Approach,* London: McGraw-Hill, 1993.
16. R. Marcella and I. Middleton, Key factors in help desk success, an analysis of areas critical to help desk development and functionality, R&D Report 6247, British Library, 1996.
17. M. Bradley and R. J. Dawson, How can I cut the cost of my computer network? *Proc. First Int. Conf. Managing Enterprises—Stakeholders, Engineering, Logistics and Achievement (MESALA'97),* Loughborough, 1997, pp. 575–580.
18. M. Bradley and R. J. Dawson, An analysis of obsolescence risk in IT systems, *Softw. Quality J.,* **7**: 2, 1998.
19. Y. Gutgeld and D. Beter, Are you going out of fashion? *McKinsey Quart.,* **3**: 55–65, 1995.
20. B. R. Nault and M. B. Vandenbosch, Eating your own lunch: Protection through preemption, *Organization Sci.,* **7**: 3, 1996.
21. M. C. Neff and W. L. Shanklin, Creative destruction as a market strategy, *Res. Technol. Manag.,* **40**: 3, 1997.
22. R. Coredero, Managing for speed to avoid product obsolescence: A survey of techniques, *J. Product Innovation,* **8**: 4, 1991.
23. R. B. Handfield and R. T. Pannesi, Managing component life cycles in dynamic technological environments, *Int. J. Purchasing Mater. Manage.,* **30**: 2, 1994.
24. Intel, web page http://www.intel.com/pressroom/quickref.htm.
25. R. S. Pressman, *Software Engineering: A Practitioners Approach,* London: McGraw-Hill, 1994.
26. J. E. Sillitoe, *WordPerfect versus Word,* Internal Rolls-Royce and Associates memo JES/CTO, Derby, 1997.

M. Bradley
R. J. Dawson
Loughborough University