

Lessons for Testing From Financial Accounting: Consistency in a self-regulated profession

Douglas Hoffman
CAST 2008

“Accounting is the discipline of measuring, communicating and interpreting financial activity.”^[1]

“The purpose of accounting is to provide the information that is needed for sound economic decision making. The main purpose of financial accounting is to prepare financial reports that provide information about a firm's performance to external parties such as investors, creditors, and tax authorities. Managerial accounting contrasts with financial accounting in that managerial accounting is for internal decision making and does not have to follow any rules issued by standard-setting bodies. Financial accounting, on the other hand, is performed according to Generally Accepted Accounting Principles (GAAP) guidelines.”^[2]

“Software testing is the process of measuring, interpreting, and communicating important qualitative aspects of computer programs. The purpose of software testing is to provide the information about software quality that is needed for to improve quality and make sound business decisions.” – Douglas Hoffman

Summary

To say that accounting concepts and methods have been around for a long time is a gross understatement. Accounting records date back over 7,000 years, recording business transactions and inventory. Over the centuries accounting has evolved (and continues to evolve) to become an effective set of practices based on principles and standards. Financial accounting is that part of accounting focused on consistent and complete reporting of financial information for outside stakeholders. Managerial accounting focuses on the reporting of financial information as input for making management decisions. At first look, the accounting field appears to be defined by strict rules using arcane language. Financial accounting standards are applied across industries to make financial statements consistent and thereby more understandable and comparable. A closer look at accounting, and particularly financial accounting, reveals that the standards are adjusted to accommodate real-world constraints and practical methods – **context-driven** standards.

By contrast, the field of software testing is young, having been around for about 50 years. Before then, programmers simply did whatever testing they felt appropriate during software development. As the software development field grew, roles and responsibilities became more clearly differentiated, During the 1960s the software creation process evolved to become

¹ Wikipedia, 13 March 2008; <http://en.wikipedia.org/wiki/Accountancy>

² QuickMBA/Accounting/Financial Accounting; Copyright © 1999-2007 QuickMBA.com
<http://www.quickmba.com/accounting/fin/>

logically separated from software validation. Many people have tried to describe a universal software testing methodology^[3], but the usefulness and applicability of these different approaches are hotly debated today.

Stakeholders cannot generally compare (or understand) test reports without well defined, stable standards. Software testing methods used within industries vary substantially; terminology is inconsistent throughout industries, test reports include many different measurements, use various formats, and have different contents. On many projects the format and content of reports evolve within the course of a single release. Factors labeled with the same terms are often counted differently or embody altogether different concepts. The methods and vocabularies between industries are so different that software testers from different industry segments often cannot even discuss testing. (This sometimes happens between projects within a single organization.) The software testing field lacks a common vocabulary and shared definitions of terms. It lacks a set of assumptions, principles, or methods that apply to software testing across diverse parts of the software development industry.

As different and the fields of Accounting and Software Testing may seem, there are valuable lessons to be learned from *accountancy* and *accounting*^[4] that are applicable to software testing. Where accounting's unit of measure is dollars, software testing has many measures for defects, tests, test outcomes, and other qualitative factors. There is currently no single element in software testing that could represent a common characteristic for everything we might measure, so it isn't possible to summarize software quality in a single equation as accounting does^[5]. But, both accounting and software testing must present information in an understandable form for it to be useful. Software testing would benefit from using a common language and common methods to consistently identify and count items. Consistency is needed within organizations and across industries so stakeholders can understand and compare qualitative information. This paper recasts several processes, rules, and measures borrowed from financial accounting to apply them in the context of software testing. The lessons help us understand the value of processes, keeping measurements simple, test strategies, documentation, and more.

Financial Accounting and Software Testing

In much the same way that software quality assurance is much broader than software testing, accounting is much broader in scope than the financial and managerial segments described here. Most implications drawn from accounting for software testing are related to measures, metrics, and processes because financial accounting is concerned about one metric: dollars. This paper focuses on some of the aspects of financial accounting that provide lessons that can be applied to software testing. The specific lessons drawn upon are outlined in [Table 1](#).

³ Some examples of prescribed test methodologies are: IEEE Standard 829-1998; Craig and Jaskiel, "Systematic Software Testing;" and Buwalda, Janssen, and Pinkster, "Integrated Test Design and Automation."

⁴ *Accountancy* is the profession of *accounting*, which is the methodology. Wikipedia, Op. cit.

⁵ Financial accounting has used a single summarizing model since the 18th century:

$$\text{Assets} = \text{Liabilities} + \text{Equity}$$

The equation basically states that the financial value of an entity (the Assets) is equal to what the entity has been given or earned (the Equity) and what it has been loaned (the Liabilities). Equity changes through operations (revenues and expenses), gains and losses, and owners' contributions and withdrawals but the equation always balances.

◇⁶ The *foundation for accounting* is based on three types of tenets: assumptions, principles, and guidelines. *Assumptions* define and limit the scope of accounting. *Principles* lay out the basic axioms for accounting. *Guidelines* describe exceptional circumstances and reasoning where assumptions or principles may not be strictly applied. Some of the tenets have parallels in software testing. In [Appendix 1](#) I provide a detailed list of the basic tenets of accounting.

Software Testing Principles can be described using a similar three-tiered approach. The rules and responsibilities for software testing could define rules such as a definition for what constitutes a defect, possible test outcomes, and test report requirements. Where the *Assumptions* describe the boundaries of software testing, Assumptions for software testing might include ideas like: ‘the purpose of testing is to provide information of interest to some stakeholders,’ ‘software testing is always incomplete since there is always a potential for undiscovered errors,’ ‘software tests should be designed and created to provide the most valuable information for important stakeholders,’ and ‘test measures and metrics represent information from specified products/projects during specific time periods.’ These assumptions describe the common scope for software testing and should apply across test groups and industries.

The *Principles* describe what needs to be done, how things are defined, and fundamental approaches to be taken in software testing. Software testing principles might include things like: a software defect is defined as a discrepancy between expected (specified) behavior and actual behavior, and defect counts should not include cosmetic errors that do not lead to inappropriate user actions.

The *Guidelines* provide a set of rules that allow flexibility to accommodate industry differences and operational reality. Possible guidelines might include: when the verdict from a test should be recorded as a failure until it is known to have passed, and accepted industry practices should be followed even if they differ from the assumptions and principles.

◇ The *Monetary Unit Assumption* says that all values are translated into Dollars (or Kroners, Rupees, Yuan, etc.). Accounting concerns itself with items that can be valued or measured in money. Things like customer loyalty (goodwill) may be quantified, but only the financial value which is quantified in dollars will be included in financial records.

For software testing, this could translate to using one unit of measure for all values. (We may use several different factors, such as defect counts and lines of code. Each factor should be measured the same way, converted to, or normalized so that we don’t combine function points and lines of code.) Each factor should have the same meaning across an industry segment. If a value cannot be ascertained (e.g., lines of code for a purchased dll library), then that factor is not included in any measures or metrics.

⁶ The symbol ◇ is used to denote the beginning of the lessons

Accounting Concept	Software Testing Implications
Accounting Tenets – Assumptions, Principles, and Guidelines.	The rules and responsibilities for software testing could be described using a similar three-tiered approach.
Monetary Unit Assumption – All values are counted in dollars. [When a dollar value cannot be assigned, it is not recorded.]	Values are all stated in (or converted to) one measure. If the value cannot be ascertained it does not become part of any measures or metrics.
Basic Accounting Cycle	Software testing and reporting may follow a similar cycle.
Generally Accepted Accounting Principles (“GAAP”) – A combination of basic accounting tenets, rules and standards from FASB, and the generally accepted industry practices.	<ol style="list-style-type: none"> 1. The tenets are only a foundation for rules, standards, and methods. 2. Rules, standards, and methods may vary by industry.
Materiality Guideline relaxes GAAP requirements when the impact is small.	Some discretion is allowed in adhering to conventions when doing so costs more than it’s worth.
Full Disclosure – Reports disclose all information thought to be important to an investor or lender within the report or in the notes to the report.	Report all relevant information without regard to its implications. Then clarify or qualify the information as necessary.
Conservatism Guideline – report minimum income values and maximum expenses	Report all potential problems and unknown outcomes in the data, with footnotes (if deemed necessary)
Balance Sheet (Current balances)	A report describing the state of testing at a specific point in time.
Income Statement (Profit & Loss)	A report showing the net effects of activities during a specified time period.

Table 1: Summary of Implications

◇ The **Basic Accounting Cycle** has five parts:

1. *Identifying and recording* business transactions
2. *Posting* to journals and the general ledger
3. *Adjusting* the general ledger to reflect actual business operations (e.g., bad debts, accrued interest, taxes, etc.)
4. *Preparing financial statements* for the time periods
5. *Closing the books* (resetting accounts to zero for the next accounting cycle)

A **Basic Software Test Execution Cycle** might consist of:

1. *Performing tests and recording results*
2. *Updating quality data* to reflect the new results
3. *Adjusting quality data* to reflect actual business operations (e.g., removing the results of obsolete tests, reflecting reclassified defects, updating code size measures, etc.)
4. *Preparing quality reports* for the time period(s)
5. *Re-Baselining the records* (resetting counts to zero for the next testing cycle)

The testing cycle described here covers only running and reporting test results, thus excluding test planning, logistics, test case creation, maintenance, etc. Even so, quality data (in part 3) is seldom adjusted. This causes distortion of the quality picture from the data. For example, rerun tests are counted the same way as running new tests, and tests results from older versions of software are accumulated with test results from the current version. Some of the distortions might be removed (in part 5) since the counts for measures and metrics would restart each test cycle.

◇ Accounting follows a set of **Generally Accepted Accounting Principles** (“GAAP”). GAAP is extremely useful for standardizing and regulating accounting definitions, assumptions, and methods. It legitimizes industry differences and codifies how standards for accounting must be defined and justified. Although variations exist between industries, using GAAP results in consistent reports over time and across industries.

Consistent reporting over time allows fair assessment of the progressive performance of an organization. This context enables stakeholders to understand, evaluate, and compare financial reports at different points in time, understand financial health of organizations and industries, and provides a basis for ascertaining the validity of financial statements. It isn’t useful to compare a financial statement for one time period where revenue is counted when money is deposited in the bank (cash accounting) with a financial statement from another time period where revenue is counted when goods are shipped (accrual accounting)^[7]. Adherence to GAAP assures that standard accounting methods are consistently applied within an organization and if methods are changed the impacts of the changes are fully explained and separated.

Adherence to GAAP results in comparable statements by organizations within an industry segment (to the extent they interpret and implement the common industry practices the same way). Although financial statements from banks are very different from manufacturing organizations, statements from different banks are largely comparable, as are the statements from manufacturing organizations. Financial statements for all firms are superficially very similar in terminology and format because industries share the same foundational principles (GAAP), and the differences are understandable and justifiable within each industry segment.

There are three primary parts in **GAAP**:

1. The basic accounting tenets (assumptions, principles, and guidelines)
2. Detailed rules and standards issued by the Financial Accounting Standards Board (FASB), a group of accounting experts independent of all other business and professional organizations^[8].
3. The generally accepted industry practices for each industry segment

The software testing profession would benefit from identifying and standardizing generally accepted software testing principles (GASP) and identifying some of the more and less applicable contexts where the principles apply. A structure similar to GAAP might be created for

⁷ Cash accounting recognizes revenue based on how soon customers pay, while accrual accounting recognizes revenue based on how fast goods are shipped. Revenue is usually lower under cash accounting.

⁸ See <http://www.fasb.org/facts/index.shtml#mission> for facts about the FASB

GASP. A Software Testing Standards Board (STSB)⁹ would need to be created to identify, articulate, oversee, and adjudicate software testing issues. The STSB would represent the many constituent private sector stakeholders and use an open decision-making process to establish GASP standards. STSB members would be expert software testing stakeholders and need to sever all connections with the firms and institutions they served prior to joining the Board. The members of the board would be selected based upon their knowledge of software quality, software testing, business, and a concern for the public interest in matters of software testing and reporting.

Under GAAP, many accounting rules require accountants to apply expert judgment in deciding key assumptions that have “material” impact on the reported results (e.g., whether to use cash or accrual accounting methods, or accounting for software development costs as a period expense or an investment to be recouped by future sales). Similarly, software testing is context-specific and requires expert judgment to decide on key factors in testing and reporting. GASP must account for the relatively young state of the science and large variation in contexts for software testing. Any “best practice” for software testing must be applicable without exceptions or qualified so that following it is not mandatory (which makes it a “good practice” in some contexts and not applicable in some others).

◇ The **Materiality Guideline** relaxes certain GAAP requirements if the impact is not large enough to influence decisions so that users of the information should not be overburdened with information overload. The general rule for assets depreciates (uses up) its value a little bit at a time over the useful life. To a large corporation it may cost more to compute and track depreciation for assets costing a few hundred dollars than the item itself costs. The materiality guideline allows accounting for those assets as immediate expenses because the numbers on the financial statements will not change.

An example parallel guideline for software testing would be removing low priority/low impact defects from defect counts. The metrics might give a false impression if it includes low priority defects not likely to be fixed. The biased metrics could lead stakeholders to poor conclusions about what to test or what to fix. Another example of materiality might be elimination of report items that do not provide any information that could change stakeholder behavior. The cost of gathering and reporting the information is wasted when no stakeholder will make decisions based on it.

◇ The **Full Disclosure Principle** and **Conservatism Guideline** for accounting state that all information thought to be important to an investor or lender should be disclosed within the statements or in the notes to the statements and it must include all known information that could negatively affect the financial statements. An example would be the possibility of losing a pending lawsuit that could force the company to pay out a large amount of money. However,

⁹ Bryan Kocher published an article “A Model for Software Practices from the Accounting Profession” (IEEE Software, Volume 17 Number 1, January/February 2000) calling for the establishment of an Information Systems Standards Board (ISSB) to create a set of standards for software system design and construction, modeled on the FASB and accounting professions. I do not share his confidence that Generally Accepted Programming Practices exist and can be simply articulated, but I do agree that establishing a system based on *personal responsibility* is preferable to *endless regulation*.

speculative information that might positively impact the organizations financial is not allowed to be included in the body of financial statements (except as footnotes).

The parallel for this in software testing is for reports to include all information known at the time of report generation. For example, non-reproducible errors would need to be documented (at least in footnotes). The preliminary outcomes from the most recent test runs would be included in all measures and metrics. Footnotes for the measures and metrics can explain and clarify if pending analysis of some test failures might change their result to passes. Another example would be where a defect report is logged for a potential problem immediately before measures or metrics are reported. Even though analysis might lead to closing the report (e.g., as 'Not a Problem' or 'Duplicate'), the report should be counted as an error in measures and metrics, using footnotes to explain and clarify.

◇ Two separate financial reports are used to describe the current state of the organization. The *Balance Sheet* provides a snapshot of the financial state at one point in time. This often appears in financial reports with the corresponding Balance Sheet values from other time periods for comparison. The *Income Statement* (a.k.a., Profit and Loss Statement) describes the recent financial changes in accounts due to business activities during a period of time (e.g., monthly, quarterly, or yearly). Together, the two reports show where the organization stands and how the organization has been doing recently.

Software test reports can be logically separated in a similar way. One component in the report covers the state of the software being tested. This quality indication is a statement of current software behavior, completeness, or readiness. The other component covers the recent history of changes in behavior, completeness, or readiness. The first part is about the product and includes information needed to understand the quality of the software at that one point in time. The second part is about progress and it describes recent changes in the product quality.

Conclusions

The rules for accounting have been established and refined for centuries, where the computer software testing profession operates without established rules and has existed for roughly half a century. Although software testing is in its relative infancy and we are still establishing a vocabulary to describe what we do and I think that valuable lessons for software testing are available from accounting. These lessons will help software testing evolve and mature more quickly.

The structure and concepts for the rules of accounting can be used to provide a useful conceptual framework for a set of rules that could govern software testing. A major lesson from accounting in this regard is that rules differ from one segment of the industry to another because of the widely differing requirements for each. Another lesson is that it is possible for a profession to standardize and regulate itself through personal responsibility rather than by externally imposed regulations.

The software testing profession will be able to approach standardizing the rules when the information we measure can be unambiguously assigned values on a single scale (or a few

Lessons for Software Testing From Financial Accounting

distinct scales). Monetary value is universally applied in accounting in spite of the scores of currencies in use and accounting methods work the same way for all of them. Fundamental measures in software testing are not well defined yet: “test case,” “defect,” “defect report,” “code complexity,” and “program size” are examples of commonly used terms that have several, often conflicting definitions. The differences in the terms are not only due to different industries, but many are hotly debated between industries and sometimes within a single company.

Generally Accepted Accounting Principles validate and institutionalize differences in rules across industries. Different rules, formats, or terms are due to justifiable context specific differences and the same accounting principles are applied across the industry segment. This results in accounting reports across each industry that can be easily read, interpreted, and compared. Similarly, context differences in different industries that test software result in different testing and information requirements. Similarly defined tenets in software testing could facilitate acceptance of common foundational rules and approaches while allowing for context specific solutions. Although I have not encountered any universal “best practices” in software testing that I could professionally support, they may exist, and context-specific “good practices” have been known for decades. The software testing profession would benefit from standardizing them and identifying some of the more and less applicable contexts.

A materiality guideline allows for some discretion in measurement and reporting of useful information to an appropriate level of detail. This flexibility is critical since the requirements and practices for software testing varies so much today. Reporting standards that require conservatism and full disclosure could also benefit software testing. An example of conservatism in testing would be a requirement for test reports to include an explicit list of tests that were not performed (including ones considered but not created) and the rationale behind their not being done. Full disclosure would require that any known information that might influence stakeholders’ decisions be included in reports (e.g., non-reproducible errors). Such standards would remove ambiguities that may be exploited to bias reports or justify sub-optimal testing by providing all of the potentially negative information that might influence stakeholders’ decisions.

Software testing might also benefit from considering two distinct types of reports: one to display the current state and another to describe recent activities. A report of the current state provides a snapshot of status, while the activity report shows changes occurring during specified time periods. One shows where we are and the other how we got here.

Because the state of the science in computer science is still young, coming up with generally accepted software testing principles may not be within our grasp anytime soon. However, endeavoring to do so will improve the state of the science by exploring the serious question of why there are different definitions and tenets. Applying some of these lessons from financial accounting could improve the software testing profession. Standardizing terms, definitions, and report formats could reduce some of the redundant work done by nearly all software test organizations, possibly freeing up resources to delve more deeply into the invaluable job of more and better software testing.

Appendix 1

Twelve basic accounting assumptions, principles, and guidelines:

1. **Economic Entity Assumption**
Accounting keeps track of *business* transactions (as separated from *personal* transactions), so that the finances of the firm are not co-mingled with the finances of the owners.
2. **Monetary Unit Assumption**
Economic activity is measured in dollars, and only transactions that can be expressed in dollars are recorded. [This means, for example, that the effects of inflation are ignored in accounting.]
3. **Time Period Assumption**
This principle assumes that it is possible to report the complex and ongoing financial activities in relatively short, distinct time intervals. The time interval must be included in P&L and Cash Flow reports, and the specific date on a Balance Sheet.
4. **Cost Principle**
“Cost” refers to the amount spent (cash or cash equivalent) when the transaction takes place. This means historical values are not adjusted to reflect increases in value. [Hence, accounting values other than stocks and bonds do not reflect the amount of money a company would receive if it were to sell the asset at today’s market value.]
5. **Full Disclosure Principle**
All information thought to be important to an investor or lender should be disclosed within the statement or in the notes to the statement.
6. **Going Concern Principle**
Accounting assumes that an organization will continue to operate and will not liquidate in the foreseeable future. This allows deferring of prepaid expenses until future accounting periods.
7. **Matching Principle**
This principle requires that expenses be matched with revenues in the same time period [using an accrual basis of accounting], even when paid at different times. Where the expenses cannot be matched with particular revenues (such as advertising costs that increase future sales), the expense is charged in the period it is incurred (when the ad is run) and the revenue recorded when the sale is made.

Lessons for Software Testing From Financial Accounting

8. Revenue Recognition Principle

Revenue is recognized as soon as everything that is necessary to earn the revenue has been completed for a product or service (but not before that), regardless of when money is actually received.

9. Materiality Guideline

A modifying convention that relaxes certain GAAP requirements if the impact is not large enough to influence decisions so that users of the information should not be overburdened with information overload.

10. Conservatism Guideline

If there are two acceptable alternatives for reporting an item, the alternative that will result in less net income and/or less asset amount is chosen. [For example, an accountant may write inventory *down* to an amount that is lower than the original cost, but will not write inventory *up* to an amount higher than the original cost.]

11. Cost-benefit Guideline

A convention that relaxes GAAP requirements if the expected cost of reporting something exceeds the benefits of reporting it.

12. Industry Practices Guideline

Accepted industry practices should be followed even if they differ from GAAP.