ABACUS, Vol. 41, No. 1, 2005

R. G. WALKER AND G. R. OLIVER

Accounting for Expenditure on Software Development for Internal Use

The methods accepted by Australian, International, U.S. and U.K. Accounting Standards for the treatment of expenditure on software development are inconsistent, and permissive. A host of methods for recording capitalized software in terms of those standards is identified by reference to an illustrative case study. It is questionable whether many in-house developed software applications satisfy the professionally endorsed definition of 'asset'. Moreover, even if accounting standards significantly reduce the range of options for capitalizing expenditure on software development, there would still be many values which could be assigned to capitalized software. That suggests that those 'measures' are not reliable, so that it would be inappropriate initially to recognize software expenditure as an 'asset'. It is contended that expensing all outlays on software development as they are incurred (accompanied by reporting that expenditure as a line item in statements of financial performance, and expanded disclosures in notes) is likely to provide a clearer and more useful report on business operations than the alternative of capitalization, amortization and subsequent assessments of whether or not recorded values should be adjusted for 'impairment'.

Key Words: Accounting; Capitalization; Enhancements; Expense; Intangibles; Maintenance; Software development.

There are a variety of ways in which firms may incur costs on software. Many organizations expend considerable sums on off-the-shelf or packaged software, licence fees, maintenance and sponsored development. Some firms, particularly in the financial services industry, adapt commercial packages to suit their own needs, incurring considerable sums on preliminary analysis, programming and testing before modifications are put into production. Some build their own applications. Some develop application software for on-sale or licensing to other parties. This article focuses on software development for internal use.

A major concern from an accounting perspective is the determination of what software expenditure (if any) should be capitalized (i.e., treated as the purchase of an asset), and what should be expensed? That judgment may have a material effect on the content of financial reports to external stakeholders, particularly in the determination of periodic profit. It also affects the information provided

---

R. G. WALKER (r.walker@econ.usyd.edu.au) is a Professor of Accounting and G. R. OLIVER a Lecturer in Business Information Systems at The University of Sydney.
The authors wish to acknowledge the research assistance of Bernadette Carr, and the constructive comments of referees and the editor.

internally to boards and senior managers regarding the expenditure incurred in providing a firm's IT functions, and invested in new software development projects. In recent years there has been a major shift in company 'assets' from tangible to intangible (Aboody and Lev, 1998; Sullivan and Sullivan, 2000). Events surrounding high profile firms which had previously reported high investments in intangibles (e.g., Enron, Global Crossing, WorldCom) have focused attention on the accounting treatments of so-called assets that are not saleable or otherwise convertible into cash.

Preparers of financial statements may face incentives in the form of bonus plans, performance-based remuneration or stock options to manipulate reported profits to achieve targets or increase executive compensation (see, e.g., Healy, 1985; Holthausen *et al.*, 1995; Healy and Wahlen, 1999). One key way to 'manage earnings' or manipulate reported profits is to selectively capitalize expenditure (such as on software development). For that reason alone, there is a case for clarity in accounting rules governing the treatment of software expenditure.

Indeed, some indications about the significance of decisions to expense or capitalize software are apparent from recent Australian annual reports or media commentary:

- The telco Telstra Australia adopted the policy of recording direct costs as software assets 'where project success was regarded as probable', and reported $3,181 million in capitalized internal use software that was being amortized over a 'weighted average of 6 years for fiscal 2003 (2002: 5 years)' (*Telstra Annual Report*, 2003).
- The 2002 financial statements of the Commonwealth Government of Australia, showed that $2.99 billion of capitalized software was valued at cost or replacement cost, and was being amortized over periods as long as twenty-eight years (*Commonwealth Government of Australia, Consolidated Financial Statements for the Year ended 30 June 2002*).
- Australia's four largest banks wrote-off more than $400 million of software during 2002–3, while software capitalization reached $1.97 billion. Analysts commented on a lack of 'conservatism', and suggested that amortization rates affected 'the quality of earnings' (*Australian Financial Review [AFR]*, 25 November 2003). Criticism was particularly directed at the National Australia Bank, which reported $301 million in capitalized software and disclosed that it was amortizing information systems projects over three to ten years (*NAB Annual Report*, 2003)—a period that media commentators contrasted with 'the usual timeframe of three to five years' (*AFR*, 25 November 2003).

This article explores the options currently available to record software expenditure in terms of conventional historical-cost based recording procedures (and in the absence of accounting standards proscribing capitalization). It focuses on the accounting treatments adopted by firms that develop software for their own use or in servicing their clients (rather than for on-sale or licensing). Consideration is given to whether in-house developed software could be regarded (or recognized)

as an 'asset' in terms of definitions and tests embodied in conceptual framework documents, such as the Australian profession's SAC 4 (AARF & AASB, 1995). Notwithstanding the possibility that software development expenditure could be regarded as an asset in terms of the profession's current conceptual framework, it is argued that the choice of what accounting treatment is appropriate should be based on the decision-usefulness of the information produced by either capitalizing or expensing outlays on software development for major users, such as managers, directors and external stakeholders.

It is contended that expensing all expenditure on software development as it is incurred, together with supplemental disclosures about expenditure on material projects, is likely to provide a clearer and more useful report on business operations than the alternative of capitalizing elements of that expenditure, and then amortizing that asset over subsequent periods. The latter accounting treatment inevitably relies on a series of contestable assumptions so that the asset valuations and cost data produced are incorrigible (i.e., impossible to verify or falsify; see Thomas, 1974) and may distort reports of the profitability and financial position of firms which are incurring major expenditure in this area, and may defy analysts' attempts to unscramble the effect of a series of accounting choices.

ACCOUNTING TREATMENTS OF SOFTWARE EXPENDITURE

The term 'software development' is used here to refer to the processes of analysis, design, programming, testing, implementation and ongoing modifications in order to meet a stated business need. 'Development is about extracting requirements from the business and translating those into a solution' (White, 2001).

With few exceptions (e.g., Burns and Peterson, 1982; Paulsen, 1983; Gannon and Parkinson, 1983; Berger, 1988; Aboody and Lev, 1998; AICPA, 1998, Dempsey, 1997), there has been little discussion in either the accounting or the information systems literatures concerning the appropriate accounting treatment of expenditure on in-house software development for internal use. Indeed, bodies responsible for producing accounting standards have presented conflicting views on this subject. Initially the U.S.A.'s Financial Accounting Standards Board in its Statement 2, *Accounting for Research and Development Costs* (1974) made only passing reference to software development costs, and concluded that costs incurred for internal use should be expensed as incurred, while 'intangibles purchased from others' should be capitalized and amortized. This was apparently reversed in an FASB Interpretation 6 (1975) which indicated that software expenditure that constituted 'the acquisition, development or improvement of a process by an enterprise for use in its selling or administrative activities' should *not* be regarded as R&D, and as such need not be immediately expensed. FASB Technical Bulletin 79-2, *Computer Software Costs* (1979), reiterated that Statement 2 and Interpretation 6 did not require that all expenditure on computer software be regarded as R&D (and hence expensed), and suggested that capitalization decisions can only be made on a case-by-case basis.

The lack of clear concise guidance was criticized by some U.S. practitioners:

> Because of the FASB's position that the purpose (intent) and source (purchase or develop) of the computer software should determine whether costs are capitalized or expensed, software development companies are faced with two choices. They may follow current accepted practice and expense most software development costs—reflecting unfavorable short-term financial results—or circumvent the intent of the FASB through *creative* financing arrangements. (Burns and Peterson, 1982, emphasis added)[1]

A decade later the FASB issued SFAS 86, *Accounting for the Costs of Computer Software to be Sold, Leased or Otherwise Marketed* (1985a), which has been described as 'the only exception in the U.S. to the full expensing rule of R&D (SFAS 2)' (Aboody and Lev, 1998). The FASB appears to have responded to an earlier statement by the Securities and Exchange Commission proposing a prohibition on the capitalization of costs incurred in developing software for sale or lease (SEC, 1983). However, while FASB Statement 86 only concerned expenditure incurred by software developers, an appendix suggested that expensing of costs on software development for internal use was 'not improper':

> The Board concluded that . . . accounting for the costs of software used internally is not currently a significant problem and, therefore, decided not to broaden the scope of this project nor add a project on internal use software to its present agenda. The Board recognized that the majority of companies expense all costs of developing software for internal-use. And the Board was not persuaded that this current predominant practice is improper. (para. 26)

While the FASB lent its support for 'expensing', it also supported a further review of the topic by the AICPA's accounting standards executive committee, which issued an exposure draft in 1997, and a more formal 'statement of position' in 1998. Participants in that project evidently disagreed with the FASB's suggestion that the subject was not a 'significant problem'. They observed:

> Accounting practices for internal-use software are diverse as a remit of a lack of authoritative guidance and escalating costs totalling billions of dollars annually. Some entities expense all costs as incurred, some capitalize most costs and some capitalize the costs of purchased internal-use software but expense costs of internally developed internal-use software. None of these entities is necessarily violating any standards. (Ameen and Noll, 1997)

The AICPA's 1998 Statement of Position endorsed capitalization of some of the costs incurred in software development (notably direct costs of materials and services, payroll costs for employees for time spent directly on the project, and interest costs incurred in the project). Capitalized software was to be amortized over its estimated useful life in a 'systematic and rational manner', and the 'impairment test' in FASB Statement 121 (1995) was to be applied. Given that

[1] While Burns and Peterson (1982) did not explain how 'creative financing arrangements' could be used to capitalize software development costs, Gannon and Parkinson (1983) suggested that 'R&D partnerships' and 'product financing arrangements' were being used to circumvent U.S. rules requiring expensing software developed in-house. Another possibility is as follows. Software development is undertaken by a subsidiary, and the parent then purchases the software from the subsidiary. Before year-end, the subsidiary is liquidated, so that no inter-company eliminations would be required.

there is no external referent in the form of a market price for used software, such allocations would be incorrigible.

The AICPA's statement was consistent with statements by the International Accounting Standards Committee in IAS 38, *Intangible Assets* (1998a), that expenditure on internally developed computer software should be recognized as an intangible asset, if 'the cost of the asset can be measured reliably'. In contrast, the U.K. accounting standard SSAP 13, *Accounting for Research and Development* (1989) allows, but does not require, this treatment (assuming that software could be regarded as an intangible asset within the meaning of the standard—discussed below). Similarly, both Australian Accounting Standard AAS 13, *Accounting for Research and Development Costs* (1983), and the identically titled AASB 1011 (1987) permit capitalization of expenditure on the development of a 'new product', to the extent that such costs 'are expected beyond reasonable doubt to be recoverable'. The term 'product' was defined in AASB 1011 as including 'product, service, process or technique' and as such could be (and has been) interpreted as encompassing processes for the delivery of financial services (see Dempsey, 1997).

At this point, mention might be made of the inclusion of seemingly inconsistent or contradictory statements within specific accounting standards. With the exception of the FASB's SFAS 86 (1985a) dealing with software development for sale, the AICPA (1998) statement on internally developed software, and other interpretations issued by professional bodies regarding the need to expense Year 2K expenditure (e.g., FASB, 1996; AASB, 1997; ASB, 1998; IASB, 1998b) or the introduction of the euro (IASB, 1998b), standard-setters have not produced rules dealing explicitly and exclusively with software expenditure.[2] Rather, the topic has usually only been addressed in passing in the course of discussions about whether certain activities could or could not be regarded as R&D (or, in the IASB's case, giving rise to intangible assets). Extracts from accounting standards referring to R&D activities concerned with *services or processes* are set forth in Table 1. It will be noted that various standards propose inconsistent treatments for items regarded as being encompassed by, or excluded from, the concept of R&D.

One implication of the capitalization rules in Australian, U.K. and international standards is that expenditure can be capitalized on the basis of expectations about future net benefits (and all projects are undertaken on the basis of such expectations) up until the time that an entity is forced to recognize that future returns are uncertain or the project was a failure. At that time, any asset previously recognized at increasing amounts must then be written off in its entirety.

Accounting for outlays on software development must also consider modifications. These may occur during the initial development, and the subsequent development (post production). During development, modifications arise from rectifying defects (von Mayrhauser, 1990), sometimes due to inadequate testing (Whitten, 1995), or

[2] While the U.K. ASB's UITF Abstract 20 (1998) dealt primarily with modifications for Year 2K, it also suggested that expenditure on modifications that enhanced service potential beyond that originally assessed 'would qualify for capitalization and depreciation' (para. 4).

TABLE 1

REFERENCES TO TREATMENT OF EXPENDITURE ON IMPROVED SERVICES OR PROCESSES IN R&D ACCOUNTING STANDARDS

Source	Included	Treatment if item included as R&D	Excluded	Treatment if item not R&D
U.S.A.'s FAS 2 (1974)	Modification of the formulation or design of a process	Expense	Adaptation of an existing capability to a particular requirement or customer's need as part of a continuing commercial activity	May capitalize
Australia's AAS 13 (1983)[a]	Formulation and design of possible new or improved process alternatives; evaluation of process alternatives	May capitalize	Adaptation of an existing capability to a particular requirement or customer's need as part of a continuing commercial activity	Expense
U.K.'s SSAP 13 (1989)	Design of services, processes or systems involving new technology or substantially improving those already produced or installed	May capitalize	Periodic alteration to existing services or processes even though they may represent some improvement	Expense
IASB's IAS 38 (1998a)	Research: the search for alternatives processes, systems or services; and the formulation, design, evaluation and final selection of possible alternatives for new or improved processes, systems or services	Expense	n.a.	n.a.
	Development: the design, construction and testing of a chosen alternative for new or improved processes, systems or services	Must be capitalized if recognition criteria are met		

[a] AASB 1011 (1987) used similar wording for inclusions, and identical wording for exclusions.

poor specification of requirements (Gibson, 1992). Once in production, further changes may be required, though the timing of further developments may be unpredictable (Sommerville, 2000). Changes in the software may also be sought as management seek different forms of analyses or reports, or to integrate software with other systems. These enhancements of functionality may be directed toward maximizing customer satisfaction or minimizing effort and schedule time (Grady, 1992). (See Appendix for a glossary for explanations of selected software development terminology.)

The foregoing indicates that there may be alternative methods for handling software expenditure in financial reporting. These methods are now considered in detail.

ALTERNATIVE METHODS FOR FINANCIAL REPORTING

The accompanying case study is a composite based on the authors' observations of around a dozen major IT projects associated with the provision of financial services in around a dozen different organizations. Following the approach of Drucker (1974), the facts are disguised to avoid identification of the firms involved. The case describes a software development project which experienced major difficulties, but which after considerable effort (and expense) may well have been regarded as technically successful.

The case study is used to illustrate the potential of current accounting practices to generate a wide range of financial data about software development projects, so that published information may mislead shareholders and external stakeholders (and possibly, managers and boards that rely on information presented to them).

The overall set of processes undertaken in a software development project is generally termed the 'system development cycle', and is described in terms of a sequence of activities. However, in practice, activities may be interspersed or repeated. For ease of presentation, Table 2 shows the activities in sequential order.

One key accounting issue concerns the starting point for tracing expenditure to be capitalized—from the start of scoping studies, or from the start of a project, or from when technological feasibility of the project was confirmed, or from when commercial viability of the project was established. In the case study, costs might have been capitalized up until the costing point (c3) shown in Table 2. Thereafter additional capitalization might have occurred at costing points (d), (e), (f), (g) and (h), with additional consideration of whether values were impaired at the end of each reporting period.

A more conservative approach to the capitalization of software development expenditure might seek to identify the costs incurred on new software that was attributable to defective *modules*. Only the AICPA's 1998 guidelines emphasize

Case Study

INTERNAL SOFTWARE DEVELOPMENT

(a)	Finserv Limited operates in the financial services industry, providing services to clients negotiated under period contracts. Finserv's information systems operations and software development are in-house activities. An in-house assessment found a poor fit between the existing system and identified stakeholder demands for new information and concluded they could not be met from the existing functionality. Hence Finserv decided to replace its existing in-house financial services software.

(b) In 2000 a scoping study was completed by an external consultant at a cost of $500,000. The option of 'buy a package and modify' was considered optimal. The board agreed a budget of $12 million for the project. This figure included an initial licence fee of $1.5 million and $500,000 per annum maintenance fee, both payable on signing the contract. A schedule of the project activities accompanying the budget indicated that the new software would be in production on 2 January 2002.

(c1) Software functionality modifications to meet the needs of the firm's clients were completed during 2001, in accordance with the project schedule, at a cost of $8.3 million. A range of programming defects were identified, and rectified prior to the

(d) software being put into production at no additional charge. Data conversion was completed at the cost of $100,000. This comprised checking of transactions and balances in the old system and uploading checked balances into the new system. A number of urgent enhancements were authorized when it was recognized that the new system required full transaction detail from legacy systems to allow correct

(c3) computations. This led to an additional, unbudgeted cost, including testing, of $600,000. Although minor problems were identified as fixes requiring action, none were 'day 1' problems. Despite the additional work, the new system was commissioned as scheduled on 2 January 2002, and the project was then under budget by $1 million.

(c2) Such were the extent of the modifications to the licensed software that $100,000 was paid to a technical documentation specialist contractor to prepare on-line manuals for operators. This was unbudgeted.

After the first end-of-month run (January 2002), 'day 2' problems emerged.

(e) As a consequence, two transaction-specific modules were re-written and re-tested before eventually being put into production on 30 May, at a cost of $1.5 million.

(f1) In addition, both modules required new computation and reporting subsystems as well as additional (and expensive) output testing of the entire system at an additional cost of $750,000. Some records which were in error remained quarantined because the new programs failed to process them correctly.

(f2) Additional programming costing $500,000 was required to handle the exceptional conditions required by the quarantined accounts. Although this work was completed by the reporting deadline of 30 June, account balances were unable to be reconciled by finance staff. At nights and weekends during July checks were made of all records to ensure accuracy (cost $50,000) and each of the

(g) modules for valid computations (cost $50,000). This confirmed further computation errors were still present in the re-written modules but also revealed that some incorrect record balances had been transferred unchecked to the new system.

On 30 June 2002 approval was given for Finserv to enter into a fixed price contract for $4 million over two years to resolve outstanding enhancement requests.

(h) Finance staff calculated that $500,000 in staff time had been spent in reconciling account balances. They estimated an additional $500,000 would be spent in correcting errors now incorporated in individual record balances that had been quarantined.

(i) In March 2003 the firm's major clients required additional modifications to the software to accommodate changes in legislation and to offer a wider range of choices to the client's customers. These modifications would necessitate total rewriting of one module of the software, at a cost of $1 million. At the same time, the software vendor indicated that it had commenced work on its new generation finance suite that it expected would be released in around eighteen months, and gave notice that the existing software package would no longer be supported after two years.

the need to focus on modules or components of software—other standards are silent in this issue. Indeed, other standards that permit some form of capitalization of software are silent on precisely what sub-projects associated with the development of internal-use software or with systems migrations and subsequent maintenance could be capitalized. Standards also provide some conflicting guidance on what categories of costs should be traced to those projects.

However, the figures listed up to costing point (e) do not represent the values which might be assigned to assets in the firm's statement of financial position, or indicate the full amount of expenses attributable to software development in the year ended 30 June 2002. That is because elements of capitalized expenditure may have been depreciated (or amortized) from different points of time during the course of the software development. Starting points might have been 2 January 2002 (when the system was initially commissioned), or 30 May 2002 (when modules were rewritten), or even 30 June 2002 (when most errors arising from the migration were identified). If the system had been migrated with a parallel run of the old system, up to (say) 30 June 2002, then the date on which the old system was turned off could also be regarded as the starting date for amortization of the new system. In conjunction with these accounting choices, depreciation on incremental expenditure might also start from the date other re-worked modules were put into production—a process which could stretch over several years.

Judgments would also have to be made about the economic life of the new software. Some might argue that a three-year period was the maximum appropriate for amortization of most software, given the pace of technological change. In fact one authority has commented that the useful life of software is likely to be very short (see, IAS 38, para. 81). Practitioners might link the amortization period to the term of contracts that the firm had with its clients (say, five years). Other practitioners might argue that the system was likely to be suitable for new clients and that the economic life could extend for an even longer period. However, another factor affecting such judgments could be the likely length of time that software vendors would support the current version of their software, and the extent to which new releases would be compatible with the customized modifications undertaken in house.

An important part of internally developed software is the preparation of documentation. This is usually made available to users online (Jenkins and Wallace, 2002). The cost of preparing the documentation and converting it to an online format (e.g., HTML or XML) could be regarded as a project cost. However, such manuals might also be regarded as part of the cost of training operators, and both AICPA (1998) and IAS 38 (1998a) state that training costs should be expensed.

Further issues concern the treatment of licence fees and maintenance contracts. The firm in the case study bought rights to use a commercial package which it then modified to suit its own needs. The up-front licence fee was a significant sum ($1.5 million) and conferred indefinite rights to use the commercial package. In association with the licence, the firm was required to enter into a $500,000 per annum maintenance contract.

TABLE 2

CAPITALIZATION OPTIONS FOR INTERNAL SOFTWARE DEVELOPMENT

Panel A: During development

Ref	Costing point in system development lifecycle (activity)	Business benefit (per case study)	Cost (per case study)	Accounting options	Relevant rules or guidelines; commentary
(a)	Preliminary assessment	Confirmation existing systems inadequate	No incremental cost	Capitalize costs, or expense	AICPA SoP (1998) provides that 'computer software costs that are incurred in the preliminary project stage should be expensed as incurred'
					IAS 38 (1998a) provides that expenditure on the research phase of an internal project should be recognised as an expense when it is incurred. However definition of 'research' is ambiguous
					AAS 13 (1983) and AASB 1011 (1987) allow expenditure during preliminary project stages to be capitalized until determinations made about recoverability
(b)		Budget for project and assessment of projected return on investment	$0.50 m (budgeted)	Capitalize costs, or expense	Capitalization of scoping studies that proceed to implementation appears acceptable in terms of Australian standards AAS 13 (1983)/AASB 1011 (1987)
					AICPA SoP (1998) permits capitalization after preliminary project stage, provided project is formally authorized and funding committed
					IAS 38 (1998a) permits capitalization during development stage, provided evidence of intention and capacity to complete the project, existence of economic benefits, and meets tests of asset recognition
					SSAP 13 (1989) states that it is permissible to defer development expenditure to the extent that its recovery can reasonably regarded as assured

TABLE 2

(CONTINUED)

Panel A: During development

Ref	Costing point in system development lifecycle (activity)	Business benefit (per case study)	Cost (per case study)	Accounting options	Relevant rules or guidelines; commentary
(c1)		Required functionality completed; additional urgent enhancements finalised	$8.40 m ($1.6 m under budget)	Capitalize costs, or expense (until technical feasibility or commercial viability established)	AICPA SoP (1998) provides that internal or external costs incurred for upgrades and enhancements should be capitalized in the application development stage SSAP 13 (1989) permits capitalization once *both* technical feasibility and commercial viability established
				Capitalize or expense interest	AICPA SoP (1998) prescribes capitalization of interest costs AASB 1011 (1987) makes no specific reference to interest, but allows capitalization of 'costs that can be attributed to R&D and identified with specific projects IAS 23 (1994) allows capitalization of directly attributable borrowing costs
				Capitalize or expense overheads	FAS 2 (1974) states that R&D costs shall include a reasonable allocation of indirect costs However AICPA SoP (1998) states that general and administrative costs and overhead costs should not be capitalized as cost of internal-use software IAS 38 (1998a) states that cost of intangible assets shall include allocations of indirect costs that are necessary to generate the asset and that can be allocated on a reasonable and consistent basis AAS 13 (1983) allows capitalization of costs that can be directly attributed to R&D

Panel A: During development

Ref	Costing point in system development lifecycle (activity)	Business benefit (per case study)	Cost (per case study)	Accounting options	Relevant rules or guidelines; commentary
(c2)	Systems analysis	Operator assistance	$0.10 m (not in budget)	Capitalize or expense (if capitalized, choice of amortization method and period)	AICPA SoP (1998) and IAS 38 (1998a) prohibit capitalization of expenditure on 'training' However, on-line manuals could be regarded as 'software'
(c3)	Development (design; programming; testing)	Readiness for production	$0.60 m ($1 m under budget)	Capitalize or expense	AICPA SoP (1998) states that guidance re capitalization/ expense should be applied to individual components or modules of software—suggesting costs of defective software should be expensed
(d)	Migration (data conversion from old to new system)	Migrate to new production system	$0.10 m (under budget).	Capitalize or expense	AICPA SoP (1998) states that costs incurred in data conversion should be expensed, while costs to develop, or obtain, software that allows for access or conversion of old data by new systems should be capitalized. Other standards are silent on treatment of this expenditure

TABLE 2

(CONTINUED)

Panel B: During production

Ref	Costing point in system development lifecycle (activity)	Business benefit (per case study)	Cost (per case study)	Accounting options	Relevant rules or guidelines
(e)	Defect fixes	Operational stability with routine transactions	$1.50 m	Write-off costs of defective module, and capitalize costs of recoded module; or expense	AICPA SoP (1998) states that guidance re capitalization/ expense should be applied to individual components or modules of software—suggesting costs of defective software should be expensed IAS 38 (1998a) only allows capitalization of subsequent expenditure if this will generate economic benefits in excess of originally assessed standard of performance
(f1)	Post-production data quality improvement	Record checking and computation validation	$0.75 m	Capitalize or expense	AICPA SoP (1998) states that data conversion costs include 'purging or cleaning of existing data, reconciliation or balancing of the old data and the data in the new systems, creation of new/additional data, and conversion of old data to the new system'—and all are to be expensed Other standards are silent on treatment of this expenditure
(f2)		Manual reconciliation of records by finance staff	$0.50 m	Capitalize or expense	AICPA SoP (1998) suggests that costs incurred in data conversion should be expensed Other standards are silent on treatment of this expenditure
(f3)		Record validation	$0.05 m	Capitalize or expense	AICPA SoP (1998) suggests that costs incurred in data conversion should be expensed Other standards are silent on treatment of this expenditure
(g)	Defect fixes	Retest entire database	$0.05 m	Capitalize, or expense	AICPA SoP (1998) suggests that costs incurred in data conversion should be expensed Other standards are silent on treatment of this expenditure
(h)	Enhancements	Outstanding change requests to be delivered over two-year period	$4.00 m	Capitalize, or expense	AICPA SoP (1998) states that expenditure on upgrades and enhancements should be capitalized if it is probable that those expenditures will result in additional functionality IAS 38 (1998a) allows capitalization of expenditure that will generate economic benefits in excess of originally assessed standard of performance Other standards are silent on treatment of this expenditure

Panel C: End of reporting periods

Ref	Accounting options	Relevant rules or guidelines; commentary
(i)	Assess impaired value (recoverable amount) by reference to present value of projected cash flows—discounted or undiscounted	AICPA (1998) and FAS 121 (1995) apply impairment test to undiscounted future cash flows IAS 38 (1998a) requires development costs to be capitalized where recognition criteria are met
		SSAP 13 (1989) requires capitalized expenditure to be written off to the extent to which it is irrecoverable, on a project by project basis
		AAS 13 (1983) and AASB 1011 (1987) require write-downs to the extent that carrying value exceeds recoverable amount
		AICPA (1998) applies to individual components or modules of software
	From migration; end of parallel run, production; over expected economic life, life of contracts, etc.	AAS 13 (1983) and AASB 1011 (1987) state that amortization commences with commercial production, and should match costs with related benefits
		IAS 38 (1998a) states that amortization commences when asset is available for use (and it is likely that useful life of software will be short)
	Amortization method—to be chosen	Method should reflect the pattern in which the asset's economic benefits are consumed by the enterprise, but if that cannot be determined reliably, the straight line method should be used
		SSAP 13 (1989) requires amortization over period of expected use of system. AICPA SoP (1998) states that straight line method should be used unless another systematic and rational basis is more representative of software use

The cost of the original licence fee might be expensed in the year it was purchased, or when the software was first put into production; or the licence fees could be capitalized and then amortized over the period of contracts with existing clients, or other estimates of economic life. As for the treatment of expenditure on maintenance, it has been suggested that maintenance consumes 40–80 per cent of the software lifecycle costs (von Mayrhauser, 1990). Experience suggests that the accounting treatments adopted by many firms treat 'maintenance' the same as expenditure incurred on servicing machinery or buildings, that is as an expense (and such an approach has been advocated in the accounting literature—see Ameen and Noll, 1997). However, the terminology used in the IT industry is ambiguous (see 'Maintenance' in the Appendix). While it is preferable to distinguish 'maintenance' from 'enhancements', many authors use 'maintenance' as an umbrella term to refer to changes which retain or restore functionality, or provide enhancements (see, e.g., Boddie, 1987; Marciniak and Reifer, 1990; Youll, 1990). The term 'maintenance' is also used to refer to software changes made by a vendor and supplied under licence renewal contract agreements. In this case the changes may include both rectification of faults, and improvements. For example, the vendor may be contractually obliged to provide business analysts and programmers onsite, working a specified number of hours per annum, to handle modification requests from the client. These requests may relate to the rectification of inefficient or incorrect code, but may also encompass minor or even major enhancements—either during the development phase, or after software is put into production. It is common for additional, unplanned development to occur after the software is put into production.

If the licensee has elected to capitalize software development expenditure, then a further option arises: Should the licensee expense all maintenance expenditure, or capitalize that part of it which could be regarded as expenditure on enhancements? While the AICPA has advocated the expensing of internal maintenance expenditure and expenditure on external maintenance contracts that constitute enhancements (AICPA, 1998, para. 26), that position only relates to enhancements that were not *pre-specified* in maintenance contracts (implying that careful drafting of those contracts could legitimise capitalization). Similarly, Australian and International Accounting Standards on R&D or intangibles could be regarded as supportive of the capitalization of *all* expenditure on enhancements, provided that this expenditure was expected to be recoverable or to produce future economic benefits (AASB 1011, para. 20; IAS 38, paras 53–4).

But that is before considering the application of the 'recoverable amount test' (RAT) which is now sometimes termed the 'asset impairment test' (AIT), whereby consideration is given to whether asset balances at the end of the financial year exceed the value of cash flows expected to be obtained from use and ultimate resale of an asset. But the RAT or AIT test can be variously applied in this situation. It could be applied to elements of internally developed software that are used to provide specific services or financial products (a treatment advocated by the AICPA, 1998) or it could be applied to the overall package. Moreover, it could be argued that the RAT or AIT test should be applied to either the net cash

flows directly attributable to servicing client contracts, or to the net cash flows after deduction of some proportion of centralized expenditure. In principle, the RAT or AIT test could be applied to either discounted or undiscounted projected cash flows (though again, the AICPA 1998 guidelines propose use of undiscounted data).

To borrow from Chambers' (1965) classic analysis of options for recording inventories or calculating depreciation, there are in this simple example ($2 \times 4 \times 3 \times 4 \times 2 \times 4 \times 4 \times 2 \times 2 \times 2 \times 3 \times 2 \times 2 \times 2 \times 2 \times 3 \times 3$) = 10,616,832 possible treatments within the context of the 'written down historical cost' approach to asset valuation embodied in Australian accounting standards. Table 3 presents the detail.

In terms of the case study, the availability of these options could have a major impact on reported operating results and financial position and hence on satisfying any comparability criteria either over time or between firms. For example, for the reporting period ended 31 December 2001, reported software expenses could have ranged from zero to $11.5 million; correspondingly, capitalized software could have been recorded at between $11.5 million and zero.

An additional three options would be available if funds expended on software expenditure were to be reimbursed. Firms operating in the financial services industry may enter into contractual arrangements whereby clients meet the costs of modifying software to accommodate legislation or rule changes. Based on accounting precedents (and the invocation of notions of matching) such reimbursements could be regarded either as current year revenues, or as reserves to be amortized against future years on the same basis as the corresponding expenditure is being amortized, or as receipts to be offset against the sums of software development costs being capitalized. Only the AICPA Statement of Position (1998) has directly addressed this issue, advocating the last mentioned treatment. Australian standard AASB 1011 suggests that 'government or other grants' should be deducted from the carrying amount of capitalized software, though it is doubtful whether reimbursements made in terms of contractual arrangements could properly be described as 'grants'. These three additional options would bring the revised total number of possible accounting treatments (in terms of Australian accounting standards) to no less a number than 31,850,496.

Readers might care to undertake their own calculations of the range of options available in terms of U.K. and International Accounting Standards, based on the summary presented in Table 2, but (as noted above) any such calculations depend on how concepts such as 'technological feasibility', 'commercial viability' or 'enhancements' are interpreted. It is suggested that even the most conservative of interpretations would still leave a very large number of options. Arguably the AICPA's Statement of Position (1998) provides the most detailed guidance for the treatment of expenditure on different phases of the development of internal-use software. It is considered that AICPA SoP (1998) unambiguously identifies a specific treatment for seven of the seventeen accounting issues listed in Table 3, reducing the number of options to an estimated 82,944.

To those who argue that a role of accounting regulation is to reduce the number of alternative practices, the elimination of more than 31 million options may seem to represent success. But if the most detailed accounting guidelines still

TABLE 3

OPTIONS IN TERMS OF AUSTRALIAN ACCOUNTING STANDARDS FOR THE TREATMENT OF INTERNAL SOFTWARE DEVELOPMENT

Accounting issues	Possible treatments	Commentary
Treatment of preliminary assessment	2	Capitalize, or expense
Starting point for tracing expenditure to be capitalized	4	From start of scoping studies; or from start of project (subject to later review); or from when technological feasibility established; or from when commercial viability established (sufficient for asset recognition)
Threshold for capitalizing expenditure	3	Suppose a choice of 'low', 'medium' and 'high' capitalization thresholds
Costs to be traced and capitalized	4	Direct costs only; or direct costs plus interest; or direct costs plus allocation of overheads; or direct costs plus interest plus allocation of overheads
Licence fees paid in advance	2	Capitalize or expense
Select starting date for amortization	4	When software successfully tested; when systems migrated; or when system placed in production; or when defects corrected so system 'fit for use'
Identify 'economic lives' for amortization purposes	4	Could be arbitrary estimate of functional economic life; or remaining life of existing contracts with clients; or estimate of life that system would be *competitive*; or remaining period that underlying software will be supported by vendor
Pattern of amortization	2	Straight line or alternative method
Handle original expenditure on defective modules	2	Retain capitalized value, or expense
Treatment of costs incurred on data conversion	2	Capitalize or expense
Subsequent expenditure on defective modules	3	Capitalize; or capitalize only if project still within budget; or expense[a]
Expenditure on sub-systems, initial output testing	2	Capitalize or expense
Treatment of expenditure on data reconciliations and correcting data in quarantined accounts (finance staff)	2	Capitalize (treat as cost of data conversion, and hence IT expense) or treat as expense of finance department
Treatment of expenditure on on-line manuals	2	Expense if regarded as 'training'; or capitalize if regarded as part of software cost
Treatment of expenditure to resolve change requests	2	Capitalize, or expense
Treatment of expenditure on modifications in subsequent years	3	Capitalize a proportion attributable to enhancements; capitalize proportion, subject to materiality test; or expense
Treatment of expenditure on rule changes	3	Capitalize; capitalize after writing off written down balance of existing modules; or expense current expenditure

[a] Arguably expenditure on re-writing defective modules would be expensed if the original expenditure on those modules was still capitalized. It might also be argued that capitalization should be applied to whatever component was the least cost. However for illustrative purposes, decisions about the treatment of these two items are regarded as distinct.

permit scores of treatments—let alone many thousands—then that suggests that there are deficiencies in the drafting of those regulations (or in the conceptual framework underpinning those accounting rules—discussed further below).

Note that the above figures relate to the treatment for a *single* project. Most organizations would have more than a single software development project active at any point in time. Different judgments may be made for each of these regarding the starting or finishing dates of capitalization practices, and projected economic lives—adding to the complexity faced by readers of financial statements in trying to interpret the impact of accounting choices in this area.

Of all of these various alternatives, one treatment is the simplest to apply and interpret: not regarding *any* expenditure on internal-use software as an asset, but immediately recording every outlay on licences, software development, data checking and 'maintenance' as an expense.

WHAT EXPENDITURE SHOULD BE CAPITALIZED OR EXPENSED?

The following discussion of the accounting treatment of software development is confined to the circumstances of firms which build their own packages, or adapt commercial packages, for their own use. It does not address the circumstances of software development companies, whose accounting practices are likely to be based on the rules contained in Australian or International Accounting Standards for research and development; or on the U.S.A.'s SFAS 86 (1985a) which has proved contentious because of its support for capitalization (see Aboody and Lev, 1998).

As already noted, accounting standards and authoritative statements reflect differing views about what is the appropriate accounting treatment of expenditure on internal-use software. In part, this reflects differing views about what principles or key concepts underlie the practice of accounting.

One view (which was dominant until the late 1970s and reflected in academic and professional literatures) was that accounting seeks to 'match costs and revenues'. On this basis, expenditure on software that was expected to produce future benefits need not be regarded as an expense in the year in which it is incurred, but could be allocated to those accounting periods in which economic benefits are expected to be derived. In terms of this rationale (possibly expressed in its most extreme form by Littleton, 1953, or by the AICPA, 1957), assets merely represent costs which have yet to be allocated to future periods. The 'matching' rationale has been invoked to support capitalization (Paulsen, 1983; AASB 1011, 1987; SSAP 13, 1989) or to support immediate expensing (e.g., Gannon and Parkinson, 1983).

A second view (expressed most systematically by Chambers, 1966, and subsequently adopted in documents issued by the accounting profession and other bodies [FASB, 1980, 1985b; ASRB, 1985; AARF, 1992]), is that income statements and statements of financial position are articulated, so that the identification and quantification of revenues and expenses is necessarily derived from successive statements of financial position. Restated, any outlays which do not give rise to the acquisition of assets (and which are not distributions to equity holders) are to be

treated as expenses. This 'balance sheet viewpoint' changes the focus of accounting practice from considering whether outlays are likely to give rise to future benefits, to whether they have led to the acquisition of recognizable assets. It replaces non-operational allusions to 'matching costs and revenues' with an approach based on the identification and valuation of assets and liabilities. Much, therefore, turns on the definition of asset, on associated criteria for asset recognition, and on the basis for asset valuation.

The accounting profession's definitions and recognition criteria for assets remain contentious.[3] The Australian accounting profession's SAC 4 (1992) states that '"assets" are future economic benefits controlled by the entity as result of past transactions or other past events', and specifies that 'an asset should be recognised in the statement of financial position when and only when: (a) it is probable that the future economic benefits embodied in the asset will eventuate; and (b) the asset possesses a cost or other value that can be measured reliably' (paras 14, 38). The SAC 4 tests are similar to the definition and tests for asset recognition adopted earlier by the U.S.A.'s FASB (FASB 1980, 1984, 1985b). Neither definition attaches any significance to whether an entity has any legal right to supposed *economic benefits*. Rather, the definitions emphasize the looser notion of *control*. SAC 4 states: ' "control of an asset" means the capacity of the entity to benefit from the asset in the pursuit of the entity's objectives and to deny or regulate the access of others to that benefit' (para. 14). Further, the profession-sponsored definitions do not have regard to whether or not an item is saleable—a matter which many have argued as critical if financial information is to be relevant to a range of judgments faced by potential users of published financial reports.

Against that background, one may consider whether expenditure on software development can be regarded as an asset in terms of these definitions and associated tests.

The prevailing model of software possession is via a licence, as the majority of software is purchased in a ready-to-run state (shrinkwrap). Software which is custom developed for one user or organization (bespoke) usually entitles the commissioning individual or organization to ownership. A third arrangement is that the vendor owns the primary software, but licences it to a user firm, which then develops a software application using it; the resulting software has no utility unless used in association with the licensed package.

In the third option above, it is debateable whether the user could be said to *control* the benefits expected to arise from use of the modified software. Initially, the complexities of control arise from the combination of software development source, funding and contractual arrangements.

Where in-house development is funded by the firm, full control exists if copyright is held by the end user organization. Some control exists where contractual arrangements provide for rights to the software to revert to the customer after a

[3] Contributors to a forum on the conceptual framework in a recent issue of this journal reviewed shortcomings in the design and application of the Australian profession's Statements of Accounting Concepts. See *Abacus*, October 2003.

nominated period, often through a balloon payment. No control exists if there is partial funding of the vendor by licensee. While it is common for the licensee to negotiate contractual arrangements for software to be placed in escrow when development is complete, the licensee typically cannot obtain control over that software unless contractual conditions concerning business failure of the vendor are met. Until that point the value of the software in escrow may be negligible. After that point the value may still be zero, owing to delays in availability before maintenance responsibility can be assumed, or the costs of undertaking maintenance on inadequately documented source code.

Even where the licensee supposedly has legal rights to exclusive use of licensee-funded modifications to licensed software, it is debateable whether the licensee can be said to control use of that material because it can only be used in conjunction with the vendor's software, which in turn is only available under a licence (subject to termination clauses). Commercial realities often dictate that modifications made in these circumstances usually appear in the vendor's next release of that software. As the non-exclusive licensee, the sponsor of the functionality requirements is in a position to benefit from the modifications, in its own business, but may not be able to restrict access to those benefits *vis à vis* the licensor or other licensees. Indeed, to avoid onerous future maintenance costs associated with maintaining compatibility, the sponsor may negotiate with the vendor to absorb them into the next edition of the base product. Given that decisions to incur expenditure on enhancing IT systems would be undertaken in the optimistic expectation that this expenditure will later produce future benefits, then (on one reading of accounting literature) virtually all outlays on software modifications could be regarded as giving rise to new assets. Since expenditure incurred in the development of new systems can be identified or costed in one way or another, some may claim that this would satisfy the accounting profession's threshold requirement for asset recognition that an item can be 'reliably measured'. However, as has been noted previously (Walker and Jones, 2003) the availability of so many options for calculating 'cost' highlights the manner in which such accounting references to the concept of reliability conflict with the use of that term in other disciplines, whereby measures are regarded as reliable if they can be independently replicated by other observers (see, e.g., Nunnally, 1978, p. 191). Indeed, the availability of such a wide range of alternative accounting treatments suggests that many of the numbers that could potentially be reported may convey a misleading impression of profitability and financial performance—they cannot all be right.

Since expenditure on software commonly does not confer rights which are saleable, the reporting of capitalized software as an asset does not convey relevant information to present or potential investors who are interested in the security of their investment in terms of net asset backing, or the risk of insolvency. Correspondingly it does not convey information about the adequacy of security or capacity to repay debt to lenders and creditors, or to employees concerned with the capacity of a firm to meet employee entitlements.

Accordingly, it is argued that expenditure on software development should not be recorded as an asset. Rather, such expenditure—be it on licence fees,

customization or maintenance contracts—should be fully expensed at the time it is incurred. The only exception may be in relation to prepayments of maintenance contracts where refunds might be payable on cancellation.

It might be argued that capitalization is acceptable, since full disclosure of accounting policies would enable readers to make their own interpretations of firm performance and financial position. Yet in terms of existing reporting arrangements, analysts may not learn that a reporting entity had been overspending on a troubled IT project (and capitalizing that expenditure) if a statement of accounting policies merely reported that 'costs of software development are capitalized until individual projects are completed and then the capitalized costs are amortized over the project's expected economic life'.

Even if statements of accounting policies identified what dollar values had been capitalized, disclosure of those sums will not in itself indicate if those sums had been spent efficiently or effectively.

It has been suggested that 'software capitalization can be easily undone by subtracting the periodic capitalization figure from reported earnings and the capitalized software asset from total assets and equity' (Aboody and Lev, 1998). However, in some countries firms may report capitalized software in combination with computer hardware or more generally as 'plant and equipment' (particularly if a turnkey solution was purchased). Hence it may not be possible to identify what software costs had been capitalized during a year, or what software costs had been capitalized, in aggregate.

Moreover, given that firms may both capitalize new expenditure on software, and write down elements of previously capitalized expenditure, it may not be possible to identify what costs had been capitalized in a particular period without compiling and analysing information reported in various notes to the financial statements for several periods (assuming they are readily available). In the sets of national or international accounting standards or guidelines reviewed, a common requirement was for disclosure of the amount of 'development' expenditure charged against profits during a reporting period (FAS 2, 1974; SSAP 13, 1989—with AASB 1011, 1987, requiring disclosure of those sums 'before crediting any related grants'). However, only U.S. standard FAS 2 requires direct disclosure of the total R&D incurred in each period and of the amount capitalized in each period. SSAP 13 requires disclosure of movements in deferred expenditure during the year, together with the amount of capitalized expenditure carried forward at the beginning and end of the reporting period. AASB 1011 only requires disclosure of the amount of deferred R&D costs at the end of the financial year, with accumulated amortization charges being shown separately as a deduction. If comparative figures were available, it would be possible to work out what had been capitalized during a reporting period—provided published balance sheet data were not affected by write-downs due to 'impairment'. Both FAS 121 and AASB 1010 (1999) require disclosure of amounts written off assets during the reporting period because of 'impairment' (or reductions in 'recoverable amount')—but that may be reported in separate notes.

Further difficulties would arise when endeavouring to extrapolate the impact of these practices on future reported earnings. A common requirement in accounting

standards is for disclosure of accounting policies, and both FAS 2 and AASB 1011 specifically require disclosure of the basis of amortization for R&D. However, there is no requirement for that information to be disaggregated by project. The exercise of attempting to assess the impact on future earnings of capitalization practices would be challenging enough for firms engaged in single software development projects, but would be more difficult when firms are engaged in multiple projects, each with its own rationale for development.

It does not seem surprising that some analysts have opposed capitalization (see Gannon and Parkinson, 1983), particularly since adoption of the full expensing option would remove one method of 'managing earnings', and ensure some degree of consistency in the way in which firms—particularly those in the financial services industry, which are heavily involved in software development—calculate and report their profits and financial position.

It might also be argued that present or potential investors (and other stakeholders) may find information about the extent of investment in software development of relevance when considering the prospective profitability of firms. There is an expectation that applications will be scalable and flexible (van der Zee, 2002). However, information about the timing and extent of expenditure on upgrading or developing software could be provided in a more accessible form than from the mix of disclosures required by contemporary accounting standards.

There is also a case for the disclosure of such expenditure as a line item in statements of financial performance and statements of cash flows. Notes to the financial statements could detail the nature and purpose of this expenditure, and describe the period over which that expenditure is expected to confer economic benefits. In relation to major projects, notes could provide some indication of the budgets and expected completion dates of that work.

It might be argued that more extensive disclosure about the nature and purpose of developmental expenditure would compromise competitive advantage. However, staff mobility in specialized industries usually means that competitors have a good idea what other firms are doing. Much information about major software development projects is publicly available in computing technology trade magazines and newsletters. Non-disclosure of information about material software development projects would usually work to the disadvantage of external stakeholders who are primarily reliant on published financial information.

It might also be argued that requiring expensing would have adverse economic consequences on stakeholders in firms making those investments. Yet if software projects are successful and produce significant economic benefits, that should soon become evident, regardless of whether expenditure is capitalized or expensed—so that any 'good news' should not be viewed as a surprise by stakeholders because this forecast is precisely the reason why the development project ought to have been authorized. On the other hand, the writing off of previously capitalized expenditure on projects that later failed could be regarded as a surprise, since prior audited financial statements would have represented that, on the balance of probabilities, this expenditure would produce economic benefits. It is tempting to observe that the adverse consequences on readers of financial statements from a

misclassification are likely to be greater if firms capitalize when they should have expensed, rather than vice versa. But such an observation assumes that expenditure on software might be properly regarded and recognized as an asset when (as argued above) such an item does not meet the tests of 'control' and 'reliable measurement'.

CONCLUDING COMMENTS

In summary, the recommended approach is: (a) the immediate expensing of internally developed software; (b) reporting of this expense as a line item where software expenditure is material; and (c) disclosing, in notes to the financial statements, information about major software development projects.

Contemporary accounting standards require a series of subjective judgments to be made about such matters as technological feasibility, commercial viability, economic life, and whether modifications constitute enhancements or changes in functionality. This leads to an extensive series of choices about accounting treatments. The range of accounting options described in the hypothetical case study illustrate the difficulties facing analysts seeking to understand and interpret published data. These difficulties would be compounded when firms are engaged in multiple projects spanning multiple accounting periods.

The immediate expensing of software development expenditure is likely to ensure that the financial statements of firms engaged in those activities are more comparable than if they could choose from a multitude of options. It is also likely to make the financial statements more readily interpretable. This article has not considered the adequacy of arrangements for internal reporting of individual IT projects to senior management or boards, though it seems likely that the practice of capitalizing software expenditure could affect the quality of information provided to decision makers concerning project arrangements, and disguise shortcomings in the productivity of IT departments. These concerns are to be addressed in a subsequent article.

The abandonment of capitalization may mean that items previously identified in a cash flow statement as expenditure from investing will now disappear. Accordingly, in cases where such expenditure is material, there is a case for treating expenditure on software development as a 'line item' in a statement of operating performance, and for expanded disclosures in notes to the financial statements. Such notes could detail the nature and purpose of this expenditure, and describe the period over which that expenditure is expected to confer economic benefits. In relation to major projects, notes could also provide some indication of the budgets and expected completion dates of that work.

REFERENCES

Aboody, D., and B. Lev, 'The Value-Relevance of Intangibles: The Case of Software Capitalization', *Journal of Accounting Research* (Supplement), Vol. 36, 1998.

Accounting Standards Board (U.K.), SSAP 13, *Accounting for Research and Development*, issued 1977, revised 1989, amended 1997, 1998.

——, UITF Abstract 20, 'Year 2000 Issues:—Accounting and Disclosure', March 1998.

Accounting Standards Review Board (Australia), ASRB Release 100, *Criteria for the Evaluation of Accounting Standards*, 1985.

Ameen, P. D., and D. J. Noll, 'Internal-Use Computer Software: The Fixed Asset of the Information Age', *Journal of Accountancy*, March 1997.

American Institute of Certified Public Accountants (AICPA), Accounting Terminology Bulletin No. 4, *Cost, Expense and Loss*, 1957.

——, Statement of Position 98-1 (SoP 98-1), *Accounting for the Costs of Computer Software Developed or Obtained for Internal Use*, 1998.

Australian Accounting Research Foundation (AARF), Australian Accounting Standard AAS 13, *Accounting for Research and Development Costs*, 1983.

——, Statement of Accounting Concepts 4, *Definition and Recognition of the Elements of Financial Statements*, March 1992.

Australian Accounting Standards Board (AASB), AASB 1011, *Accounting for Research and Development Costs*, 1987.

——, UIG 12, 'Accounting for the Costs of Modifying Computer Software for the Year 2000', 1997.

——, AASB 1010, *Recoverable Amount of Non-Current Assets*, 1999.

Berger, P., 'Selecting Enterprise-Level Measures of IT Value', in P. Berger, J. G. Kobielus and D. E. Sutherland (eds), *Measuring Business Value of Information Technologies: Useful, Innovative Approaches for Management*, International Centre for Information Technologies, ICIT Press, 1988.

Boddie, J., *Crunch Mode: Building Effective Systems on a Tight Schedule*, Prentice Hall, 1987.

Burns, G. W., and D. S. Peterson, 'Accounting for Computer Software', *Journal of Accountancy*, April 1982.

Buttrick, R., *The Project Workout: A Toolkit for Reaping the Rewards From all Your Business Projects*, 2nd ed., Financial Times/Prentice Hall, 2000.

Chambers, R. J., 'Financial Information and the Securities Market', *Abacus*, September 1965.

——, *Accounting, Evaluation and Economic Behavior*, Prentice Hall, 1966.

Dempsey, G., 'Reusability and Accounting for Software Development Costs', *Accountability and Performance*, Vol. 3, No. 2, 1997.

Drucker, P., *Management: Tasks, Responsibilities*, *Practices*, Harper and Row, 1974.

Financial Accounting Standards Board (FASB), Statement of Financial Accounting Standards No. 2, *Accounting for Research and Development Costs*, October 1974.

——, FASB Interpretation No. 6, *Applicability of FASB Statement No. 2 to Computer Software*, February 1975.

——, FASB Technical Bulletin 79–2, *Computer Software Costs*, December 1979.

——, Statement of Financial Accounting Concepts No. 3, *Elements of Financial Statements of Business Enterprises*, 1980.

——, Statement of Financial Accounting Concepts No. 5, *Recognition and Measurement in Financial Statements of Business Enterprises*, 1984.

——, Statement of Financial Accounting Standards No. 86, *Accounting for the Costs of Computer Software to be Sold, Leased or Otherwise Marketed*, October 1985a.

——, Statement of Financial Accounting Concepts No. 6, *Elements of Financial Statements*, December 1985b.

——, Statement of Financial Accounting Standards No. 121, *Accounting for the Impairment of Long-Lived Assets and Long-Lived Assets to Be Disposed Of*, 1995.

——, EITF Abstract 96-14, *Accounting for Costs Associated with Modifying Computer Software for the Year 2000*, 1996.

Gannon, J. J., and D. Parkinson, 'Software Development Costs Should Be Expensed', *Management Accounting*, November 1983.

Gibson, R., *Managing Computer Projects: Avoiding the Pitfalls*, Prentice Hall, 1992.

Grady, R. B., *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, 1992.

Healy, P., 'The Effect of Bonus Schemes on Accounting Decisions', *Journal of Accounting and Economics*, April 1985.

Healy, P. M., and J. M. Wahlen, 'A Review of the Earnings Management Literature and its Implications for Standard Setting', *Accounting Horizons*, December 1999.

Holthausen, R., D. Larcker and R. Sloan, 'Annual Bonus Schemes and the Manipulation of Earnings', *Journal of Accounting and Economics*, Vol. 19, 1995.

International Accounting Standards Board (IASB), IAS 23, *Borrowing Costs*, 1994.

——, IAS 38, *Intangible Assets*, 1998a.

——, SIC 6, *Costs of Modifying Existing Software*, 1998b.

Jenkins, G., and M. Wallace, *IT Policies and Procedures: Tools and Techniques That Work*, Prentice Hall, 2002.

Jones, C., *Estimating Software Costs*, McGraw-Hill, 1998.

Littleton, A. C., *Structure of Accounting Theory*, American Accounting Association, 1953.

Marciniak, J. J., and D. J. Reifer, *Software Acquisition Management: Managing the Acquisition of Custom Software Systems*, John Wiley, 1990.

Myers, G. J., *The Art of Testing*, John Wiley, 1979.

Nunnally, J. C., *Psychometric Theory*, McGraw-Hill, 1978.

Paulsen, N. E., 'Software Development Costs Should be Capitalized', *Management Accounting*, November 1983.

Public Sector Accounting Standards Board of the Australian Accounting Research Foundation, and the Australian Accounting Standards Board, Statement of Accounting Concepts 4, *Definition and Recognition of the Elements of Financial Statements*, March 1995.

Securities and Exchange Commission, Securities Act Release No. 6461, *Accounting for Internal Costs of Developing Computer Software for Sale or Lease to Others*, SEC, 1983.

Sommerville, I., *Software Engineering*, Addison-Wesley, 6th ed., 2000.

Sullivan, P. H., Jr., and P. H. Sullivan, Sr., 'Valuing Intangibles Companies', *Journal of Intellectual Capital*, Vol. 1, No. 4, 2000.

Thomas, A. L., *The Allocation Problem, Part II*, American Accounting Association, 1974.

Van der Zee, H., *Measuring the Value of Information Technology*, Idea Publishing, 2002.

von Mayrhauser, A., *Software Engineering: Methods and Management*, Academic Press, 1990.

Walker, R. G., and S. Jones, 'Measurement: A Way Forward', *Abacus*, October 2003.

White, T., *Reinventing the IT Department*, Butterworth-Heinemann, 2001.

Whitton, N., *Managing Software Development Projects*, John Wiley, 2nd ed., 1995.

Youll, D. P., *Making Software Development Visible: Effective Project Control*, John Wiley, 1990.

APPENDIX

GLOSSARY

Day 1 issues	Urgent or critical issues preventing software being put into production.
Day 2 issues	Problems with inputs, processing or outputs which either become apparent after the software has been put into production or do not need attention until later dates (e.g., preparation of half-yearly statements or end-of-year tax filings).
Defect	A deviation from specified requirements (Whitton, 1995).
Enhancement	A special class of adaptive maintenance associated with varying functionality (von Mayrhauser, 1990).
Maintenance	Maintenance (following use of this term in the antitrust court order on IBM—see Jones, 1998) is regarded as the repair of defects, also known as corrective maintenance. Changes to the system environment may require adaptive maintenance, while eliminating inefficiencies in existing functions may be termed perfective maintenance (von Mayrhauser, 1990). Compare with Enhancement.
Migration	The transition from an existing system to a new system. It may include preparation and roll-out of the new system where more than one department or site is involved as well as the accurate and complete conversion of data from the existing system.
Milestone	A significant event at which progress is measured. It may be either the completion of a significant portion of work or achievement of a key element of the project (Buttrick, 2000). Also see Schedule.
Module	A complete set of instructions to execute a related set of tasks with a defined single entry and exit point.
Production	The software is in day-to-day use by end users.
Requirements	A precise expression of what is required by a viable solution allowing the finished software to be evaluated.
Schedule	A timetable at two levels, summary and detail showing activities to be performed (work packages), dates of check and review (milestones), time constraints and interdependencies with other projects (Buttrick, 2000). In the detailed level it will include resource and responsibility allocation.
Testing	Procedures and processes to discover errors (Myers, 1979).
Turnkey	Supply of both hardware and software under a contract.