Software & Valuation In The Information Society

Part Two: The Software Inventory

By Dwight Olson and Denny Kolb

The intangible assets owned by any software development enterprise, include: the technical knowledge (and know-how) of its staff, the competence of its sales force, the business knowledge (and experience) of its management, its goodwill and reputation (including trademarks), the value of its intellectual property (including copyrights, trademarks and patents), the commercial value of its licenses, and the value of its software inventory.

Recent *les Nouvelles* articles[1] discussed the need to accurately identify, analyze, and evaluate intangible assets within an enterprise as regulations such as Sarbanes Oxley and Financial Accounting Standards Board (FASB) Statements 141, 142 that force governance of these assets. There were three questions asked: Which software assets have value? Which software assets can be monetized? And third, what is the commercial value of software, whether as a capital asset or as debt security? Further to a recent les Nouvelles article's discussion of a software asset's monetization,[2] this article will address which software assets have value.

Mr. Kemerer, a writer for *Information Week* asked "Do you know what your software inventory is worth?" It's a question you may not have addressed, or even thought of, but knowing the value of your software inventory can help you manage it better.[3] Furthermore, the current spate of mergers and acquisitions means that enterprises must estimate the value of existing software application inventories for financial and tax reasons.[4] It is not an easy task as many software products today, including embedded software, are actually complex information systems that involve many components, networks and other related assets. Those assets are highly sophisticated, highly integrated, and many have the ability to communicate directly via secured network connections. Many software products cost millions of dollars to develop, millions of dollars to deploy, and tens of millions of dollars to adapt to ever-shifting conditions over time.

The Impact of Software Accounting Standards

A very significant amount of time has passed since the American Institute of Certified Public Accountants (AICPA) Statement of Position 98-1 (SOP 98-1) and the Financial Accounting Standards Board (FASB) Statement No. 86 (No. 86) were approved. According to many experts, both standards exhibit a weakness in software attributes, and particularly as it relates to what financial information is important to provide.

FASB's Statement No. 86 and AICPA's Statement of Position 98-1 are the current accounting standards concerning how software assets should appear on corporate financial statements. However, some current proposals on software valuation, such as by Mordechai Ben-Menachem and Ilanit Gavious who believe that the techniques, models and concepts used by these standards have not kept pace with software advancements, such as those commonly used in costing and development life cycle models for creating and maintaining software assets.[5] The current accounting treatment of software as if it were a mechanical device, such as the case under FASB 86, results in a misunderstanding and potentially incorrect valuation of software. The fact is that only a small fraction of the costs associated with the development of software is traditionally recognized as an asset, while the vast majority of those costs are immediately expensed as they are incurred. Consequently, the true value of software is immensely underestimated on the balance sheet.[6]

To compound such undervaluation, Hughes and Cotterell, discuss FASB's Statement No. 86 and AICPA's Statement of Position 98-1, where capitalizeable costs are principally restricted to certain development costs that do not include the costs of the

1. "Intangible Compliance: Capturing Overlooked Value," *les Nouvelles*, March 2008 and "Software & Valuation in the Information Society," *les Nouvelles*, June 2008.
2. "Leveraging Software via the Capital Markets" dated September, 2008.
3. "Value Your App Inventory," Chris F. Kemerer, *Information Week*, May 25, 1998.
4. *Ibid*.
5. "Accounting Software Assets: A Valuation Model for Software," *Journal of Information Systems*, Fall 2007.
6. *Ibid*.

software's future modification, improvement, and general evolution.[7] They imply that this may produce an un-intended effect in the current accounting treatment of software that leads to a disproportionate downward bias in the reported fair value of software. Indeed, costs incurred for such on-going software evolution may be five (5) to twenty (20) times the original development costs that are incurred for the software's first release.[8]

In addition to undervaluation problems resulting from only capitalizing a small part of software's costs, both accounting standards are based on an outmoded system lifecycle model that is essentially incorrect. FASB's Statement No. 86 and AICPA's Statement of Position 98-1 both use the same basic model.[9] This model is very similar to the waterfall model first described by Winston Royce in 1970, ironically as an example of a flawed model for software development. The software engineering community has not used this method for at least twenty years because it leads to misleading and untrustworthy project results. In fact, software products (internal or external) are almost always put into use with only partial functional requirements fulfilled, and then evolve over time (frequently, over very long times) as the software is enhanced, customized, adapted and new requirements are discovered and changed.[10] One of the greatest inherent values of software is its ability evolve and adapt to new business challenges through modifications, revisions and continuous improvement.

Another valuation problem arises when overhead development costs are not to be capitalized even if management believes such overhead is incremental to the software project.[11] The current accounting standards only allows a company to capitalize costs that are directly associated with developing or obtaining the software, such as materials, services and payroll-related cost for specific employees. Therefore, it has been proposed that not only is software immensely undervalued in the balance sheet, but current earnings are also excessively depressed by immediate expensing of the vast majority of software costs. As a result, in an era where most business activities essentially rely on IT systems, reported earnings become less reliable and less relevant for investment decisions.[12]

The current capitalization standard for software development is often and aggressively criticized. Almost twelve years ago, the software Publishers Association (SPA) wrote a letter to FASB stating, "We do not believe software development costs are a useful predictive factor of future product sales and the users of financial statements have a high degree of skepticism when it comes to soft assets resulting from the capitalization of software development costs."[13] Of course, respondents to the SPA counter claimed that the current method of software capitalization is relevant to investors. Yet, analysis made by Kevin Den-Adel implied that a full capitalization of software development costs with uniform amortization is more value relevant than the current accounting practice, which may explain a reason for the variation in stock prices.[14]

■ Dwight Olson, CLP, V3Data, Principal, San Diego, CA, USA
E-mail: dcolson@ix.netcom.com
■ Denny Kolb, DNE Information Services, LLC., Managing Partner, San Diego, CA, USA
E-mail: dckolb@gmail.com

Generally Accepted Accounting Principles (GAAP) tend to treat "purchased" and "developed" assets differently with purchased recorded as an asset at purchase price, regardless of how the asset was originally produced. In the case of developed software, SOP 98-1 requires a differentiation between costs that are expensed and those that can be recorded as assets. The discernment of what is an asset via where the costs occurred is not relevant to asset-value. The proper discernment must be between cost of creative processes that create objects (components) and non-creative processes that manipulate existing objects.[15] SOP 98-1 does not recognise objects, only some of the input occurrences material to the formation of some objects. By definition, this does not match the concept of asset as object. The conclusion is that accounting has to recognise that software is an object.[16]

A 2004 report, that was based on a survey of 226 top professionals from many industries revealed

7. Business Performance Management (BPM) Forum, 2004 article.
8. *Ibid*.
9. Accounting Software Assets: A Valuation Model for Software, *Journal of Information Systems*, Fall 2007.
10. *Ibid.*
11. *Ibid*.
12. *Ibid*.
13. Software Publishers Association, letter to FASB dated March 14, 1996.
14. "The Value Reference of Alternative Accounting Treatments of Software Development Costs," University of Iowa - Department of Accounting, September 10, 1999.
15. "Accounting Software Assets: A Valuation Model for Software," *Journal of Information Systems*, Fall 2007.
16. *Ibid*.

that more than 75 percent of respondents reported their businesses were dependent on IT, that nearly 50 percent of all capital expenditures were spent on IT.[17] As well, nearly 70 percent of respondents report an inability to compute return on investment for software capital goods and typical system lifetimes of information systems have an average over twenty years.[18] When buying complex technology, companies must look at the total cost of the project, which consists of hardware, software and services. An enterprise's financial dependence on IT, and with an order-of magnitude difference between license cost vs total cost, clearly shows that the present standards situation really does not reflect reality.[19]

Whereas computer hardware may deteriorate from normal "wear and tear" over the course of its ordinary use, software is strengthened and enhanced by use, and only loses value as a result of inherent defects that remain uncorrected or where the business premise for its original functionality changes renders the fundamental operation of the software obsolete.[20] Software products should be understood by looking at the processes through which a software product is created and evolved rather than looking at the mere cost of labour associated with its original development. Focusing on only direct, hard programmer costs (for example payroll), while ignoring the overall investment in the processes, project, and other development outputs, misunderstands the true investment that is required for software development, which leads to the mistreatment, and under valuation of such software. Today, software products are built by software designers, programmers, engineers, quality assurance, document writers, system and market analysts, etc., all with highly integrated, multi-disciplinary knowledge-bases that provide added value to the software inventory.

Therefore, if one values software on a governance and fair value basis one must also take into account not only direct, first-release, project costs, but also all of the developments and processes that burn through the vast majority of money. Then it may be possible to provide meaningful information for financial governance of software as well as dealing intelligently with software revenue recognition.

17. BPM Forum, December, 2004.
18. *Ibid*.
19. "Technology Paradise Lost: Why Companies Will Spend Less to Get More from Information Technology," Erik Keller.
20. "The Mythical Man Month: Essays on Software Engineering," Addison-Wesley, 1995.

Software products are rarely licensed on a standalone basis. More often, they are licensed as a package including: upgrades, maintenance, special services, training, customer specific enhancements, hardware interfaces, and installation services. A software license is typically negotiated as a whole, with the license forming the basis for the relationship. This inevitably raises two financial issues; first, how to allocate revenue among the components, and then how to allocate revenue among accounting periods if components are delivered over more than one reporting period.

Under United States (U.S.) GAAP, no software revenue can be recognized unless a signed license agreement exists. U.S. GAAP rules state that where customary practice is to obtain a signed contract (license), no revenue may be recognized until a final written contract has been signed by both parties. Thus, no revenue may be recognized until the license is signed even if fees have been paid and certain components and services delivered.

Under SOP 97-2, total license revenue must be allocated among the components of a license in proportion to the revenue value of each component, regardless of the prices specified in the contract. However, revenue may only be recognized if there is Vendor-Specific Objective Evidence (VSOE) of revenue value for each component in the license, or when the revenue value of all undelivered components are known. SOP 97-2 stipulates that VSOE can only exist if the component in question is also sold separately. Rarely in software product licensing are software upgrades, maintenance, special services, training, customer specific enhancements, hardware interfaces, and installation services offered separately. Thus, if there is no VSOE of revenue value, no revenue can be recognized, although there is exception for software maintenance.

This concept implies that revenue value cannot be measured using estimated costs for completion of the components or services to be delivered. The only factor that can be taken into account in practice is the price of the component when licensed separately. Applying this rule has created numerous problems for companies in practice. For example, if the element is never licensed separately, there can be no VSOE of value. Licenses often include rights to future versions, products or enhancements, which, by definition, are not yet available. Consequently, there can be no VSOE of fair value for these elements. A software company, therefore, cannot recognize any revenue on delivered elements (components) so long as there are undelivered elements for which no VSOE of fair value exists.[21]

Software Financial Governance and Valuation

In any valuation discussion, one must first recognize that software is a valuable business asset. For the purpose of this discussion, the identification of software assets will focus on software products and components that are developed by a business for commercialization. In that context, software could be categorized as follows: software product (developed for internal use or external licensing), embedded software (typically in special hardware such as in equipment or in a cell phone), or databases managed by software. Included also are Web sites (content, service, etc) in the software product category.

In light of the standards issues discussed above, one might question whether or not a non-specialized accountant or auditor is qualified to provide accurate, up-to-date and reliable information concerning software governance or fair value calculations. Can any accounting audit provide enough information for such an undertaking where the software inventory, as a significant and expensive group of assets is essentially absent from corporate management's or an investor's view? The hidden business value of such software assets has potentially significant implications. While the inability to account for software value means that its contributions to an enterprise's true value are not reflected on the enterprise's income statement or balance sheet, the lack of an accurate valuation may also hamper any stakeholders' decision making ability.

One might wish to consider a more reliable valuation model for software based upon measurements of software that takes into full account all of the costs incurred to create the software inventory. In this model, costs are collected by an automatic tool and stored in a database of software enterprise assets and to costs is added the effect of each individual module's relative significance to the enterprise.[22] This model provides a systematic algorithm for software amortization, based on the decrease in its usability (or commercial value). True software value, therefore, recognizes that the intrinsic value of a software product must include all costs incurred in the software's development. That calculation will include the costs incurred prior to capitalization, as well as costs incurred after first release. These measurements should ideally be collected in a software inventory system.[23]

The software product inventory system should be designed to provide accurate and up-to-date information concerning software's product components and recognition of all related software components. Such a scheme should show all software component assets hierarchically with their relationship, history, complexity, other influencing factors, and costs incurred. This information could also be useful in other valuation models for computing context driven valuations for software or an appropriate software based IP. This system should provide for information necessary for setting software risk and influencing factors that can be applied to valuation calculations. Hopefully, this could lead to better governance, monetization and valuation of software products.

Which Software Assets Have Value?

Data Securities International introduced the concept in the mid 1980s for a Total Software Value (TSV) that uses the composites of Ownership Value (OV) or the software inventory, Market Value (MV),

Table 1: Software IP Bundles

1. Marketing and Sales	2. Trade Secrets and Know How
• Marketing plans and collateral	• Design documentation
3. Patent(s)	• Source code
• Defensive and offensive	• Formulas
4. Copyrights	• Process know how
• Executable code	• Operating platforms
• User documentation	• Manufacturing instructions
• Installation instructions	• Configuration data
• GUI's	• QA test and procedures
5. Domain Name(s)	• 3rd party software
6. Licenses	• SAAS or ASP databases
• Encumbrances	• Client databases

21. Price Waterhouse Coopers - A shifting software revenue recognition landscape? Insights on potential impacts of IFRS and US GAAP convergence. *www.pwc.com/extweb/pwcpublications.nsf/docid/568C611407DEAAE4852573E20006A7D5/$FILE/shiftingsoftware.pdf.*

22. "Accounting Software Assets: A Valuation Model for Software," *Journal of Information Systems*, Fall 2007.

23. *Ibid*.

Table 2: Software—Intangible Asset Inventory	
1. Marketing and Sales	2. License Management
• Marketing plans and collateral	• DRM and license controls
3. Client Support Systems	• Back office system
• Installation and training	4. R&D Systems
• User documentation and help	• Internal Design Documentation
• Client databases	• Source code with comments
5. QA and Testing	• Source code control with comments
• Bug/support system	• ASP databases
• Testing code and data	• QA test and procedures
6. Manufacturing System	• 3rd party software
• Specific build guides	• Open source & strategy
7. Commercialization Strategy	• Client databases
• Product plan (release and updates)	

and Internal Cost Savings (ICS) as values and influencing variables of software as a financial asset.[24] A TSV software inventory valuation (OV) analysis looks at the sum total (or bundle) of the various software components or intellectual assets that make software usable as a product. The following three tables are included to help understand these software product asset components. Table 1 is a view of the various types of Intellectual Property (IP) involved and a list of the software components typical of that IP. The items listed on the left side of Table 1 are typically known or available to the market. The items on the right are the background assets (or software inventory) owned by the software owner, and are what Karl Jorda hints to in his comment "Patents are but the tips of icebergs in a sea of trade secrets.... patents have value and possibly more valuable are the background assets."[25] These are listed for reference. Table 2 organizes the right side of Table 1's "Trade Secret and know how" background components that are typically produced during the development cycles for a software product. Table 3 shows the complexity at stake in determining which IP may control the software component as each of the identified components can be protected by different types of intellectual property rights.

IP and Software Inventory Valuation

It is in commercialization where software IP (see Table 1 Software IP Bundles) takes shape and the software patent, trademark, and copyright emerge and sometimes collide (see Table 3). There is often confusion as to what software IP to value and how to value each type of software IP property. However, how can one really value the software IP property, such as a software patent, outside the context of the software inventory? If there is no intent to commercialize then there is no software inventory and thus no monetary value. This is actually a very powerful factor influencing any corresponding software valuation as discussed by FASB 86. For example, there is a potentially huge difference in the value of a software patent that has not been reduced to practice, and

Table 3: Software Components as Intellectual Property				
Software Component	Copyright	Trademarks	Patents	Trade Secrets
GUIs	√	√		√
Source code	√	√	√	√
Object code	√			
Business processes	√	√	√	√
Data	√	√		√
Table structures	√			√
Documentation internal design and external use	√	√		√

24. "Software & Valuation in the Information Society," *les Nouvelles*, June 2008.

25. "The Differences Between Patent and Trade Secret," Franklin Pierce Law School, Summer/Fall 2004.

for which no commercialization components have been developed, compared to a software patent that has been reduced to practice or commercialized. Aura Soininen said in this thesis, "The value of patents is dependent on the value of the invention it claims, and how it is commercialized. Thus, most of them are worth very little, and most inventions are not worth patenting: it may be possible to protect them in other ways, and the costs of protection may exceed the benefits."[26] A caution note is to recognize that some patents come into existence for market monopoly and portfolio management purposes to be used in cross licensing and is outside the discussion of this article.

In the commercialization of any software product, the projects involved produce a quantity of asset components. These components need to be identified and governed, just as all other "hard" assets. No one would ever think of "not managing" inventory, buildings, or machines, yet for software, a black hole seems to be acceptable. Software governance must be asset-based to be of value. The basic requirement for this is an accurate and up-to-date inventory of all software assets, showing all software components and any hierarchy (updates) with history, complexity and all costs incurred.

Software product development produces software inventory component assets; that is why the component came into existence. Someone determined that the component was needed to achieve a return on investment for the software product or service. The critical issue is whether those component assets are development errors and of no interest for governance and valuation or are they in fact assets in the sense that, for monetization, stakeholders need to receive information concerning their fair value? Tools that enable accountants to attain the level of understanding necessary to quantify them can be attained easily with today's availability of IP management and information products.

Changes to software components are constant and these changes must be reflected in the software inventory database. As stated, there is no technological reason to differentiate between costs for "initial development" and costs that occur later. Many costs are incurred as the component changes, and are usually associated with events that occurred after the initial release of the software product. In many instances those additional costs incurred increase the value of the component. Also, each component will have associated risk and influencing factors that affect its valuation.

Software Valuation Risk and Influencing Factors

To the basic components in the software inventory, risk and influencing factors should be factored-in. But, what risk and influencing factors? In software development, software risk and influencing factors are called Control by Importance and Exception (CIE) factors.[27] In this working paper there is a discussion of a technological-view mapping between actual costs and an individual module's importance to the enterprise (remembering that all software is multi-level, hierarchical and granular). The view of a module's technological importance provides accounting with a valid basis for estimating sensitivity and, hence, a risk factor, accounted for in the software valuation.[28] In this working paper, sensitivity refers to the "degree of care" needed when making a change in a module. In this article, five quantifiable parameters control sensitivity:

- reuse count,
- application mission criticality,
- complexity,
- updating difficulty, and
- internal functionality or interface implementation.[29]

Included here is the Reuse factor from the article. "Reuse count refers to the quantity of uses for this module. For example, in a bank, a module which computes interest may be used to compute interest for many different types of accounts, though each account type may be implemented via a different system. Additional uses of a module means the module has been more expensive to produce but this additional use has a payback by reducing expense in other projects utilizing it. This is similar to what is occurring in the automotive industry. A common platform is more difficult and more expensive to produce, but much more cost-effective to the enterprise. However, if a change is made in this platform, those that change it must take into account all of its users."[30]

26. Patents in the Information and Communications Technology Sector—Development Trends, Problem Areas and Pressures for Change, Aura Soininen, ISBN 978-952-214-344-0 (PDF) 2007.

27. IT Assets: Control by Importance and Exception; a Paradigm of Change Support technology. Working paper 2003, Ben-Gurion University.

28. *Ibid.*

29. *Ibid*.

We also suggest that in any software product valuation the corresponding software inventory should be valued for the current and prior year. Any positive or negative delta variance will provide another risk or influencing factor for biasing the estimated valuation. This factor may also help management understand any software asset impairment should there be a negative variance.

You might remember that in FASB 86 there are different influencing factors at play that more aptly apply to a Market Value.[31] One might be cautioned in preparing a market valuation or a software inventory valuation to include an additional risk or influencing factor. Not present in 1985 during the development of FASB 86 was the world of Open Source including its risks and benefits. Today, software valuations should include the risk or influencing factor of Open Source. For example, the mingling of Open Source source code in proprietary software may imposes a significant risk or influencing factor that may reduce or increase software value.

Open Source Software Product Valuation

How do you put a value on free? That's the question facing investors in open source software enterprises. Few doubt that such enterprises are reshaping the software landscape. Their products based on publicly available source code and developed by global open source communities are giving proprietary software enterprises a run for their money.

If we look at recent Open Source acquisitions, Yahoo acquired Zimbra, an open source email and communications suite provider, for $350 million. Yet, Zimbra only received funding from Benchmark Partners, Redpoint Ventures and Accell Partners, raising $30.5 million with three rounds of investment funding. Another acquisition was BuzzTracker, a tiny news aggregation site, for between $2 to $5 million. Sun Microsystems became the owner of MySQL, the pioneering open source database system. Sun paid about $800 million in cash in exchange for all of MySQL stock and assumed another $200 million in options as part of the deal.

There are many reasons why an enterprise would opt for open source and forego the possibility of making money by the proprietary licensing route. Owners gain profits in other ways than direct revenue license gains. Pratibha Gupta, suggests the reasons include cost savings, in terms of zero investment on seeking patent protection in various jurisdictions, productivity gains, as more and more people will be able to access and try the usage, brand building, since more and more people will be-come aware of the product in the markets and, most importantly, an expanded user base.[32] As the market expands, revenues from sales, one-off licenses, dual licensing, and complementary products and services may be enough to offset the opportunity cost of open source licensing. IBM has used this approach very successfully. Today, almost 40 percent of companies already use some type of open source software, and a further 8 percent have plans to pilot it during 2006. Utility and telecommunications firms, media companies, and public sector bodies lead enterprise adoption by a wide margin. Forty-five percent of the firms using open source have deployed it in mission-critical environments, although the vast majority (70 percent) use it for non-key applications. Web server and server operating systems are the top two areas, with two-thirds of firms using alternatives like Apache, Tomcat or Linux.[33]

At a minimum, one could reason that in certain situations a TSV could also apply to an open source product, even though the source code is left on the Internet and downloadable by anyone. Use, obviously would still be governed by the open source license selected by the owner. One could easily argue that the software inventory or ownership value would be the same as proprietary software inventory, as well as the MV and ICS valuations. Obviously, the risk or influencing factors would be different.

Adding a word of caution: since the underlying software of open-source companies is free, Matt Asay[34] suggests analysts wonder if they're worth multimillion-dollar valuations from investment funds and big-name suitors. Even fans of the open-source movement see signs of froth. We're in the midst of another bubble time for open-source companies and would do well to be prudent in our expectations. He raises an important issue: is most of the money going into open source dumb money? Not dumb in terms of the VCs funneling the money to startups, but dumb in terms of the business thinking underlying the startups?[35]

Then again, Kim Polese,[36] states that the deals might not make sense using traditional valuation benchmarks.[37] The old assumptions about return

30. *Ibid*.
31. "Software & Valuation in the Information Society," *les Nouvelles*, June 2008.
32. "Copyright and Open Source Licensing of Software Work," *les Nouvelles*, June 2008.
33. *Ibid*.
34. CNET's The Open Road blog Web site.

on capital might not apply when it comes to open source. Most users don't pay for the product. At the same time, the collaborative nature of open source cuts development and marketing costs. Open-source firms have a phenomenal user base and are a disruptive force that can be worth a lot of money.[38]

Yet, it seems obvious that open source enterprises that make friends but no money will end up in the same bankruptcy as failing proprietary software enterprises.

Mixing Open Source in Proprietary Software

Palamida and Blackduck Software, two companies that provide products and services to software enterprises that co-mingle the use of open source with their proprietary code, believe enterprises are increasingly mixing internally developed code with external code from the open source community and other third-party suppliers.[39] This practice is a huge productivity multiplier, and according to Blackduck, when managed poorly, mixing code can introduce significant business issues.[40] Un-managed use of open source and other non-owned proprietary code, if ever detected, could open the door for copyright infringement, patent validity issues, or generally be catastrophic for the public software enterprise executives. Such undisclosed use will cause significant errors in any proprietary software inventory valuation or market valuation, especially if part of the valuation is based on lines of source code.

According to Blackduck, application development has evolved into a process of assembly. As developers are writing their code, they capitalize on software reuse by finding software components already written that enhance their projects. Such development techniques are legal—the reused components are available within open source applications and libraries, or developers use third-party components that have been purchased by their employers.[41] When managed correctly, by legal counsel and executive management who set policies regarding use of open source and 3rd party proprietary code, then financial governance of the software product is possible by providing a more accurate valuation. Assuming the open source or third-party components can be detected or extracted from the software inventory, then the appropriate software inventory valuation and Market Valuation (MV) can consider the risk and influencing factor of open source use.

Concluding Remarks

Software valuation needs the help and support of financial and software business professionals. Software management can create and maintain a functional database of the enterprise's software asset inventory to report correctly software assets. The software asset inventory database is only rarely available today, because management has not demanded it. There exist models available for implementation by IT professionals, supported by computing technology that can help alleviate the information gap and support IP valuation professionals, economists, and accountants in their critical corporate mission to provide financial software asset valuation information.

Many valuation experts disagree on which method is best for valuing different types of software. Current methods for valuation of software generally fall within one of the three categories; the Cost Approach for valuing software based on how much it actually costs to create or recreate that asset, the Income (Net Present Value) approach for valuing software based on all revenue (licensing, support, and maintenance), and the Market approach for valuing software based on the licensing, patent royalty rates (if included in the software product), or sales of similar software products from one company to another. One could consider the Cost Approach value and the software inventory value to be one in the same assuming all components were valued and biased by associated risk and influencing factors. Such would not be the case, however, if the "waterfall" development model was used in the valuation for determining the individual value of the components (see above reference).

Naturally, valuation of software is largely based on experts' assumptions and estimations. The objectiveness of these is sometimes questioned. That is the main reason for the difficulties of conservative accounting to cope with the recognition of such assets in financial reports. However, it is unrealistic to not recognize the role and governance of software assets in the functioning of enterprises in this information economy. ■

35. *Ibid*.
36. CEO of open-source firm SpikeSource.
37. *Ibid*.
38. *Ibid*.
39. EBiz "Black Duck Software Launches Open Source Code Center" 01/28/2008.
40. *Ibid*.
41. *Ibid*.