*Best Practices*

# The Valuation of Computer Software in the Health Care Industry

*John E. Elmore, JD, CPA*

*Taxpayers are often unaware of the fact that in many tax jurisdictions a portion, if not all, of the software incorporated in medical equipment and health-care-related information technology (IT) systems may be exempt from ad valorem property taxation. Under those circumstances, the property tax assessment should reflect a deduction for the value of the nontaxable software. This discussion presents generally accepted methods that valuation analysts may use to value health care industry computer software for property tax purposes.*

## Introduction

In many jurisdictions, the property tax is an "ad valorem" tax. That means that the taxpayer property is taxed according to the value of the property. Valuation analysts often assist in the taxation process by valuing the personal property subject to the tax. The taxation of computer software as personal property is a phenomenon of the modern era that may not fit easily within the traditional definitions of tangible personal property and intangible personal property.

Attempts by state tax authorities to address this issue has resulted in an incongruous collection of state-specific rules and methods by which valuation analysts and tax advisers contend for guidance in determining what portion of a taxpayer's computer software assets is taxable and what portion is tax-exempt. This discussion presents an overview of the valuation of computer software for property tax purposes, with an emphasis on the health care industry.

## Computer Software in the Health Care Industry

Computer software is revolutionizing health care. Advances in the delivery and efficacy of health care are driven mostly by advances in technology—technology that depends largely on software. Computer software is used in virtually all fields of medicine and throughout the health care industry.

Examples of the use of software in health care include the following:

1. Medical devices, including devices for diagnostics and monitoring
2. Surgical robots
3. Medical imaging systems
4. Telemedicine
5. Electronic medical records processing and storage
6. Medical diagnosis and expert systems
7. Nuclear medicine equipment
8. Radiation oncology and linear accelerator equipment
9. Pharmaceutical and biotechnology research, including drug discovery
10. Genetic testing and personalized medicine
11. Health care management information and billing systems
12. Health care analytics for epidemiology and population health management

In many cases, software operates on conventional computer servers and laptops. Many of the categories include specialized hardware devices, such as surgical robots and diagnostic machines. These devices tend to be more specialized than general-purpose computers and operate "embedded" software.

Embedded software is similar in most respects to other software, though traditionally embedded software was designed to operate under memory size and computer power constraints, often using specialized computer processors.

For example, in California, the Orange County Assessor's Office levied significant personal property taxes on Cardinal Health 301, Inc. (Cardinal), a manufacturer and lessor of medical equipment for automatically dispensing and tracking medicine—a kind of computerized "medicine storage cabinet."[1]

The Assessment Appeals Board noted that 90 percent of the assessed value of each unit leased was attributed to proprietary embedded software. Cardinal challenged the property tax assessment in court and won on the position that the embedded software did not constitute a taxable asset under California property tax law.

Taxpayers are often unaware of the fact that in many tax jurisdictions a portion, if not all, of the software incorporated in medical equipment and healthcare-related information technology (IT) systems is exempt from property tax. Under those circumstances, the property tax assessment should reflect a deduction for the value of the nontaxable software.

## Identification of the Computer Software Subject to Taxation

A few states assess property taxes on intangible personal property, including computer software. Virginia, for example, specifically defines "computer application software" as taxable intangible personal property.[2]

As a general rule, however, most state tax jurisdictions do not tax intangible personal property. Therefore, taxpayers have sought to avoid taxation of computer software by claiming that the programs and services of which it is composed constitute intangible property.

Three general lines of reasoning have been devised by state courts to determine whether computer software is tangible or intangible:

1. Whether one is purchasing a tangible storage medium versus the intangible knowledge contained therein
2. Whether software is an operational program or an application
3. Whether software is "custom" or "canned"

The first line of reasoning, which we may call the "container test," focuses on a substance-over-form inquiry involving two components:

1. A physical storage medium (e.g., a magnetic tape, compact disc, or digital versatile disc)
2. The knowledge/information contained on the medium

Intangible knowledge in this context refers to the abstract representation of human knowledge in the form of computer code, which instructs a microprocessor to perform computational tasks to manipulate and communicate this intangible knowledge.

Starting in the early years of computing, tax authorities sought to characterize software by the tangible form in which it was stored and distributed. The container test examines whether the intangible knowledge (that is, the computer code) contained within a tangible medium is a significant factor for tax purposes and whether the tangible medium may be considered merely incidental to the purchase of that intangible knowledge.

The container test appears increasingly outdated in today's computing environment, as the use of tangible storage mediums for software distribution has waned and software is routinely downloaded to computers directly or accessed on demand from servers in a cloud network.

For example, in 1996, the Texas Court of Appeals ruled that software was intangible property and, therefore, not subject to ad valorem taxation.[3]

The court said that the software was intangible because the "essence of the transaction" was not the tangible medium that was used to transport the software to the consumer (for example a disk or CD-ROM) but rather the software it contained.

"Computer application software," the court reasoned, is intangible personal property consisting of imperceivable binary pulses, programs, routines, and symbolic mathematical code that controls functioning of computer hardware and directs hardware operations; therefore, it was not subject to ad valorem taxation as tangible personal property.

A number of states have emphasized a second line of reasoning that focuses on how separable the software is from the computer hardware on which it operates. Some states insist that software is essentially inseparable from the tangible hardware on which it operates.

The Ohio Supreme Court, for example, upheld the Ohio Department of Taxation position that all software was taxable under the reasoning that the encoded instructions are always stored in some form of physical memory—a tangible medium—when operating in a computer.[4]

Therefore, in Ohio, the entire computer and all of the software operating thereon is taxable.

In other states, the issue of separability usually takes the form of classifying software either as:

1. operational software or
2. application software.

Operational software is generally required in order for the computer to function. Sometimes operational software is described as "embedded" software (or "firmware"). This is based on the fact that the software is encoded into memory chips attached directly to the circuit board of a computing device. Embedded software is often ascribed to specialized computing devices that lack many of the features and attachments associated with a general-purpose computer.

But even a general-purpose computer, like a laptop computer, contains embedded software in the form of a basic input output system (BIOS). BIOS is permanently stored in a memory chip on a computer motherboard (the primary circuit board). It is always and automatically executed when the computer is turned on.

It serves as the fundamental, real-time operating system (OS) for managing the microprocessor(s) on the motherboard and the peripheral devices that attach to the motherboard. For a laptop computer, these attached devices include a hard drive, a video graphics card, a network adapter, a keyboard, and a touchpad.

Depending on the tax jurisdiction, however, operational software may have a more expansive definition and include a general-purpose OS that works in conjunction with the BIOS.

The general-purpose OS is software typically stored on a larger memory medium, such as a disk drive, that is loaded and executed by the BIOS to provide a more sophisticated operating environment (e.g., graphical user interface, multitasking features). Two examples are:

1. the Linux operating system and
2. the Microsoft Windows operating system.

It is upon the foundation of the BIOS and the general-purpose OS that application software operates.

The Kansas Department of Revenue describes the distinction between operational software and application software as follows:

> The Kansas Supreme Court has held that software programs are taxable if they are operational programs; programs the computer cannot operate without. These programs are considered an essential portion of the computer hardware and are taxable as tangible personal property in conjunction with the hardware. On the other hand, application programs, which are particularized instructions, are intangible property, which is not subject to taxation in Kansas.

As a simple illustration, a laptop computer first executes a BIOS when it is turned on. This BIOS typically would be considered a tangible asset that is taxable. Once the laptop computer has booted up, a user may choose to execute an application such as Microsoft Office.

Office would qualify as tax-exempt application software because it executes "on top" of the BIOS and is not required for the computer to function (the laptop will operate normally regardless of whether Office is installed). The classification of the Windows OS, which also executes on top of the BIOS, as taxable operational software or tax-exempt application software can vary by tax jurisdiction.

This interplay of embedded operational software and general-purpose OSs can lead to complicated tax rules. Wisconsin statutory law exempts from property tax "mainframe computers, minicomputers, personal computers, networked personal computers, . . . electronic peripheral equipment, tape drives, [and] printers."[6]

The exemption does not apply to "equipment with embedded computerized components." In 2012, the Wisconsin Tax Appeals Commission rejected the property tax assessment of the City of La Crosse, Wisconsin, against a medical clinic on the grounds that the state's property tax law exempted medical equipment that connected to, and was controlled by, an external general-purpose computer.[7]

In that case, the taxpayer had reported its medical equipment as exempt in its personal property statements for the years at issue. The city tax assessor reclassified as taxable all the medical equipment except ultrasound and MRI equipment.

The Commission viewed the issue of taxability as whether the function of a medical device depended solely on an embedded OS, as opposed to being subject to control from a general-purpose OS executing on an externally attached computer.

The operational software/application software dichotomy offers a helpful guideline. But it is only a general guideline. Not all operational software is subject to property tax and not all application software is tax-exempt.

For example, California state law provides that the operational software must be preinstalled, or "bundled," on the computer equipment purchased or leased.[8]

Operational software that is not bundled generally is not subject to the property tax. Applications that are bundled with computer equipment are presumed by the California tax authorities to be subject to the property tax—a presumption that may be rebutted by a taxpayer with a sufficient evidentiary showing.

The third line of reasoning classifies computer software as either (1) software developed for internal use—"custom" software—or (2) software that is developed for commercialization (that is, for resale)—"canned" software.

Canned software typically includes software that is licensed to others and may be held by the developer as inventory. Under many state property tax statutes, custom software is taxed, while canned software is not.

An example of canned software is the Microsoft Office software suite. If company ABC purchases Microsoft Office along with a new laptop computer, the value of Microsoft Office ordinarily would not be included in the tax base (we are assuming the tax jurisdiction exempts "canned" software), while the value of the laptop computer would be included as a tangible asset.

This concept is fairly consistent with the operational software/application software dichotomy. The distinction in this line of reasoning becomes more evident if one considers that company ABC may be taxed on its laptop software if it instead builds a custom application with word-processing and other office-productivity features.

Taxability, under the third line of reasoning, hinges on the issue of customization, not on whether the software is application software.

In practice, the distinction between custom software and canned software sometimes can be difficult to discern. Classification problems arise when one considers the many ways in which software can be created, modified, and distributed. If a software developer is engaged to create software for a particular customer's specifications that will not be resold to others, it may be considered custom software.

But if the developer creates the software for a franchise chain and then licenses the software individually to 100 franchisees, some tax jurisdictions may classify the software as having been developed for commercialization even though the customers belong to the same franchise chain.

Another problem is reclassifying canned software as customized software. Canned software can be modified and/or incorporated into custom software, thereby changing its nature in the process. To what extent does modifying or incorporating canned software transform it into custom software for tax purposes?

There are no clear rules defining what constitutes customization. The Kentucky Department of Revenue recognized this problem, stating: "At present, there are no solutions to the problem of classifying software. Until such determination changes the classification of software, the Department classifies all software as tangible personal property."[9]

## Computer Software Valuation Approaches and Methods

There are several generally accepted methods used in the valuation of computer software. These methods can be categorized into the three generally accepted intangible asset valuation approaches:

1. The cost approach
2. The income approach
3. The market approach

The following discussion of these approaches summarizes the common methods employed by valuation analysts in valuing computer software for property tax purposes.

### Cost Approach

The cost approach is premised on valuing computer software based on some measure of cost. Two general types of cost may be estimated:

1. The reproduction cost new
2. The replacement cost new

The reproduction cost new reflects the cost to recreate the functionality of the subject computer software but in a form or appearance that may differ from the subject computer software. The replacement cost new typically establishes a maximum amount that an owner would pay for a fungible intangible asset.

However, specially developed computer software is often unique and may not qualify as a fungible intangible asset. In many cases, an intangible asset is less useful than its ideal replacement. The cost of the subject intangible asset should then be adjusted to reflect the loss in economic value due to functional, technological, and economic obsolescence.

Under the cost approach, three methods that may be used to provide a cost indicator for computer software are as follows:

1. The trended historical cost method
2. The estimated historical cost method
3. The software engineering cost estimation model method

## The Trended Historical Cost Method

In this method, actual historical computer software development costs are identified and quantified and then "trended" through the valuation date by an appropriate inflation-based index factor. The valuation analyst ordinarily should include all costs associated with the development of the subject computer software. An allocation of taxpayer company overhead costs and the cost of employee fringe benefits ordinarily should be included in addition to employee payroll costs if the taxpayer company personnel are employed in tasks related to the software development.

Historical costs ordinarily should include an allowance for the software developer's profit on the software development project, an allowance for entrepreneurial incentive to motivate the software development project, all direct development costs such as salaries and wages, and all indirect development costs, such as taxpayer company overhead and employment taxes/employee benefits.

The application of the trended historical cost method typically estimates the reproduction cost new of the subject computer software. In many cases, due to technological advances in programming languages or programming tools, for example, the replacement cost new for software may be lower than the reproduction cost new for the subject taxpayer software.

## The Estimated Historical Cost Method

Sometimes historical development costs are not readily available. In this case, software development costs can be estimated using actual or estimated software development time (person hours, person months, and so on). The development cost estimate is computed by multiplying the development time by an associated cost metric using specific costs per software development person or a weighted average cost for the software development team. This cost is typically a full absorption cost.

As with the trended historical cost method, the valuation analyst should consider all relevant costs related to the software development as well as allowances for the software developer's profit and for entrepreneurial incentive.

## The Software Engineering Cost Estimation Model Method

The valuation analyst may employ software engineering models in order to estimate either the reproduction cost new or the replacement cost new of the taxpayer company's computer software. Generally, the software engineering models were originally developed to assist software developers in estimating the effort time and human resources needed to complete a software project. These models have been adapted by valuation analysts for computer software valuation purposes.

The primary input to the software engineering models is a size-related metric. Capers Jones, a pioneering authority in the field of software cost estimation, observed: "Every form of estimation and every commercial software cost-estimating tool needs the sizes of key deliverables in order to complete an estimate."[10]

Jones lists six types of sizing:

1. Sizing based on lines of code
2. Sizing by extrapolation from function point analysis
3. Sizing by analogy with similar products of known size
4. Guessing at the size using "project manager's intuition"
5. Guessing at the size using "programmer's intuition"
6. Sizing using statistical methods or Monte Carlo simulation[11]

Historically, the most common sizing metric has been the number of lines of code. The definition of a line of code and the associated line of code counting conventions vary among the common software engineering models. A line of code can be defined as source code instructions (i.e., instructions as written by human programmers) or object code instructions (what the computer produces after it has compiled, or translated, the source code into instructions the computer can more directly process).

Lines of code have meaning only within the context of the computer language being employed. Languages have evolved over time and can be classified into generations. As a general observation, higher-generation languages require less source code to perform the same tasks than lower-generation languages.

Source code written in assembly language—a second generation language—typically requires more source code instructions to perform a given set of tasks than third generation languages such as C, C++, and Java. And, source code written in a third generation language typically requires more source code instructions to perform a given set of tasks than fourth generation languages such as Python or Ruby.

To illustrate, Figure 1 presents the source code to display the words "Hello, world" in (1) assembly (a second generation language) and (2) Python (a fourth generation language). The valuation analyst should use software engineering models that account for language differences in estimating cost.

In an effort to address the deficiencies in the use of simplistic lines-of-code metrics, function-related metrics were developed to measure software development effort. The most common of these metrics is function points.

The number of function points in a computer program is often calculated with an algorithm that uses a weighted count of the number of inputs, outputs, user interactions/inquiries, data files, and external interfaces. The function point count is modified by the complexity of the development project.

Function point counts are sometimes used by software engineering models to estimate the number of lines of code based on an average number of lines of code established per function point for a given language. The discipline of function point analysis has evolved over time and has been standardized to a large extent by the International Function Point Users' Group.

Other inputs to the software engineering models include attributes such as: programming language experience and quality of the project team, software development tools used, programming practices, complexity type of application, time constraints, level of system documentation, and required program reliability.

Presently, three of the most commonly used algorithmic software cost estimation models are the following:

1. The Constructive Cost Model (COCOMO) and its derivatives
2. The KnowledgePLAN model
3. The Software Lifecycle Management (SLIM) model

These software cost estimation models are considered "algorithmic" models because they generate cost estimates using a set of quantified inputs, such as lines of source code, which is processed automatically in accordance with metrics and formulas derived from the empirical analysis of large databases of actual software projects.

Typically, the cost estimation models calculate an estimate of the effort required to develop a software system in terms of person-months. The number of person-months is multiplied by a blended cost per person-month to arrive at the indicated value of the computer software.

The blended cost per person-month is typically a full absorption cost (e.g., the cost of a software programmer would include benefits as well as wages).

**Figure 1**
**Comparison of the Number of Lines of Source Code to Display "Hello, World"**

```
.text
.global _start
_start:

mov $4, %eax /* write system call */
mov $1, %ebx /* stdout */
mov $msg, %ecx
mov $msgend-msg, %edx
int $0x80

mov $1, %eax /* _exit system call */
mov $0, %ebx /* EXIT_SUCCESS */
int $0x80

.data
msg: .ascii "Hello, world\n"
msgend:
```

**(a) Assembly (2nd Gen.) —14 lines of code**

```
print "Hello, world"
```

**(b) Python (4th Gen.)—1 line of code**

## *COCOMO*

The first generation of COCOMO was developed in the 1980s.[12]

The software cost estimation methods estimate the amount of effort in person-months required to develop software, taking into consideration the size of the developed programs (particularly in lines of code), the program characteristics, and the environment in which they are developed.

The basic software development equation defined by the COCOMO II model is as follows:

$$PM = a(KLOC)^{b} \times EM$$

where:

| | | |
|---|---|---|
| $PM$ | = | Person-months |
| $KLOC$ | = | Thousands of delivered lines of code |
| $a$ | = | Coefficient dependent on the class of project (organic, semi-detached, embedded) |
| $b$ | = | Scaling exponent |
| $EM$ | = | Effort multiplier |

A more updated model, COCOMO II, was developed by researchers at the University of Southern California (USC).[13]

The updated model supports the cost estimation of a variety of third and fourth generation language-based projects. It also incorporates function point analysis. An online estimation tool encompassing the COCOMO II model is available through the USC Center for Systems and Software engineering website.[14]

We provide an illustration of a cost approach valuation analysis using COCOMO II, as described later in this discussion.

A third model, COCOMO III, is being developed by USC and its project partners with the aim of improving the model with new and updated software cost drivers and new development paradigms.

The COCOMO III project purpose statement indicates that this model will be more attuned to the increasingly diverse use of computer software in the health care environment, including software in biomedical devices (both as embedded systems and mobile devices) and "Big Data" health management analytics.[15]

## *KnowledgePLAN*

KnowledgePLAN (KPLAN) is a proprietary function point-driven model that incorporates a historical knowledge base of project data derived from over 11,000 software projects that have been collected and researched by Software Productivity Research, LLC (SPR).[16]

The particular algorithms utilized by KPLAN have not been fully disclosed. The model uses a base of functional metrics to derive predictive/analytical productivity rates given a large number of known (or assumed) parameters. Projects are classified by, among other things, scope (e.g., program or application, sub-system), topology, (e.g., standalone, client/server), class (e.g., end-user developed, IT developed), and type (e.g., interactive graphical user interface, multimedia).

The size of the system can be expressed in several ways, including function points or lines of code, by language. The valuation analyst assigns attribute values that describe the personnel, technology, process, environment, and product.

KPLAN was updated in 2011 with the release of version 4.4, but SPR appears to have ceased support for the software cost estimation tool. The tool is still available for download from various software archive websites.

## *SLIM*

The SLIM software engineering model was developed by Lawrence Putnam, the founder of Quantitative Software Management, Inc. (QSM). QSM licenses software cost estimation tools incorporating the model. The SLIM model (also referred to by commentators and in academic literature as the "Putnam model") estimates the amount of effort in person-months required to develop software based on the following:

1. A manpower build-up parameter (a number representing a range from entirely new software to rebuilt software)
2. The software delivery time
3. A productivity environment factor

The SLIM model was developed using a knowledge base of project data derived from over 6,000 software projects that have been collected and researched by QSM.

The main equation for the SLIM model is:

$$PY = \left[\frac{KLOC}{PROD \times TIME^{4/3}}\right]^{3} \times B$$

where:

| | | |
|---|---|---|
| $PY$ | = | Person-years |
| $KLOC$ | = | Thousands of delivered lines of code |
| $PROD$ | = | Productivity environment factor |
| $TIME$ | = | Software delivery time |
| $B$ | = | Manpower build-up parameter |

## Obsolescence

In computer software valuation under the cost approach, the valuation analyst ordinarily should consider all relevant forms of obsolescence. When the subject computer software is less useful than its ideal replacement, its cost should be adjusted to reflect a loss due to the following types of obsolescence:

1. Functional,
2. Technological
3. Economic

A fourth form of obsolescence, physical deterioration, is not generally applicable to computer software, as software typically does not experience physical wear and tear.

Functional obsolescence is the loss in value of an intangible asset because the subject intangible asset does not have the functionality of—or is less useful than—a replacement intangible asset. In the case of computer software, functional obsolescence is often mitigated when the subject software is continually maintained.

Technological obsolescence is often considered to be a particular component of functional obsolescence. It is the loss in value of intangible asset to two technological improvements that make the replacement intangible asset more efficient or effective than the subject intangible asset. In the valuation of computer software, technological obsolescence usually exists when:

1. the subject computer software is written in an inefficient or outdated language or
2. runs on a platform (hardware, operating system, and so on) that is becoming obsolete (and the software is not portable).

Technological obsolescence may also exist if the outdated models or practices of the developers result in a less-than-optimal use of resources.

Economic obsolescence is a reduction in the value of the subject computer software due to events that are typically outside of the control of the computer software owner/operator. Such events may include legal or regulatory changes or restrictions, or market conditions (for example, new competitors).

Economic obsolescence may be an important issue in the valuation of software developed for resale. Economic obsolescence is generally not very evident with regard to internally developed operational computer software that is being used by a financially successful taxpayer company.

## Income Approach

In the income approach, the value of computer software is estimated as the present value of the future economic income attributable to the ownership of the computer software over its expected remaining useful life (RUL). This economic income may result from prospective (1) revenue, (2) cost savings, or (3) royalty or license income associated with the computer software.

The income approach methods used in the valuation of computer software include the following:

1. The yield capitalization (or “yield cap”) method
2. The direct capitalization method

The discounted cash flow (DCF) analysis is a common yield capitalization valuation method.

The yield cap method, and in particular the DCF analysis, is typically used in the valuation of computer software when there is an identifiable income stream associated with the subject software.

Therefore, this method is often used in the valuation of product software or databases that generate income through their sale or license. The future cash flow related to such product software, for example, may be estimated by projecting revenue, expenses (excluding depreciation and amortization expense), and capital investments over the software estimated remaining useful life (RUL). The future cash flow projection is discounted to a present value using an appropriate present value discount rate.

## Market Approach

In the market approach, the value of computer software is estimated by reference to actual market sale or license transactions involving comparable or guideline software systems. This valuation approach may be difficult to use in the valuation of internally developed software.

The relief from royalty valuation method is used to estimate the cost savings that accrue to the taxpayer company owner/operator of the computer software. This valuation method assumes that the taxpayer owner/operator would otherwise have to pay a royalty or license fee on the revenue earned through use of the subject software.

The royalty rate used in the valuation analysis is based on an analysis of empirical, market-derived royalty rates for comparable or guideline computer software systems.

In the case of product software, a product revenue is projected over the expected RUL of the subject computer software. The market-derived royalty rate is then applied to estimate the royalty savings. The net after-tax royalty savings are calculated for each year in the RUL of the subject computer software. The net after-tax royalty savings are then discounted to a present value, as with the yield cap method.

Another market approach method used to value computer software is the market transaction method.[18] Under the market transaction method, where arm's-length market transaction data are available for comparable or guideline computer software, the implied value is typically expressed as a dollars-per-line-of-code or dollars-per-function-point figure.

This value per unit is then applied to the subject taxpayer company software lines of code (or function points) to estimate the value of the subject software. As with any valuation method that relies on comparable or guideline intangible assets, adjustments should be made for material differences between (1) comparable or guideline computer software and (2) the subject computer software.

A simple example of the market transaction method is presented in Exhibit 6.

## Remaining Useful Life

Remaining useful life reflects the period during which the subject computer software is expected to contribute directly or indirectly to the owner's or licensee's future cash flow. It reflects the economic useful life and may differ from other measures of useful life, such as the amortization period for financial reporting purposes under generally accepted accounting principles (GAAP).

According to the Financial Accounting Standards Board (FASB) Accounting Standards Codification (ASC) topic 350-40-35-5, "Given the history of rapid changes in technology, software often has had a relatively short useful life."

In some instances, software that has been fully amortized under GAAP—based on expectations of a short useful life—may still be in use. It is not uncommon in taxpayer companies for software systems that were initially developed 20 to 30 years ago to remain in current use.

The estimation of the RUL may be an important consideration in each of the three generally accepted approaches to computer software valuation.

In the income approach, an RUL analysis may be performed in order to estimate the projection period for the prospective computer software economic income. In the cost approach, an RUL analysis may be performed in order to estimate the total amount of obsolescence, if any, from the estimated measure of cost.

In the market approach, an RUL analysis may be performed in order to:

1. select or reject comparable or guideline software license or sale transactions and/or
2. make adjustments to the comparable or guideline software sale and/or license transactional data.

## Valuation Example

Exhibits 1 through 6 of this discussion present an example of a computer software valuation analysis. The results of the three methods are synthesized and presented in Exhibit 1.

Our example focuses on the fictional AlphaMed Company (AlphaMed), which performs medical diagnostic services and toxicology drug testing. Let us suppose the fair market value of the AlphaMed medical diagnostic and testing equipment (the "subject equipment"), as of the valuation date (January 1, 2016), has been estimated as $16.0 million. This value is inclusive of any software associated with the subject equipment.

Under the applicable local and state tax laws and guidelines, the software component of the subject equipment (the "subject computer software") qualifies as a tax-exempt intangible asset. AlphaMed has hired the valuation analyst to estimate the fair market value of the subject computer software.

### Cost Approach—Replacement Cost New less Depreciation Method

For simplicity, let's assume the following;

1. The replacement cost new is estimated using the average results of two software engineering cost estimation models: COCOCO II and SLIM.
2. The line-of-code counts and other model inputs are as presented in Exhibits 2 through 4.
3. The average of the COCOMO II and SLIM efforts is multiplied by the obsolescence factor, where applicable, to arrive at the adjusted effort in person-months.
4. The analyst determined that the blended development cost per person-month was $8,600.
5. The analyst applied the blended development cost to the total adjusted effort in person-months to arrive at the total development costs.
6. The analyst applied a 10 percent developer's profit and a 15 percent entrepreneurial incentive to reflect the profit motive and opportunity cost associated with developing the AlphaMed software.

This method results in an indicated value estimate of the subject computer software of $8.3 million as presented in Exhibit 4.

## Market Approach—Relief from Royalty Method

Let's assume the following additional facts related to the AlphaMed software:

1. Next year projected revenue attributed to the sale of medical diagnostic and testing services using the software is $45 million.
2. The annual revenue growth rate is 5 percent.
3. The market-derived royalty rate is 8 percent.
4. The effective company income tax rate is 40 percent.
5. The expected RUL of the software (until replacement or retirement) is five years.

This method results in an indicated value estimate of the subject computer software of $8.5 million, as presented in Exhibit 5.

## Market Approach—Market Transaction Method

Let's assume the following additional facts related to the AlphaMed software:

1. The analyst estimates the total number of LOC as 570,000.
2. The comparable arm's-length software sale/licensing transactions were identified, yielding a sale transaction price per LOC.
3. The indicated price range is between $12.60 per LOC and $18.50 per LOC.
4. The range of indicated values for the subject software code is calculated as the market-derived price per LOC times the total number of LOC.

This method results in an indicated value estimate of the subject computer software of $8.9 million, as presented in Exhibit 6.

## Valuation Synthesis and Conclusion

As presented in Exhibit 1, the three methods were provided an equal weighting.

The fair market value of the subject computer software, based on the valuation analysis described herein, as of the valuation date (January 1, 2016), is $8,540,000.

## Effect on the Property Tax Assessment

The fair market value of the subject equipment was estimated as $16.0 million. However, this fair market value estimate incorporated the value of the subject computer software.

As presented in Exhibit 1, the estimated fair market value of the subject computer software was $8.5 million as of the valuation date. Subtracting the value of the subject computer software yields a fair market value of $7.5 million ($16.0 million less $8.5 million) for the taxable portion of the subject equipment.

Therefore, the computer software valuation analysis resulted in properly reducing the AlphaMed property taxes on the subject equipment by more than 50 percent.

Notes:

1. See Cardinal Health 301, Inc., v. County of Orange, 167 Cal.App.4th 219 (2008).
2. Rulings of the Tax Commissioner, Document 13-47, Virginia Department of Taxation, available at http://www.tax.virginia.gov/laws-rules-decisions/rulings-tax-commissioner/13-47.
3. See Dallas Cent. Appraisal Dist. v. Tech Data, 930 S.W.2d 119 (Tex.App.-Dallas 1996, pet. denied).
4. See Andrew Jergens Company v. Wilkins, Tax Commr., 848 N.E.2d 499 (Ohio 2006).
5. "2016 Personal Property Valuation Guide," Kansas Department of Revenue, available at http://www.ksrevenue.org/pdf/PPVG.pdf.
6. See Wisconsin Statute §70.11(39).
7. See City of La Crosse v. Wisconsin Department of Revenue and Gundersen Clinic, Ltd., [2 Wis.] St. Tax Rep. (CCH) paragraphs 401-589 (Wis. Tax App. Commission June 8, 2012, incorporating June 9, 2008 ruling), aff'd id. paragraphs 401-658 (Wis. Cir. Ct. Dane County Dec. 7, 2012).
8. As discussed in Cardinal Health v. County of Orange, 167 Cal.App.4th 219, 222 (2008).
9. Kentucky Department of Revenue, Audit Manual, 2007.
10. Capers Jones, *Estimating Software Costs: Bringing Realism to Estimating*, 2d ed., (New York: McGraw-Hill, 2007), 8.
11. Ibid., 9.
12. For a detailed description of COCOMO, see Barry W. Boehm, *Software Engineering Economics* (New York: Prentice-Hall, 1981).
13. For a detailed description of COCOMO II, see Boehm et al., *Software Cost Estimation with COCOMO II* (New York: Prentice-Hall PTR, 2000).
14. See http://csse.usc.edu/research/COCOMOII/.
15. See http://www.cocomo3.com/about/.
16. KPLAN is described in a number of publications by Capers Jones. See note 6.
17. More detailed information about the SLIM model is available from the QSM website, http://www.qsm.com.
18. The market transaction method is often described in valuation literature as the comparable sales method, the comparable transaction method, or the like. See, e.g., James A. Amdur, "Telecommunications Property Taxation," *Federal Communications Law Journal* 46, no.2 (1994): 231.
19. Ibid.: 232.

*John Elmore is a vice president in our Atlanta practice office. John can be reached at (404) 475-2303 or at jeelmore@willamette.com.*

**Exhibit 1**
**AlphaMed Company**
**Valuation Synthesis and Conclusion**
**As of January 1, 2016**

| Valuation Approach and Method | Indicated Value | Relative Emphasis | Concluded Value | Reference |
|---|---|---|---|---|
| Cost Approach—Replacement Cost New less Depreciation Method | $ 8,290,000 | 1/3 | $ 2,763,333 | Exhibit 4 |
| Market Approach—Relief from Royalty Method | 8,470,000 | 1/3 | 2,823,333 | Exhibit 5 |
| Market Approach—Market Transaction Method | 8,860,000 | 1/3 | 2,953,333 | Exhibit 6 |
| **Fair Market Value of Subject Computer Software (rounded)** | | | **$ 8,540,000** | |

**Exhibit 2**
**AlphaMed Company**
**Cost Approach**
**COCOMO II Variables—Scaling Exponent**
**As of January 1, 2016**

| | | | Rating | Scale Factor |
|---|---|---|---|---|
| **Scale Factors:** | | | | |
| | PREC | Precedentedness | High | 2.48 |
| | FLEX | Development Flexibility | High | 2.03 |
| | RESL | Architecture/Risk Resolution | Nominal | 4.24 |
| | TEAM | Team Cohesion | High | 2.19 |
| | PMAT | Process Maturity | Nominal | 4.68 |
| | | | **Sum of the Scale Factors** | **15.62** |

**Scaling Exponent (b) = 0.91 + 0.01 x 15.62 = 1.07**

**Exhibit 3**
**AlphaMed Company**
**Cost Approach**
**COCOMO II Variables—Effort Multiplier**
**As of January 1, 2016**

| | | | Rating | | Multiplier |
|---|---|---|---|---|---|
| **Product Factors:** | | | | | |
| | RELY | Required System Reliability | Very High | | 1.28 |
| | DATA | Data Base Size | Nominal | | 1.00 |
| | CPLX | Software System Complexity: | | | |
| | | *Complexity-Control Operations* | *Nominal* | *1.00* | |
| | | *Complexity-Computational Operations* | *High* | *1.20* | |
| | | *Complexity-Device-Dependent Operations* | *Nominal* | *1.00* | |
| | | *Complexity-Sensor Operations* | *High* | *1.17* | |
| | | *Complexity-Data Management Operations* | *Nominal* | *1.00* | |
| | | *Complexity-User Interface* | *Nominal* | *1.00* | |
| | | Average | | | 1.06 |
| | RUSE | Required Reusability | Low | | 0.75 |
| | DOCU | Documentation Match to Life Cycle Needs | Nominal | | 1.00 |
| **Computer Factors:** | | | | | |
| | TIME | Execution Time Constraint | High | | 1.09 |
| | STOR | Storage Restraint | Very High | | 1.32 |
| | PVOL | Platform Volatility | Low | | 0.87 |
| **Personnel Factors:** | | | | | |
| | ACAP | Analyst Capability | High | | 0.80 |
| | PCAP | Personal Continuity | High | | 0.87 |
| | PCON | Applications Experience | High | | 0.91 |
| | APEX | Applications Experience | Very High | | 0.95 |
| | PLEX | Platform Experience | Nominal | | 1.00 |
| | LTEX | Language and Tool Experience | Nominal | | 1.00 |
| **Project Factors:** | | | | | |
| | TOOL | Use of Software Tools | Nominal | | 1.00 |
| | SITE | Multistate Development Site Collocation | Nominal | | 1.00 |
| | SCED | Required Development Schedule | Nominal | | 1.00 |
| | | | **Product of the Effort Multipliers** | | **0.77** |

**Combined Effort Multiplier = 0.77**

## Exhibit 4
## AlphaMed Company
## Cost Approach
## Replacement Cost New less Depreciation Method
## Computer Software Valuation Summary
## As of January 1, 2016

| Software Program | Total Physical LOC | Physical Executable LOC [a] | Logical Executable LOC [b] | COCOMO II Effort in Person-Months [c] | SLIM Effort in Person-Months [d] | Method Average Effort in Person-Months | Obsolescence Adjustment [e] | Adjusted Effort in Person-Months [f] |
|---|---|---|---|---|---|---|---|---|
| Program 1 | 155,000 | 131,750 | 98,813 | 302.25 | 232.73 | 267.49 | 0% | 267.49 |
| Program 2 | 75,000 | 63,750 | 47,813 | 139.39 | 160.30 | 149.84 | 100% | - |
| Program 3 | 160,000 | 136,000 | 102,000 | 312.66 | 343.92 | 328.29 | 10% | 295.46 |
| Program 4 | 180,000 | 153,000 | 114,750 | 354.49 | 177.25 | 265.87 | 25% | 199.40 |
| Total | 570,000 | 484,500 | 363,375 | 1,108.79 | 914.20 | 1,011.50 | | 762.36 |
| Times: Blended Development Cost per Person-Month [g] | | | | | | | | $ 8,600 |
| Equals: Development Cost (net of obsolescence) | | | | | | | | 6,556,280 |
| Add: Developer's Profit (10 percent) [h] | | | | | | | | 655,628 |
| Equals: Total Development Cost with Developer's Profit | | | | | | | | 7,211,908 |
| Add: Entrepreneurial Incentive (15 percent) [h] | | | | | | | | 1,081,786 |
| **Indicated Value of Subject Computer Software (rounded)** | | | | | | | | **$ 8,290,000** |

LOC = Lines of code
[a] The analyst used a source to executable line-of-code reduction percentage of 15 percent.
[b] The analyst used a physical to logical line-of-code reduction percentage of 25 percent.
[c] Based on coefficient $a$ = 2.94, scaling exponent $b$ as presented on Exhibit 2, and effort multiplier as presented in Exhibit 3.
[d] Derived by the analyst using a SLIM valuation tool (details not presented).
[e] An obsolescence adjustment was applied by the analyst for programs that have been scheduled for replacement/update or retirement based on the RUL and age of each of the programs.
[f] Calculated as the average effort in person-months multiplied by the obsolescence adjustment.
[g] Calculated as a blended rate based on the full absorption cost of employees and contractors involved in development.
[h] Analyst estimate.

## Exhibit 5
## AlphaMed Company
## Market Approach
## Relief from Royalty Method
## Computer Software Valuation Summary
## As of January 1, 2016

| | Year 1 | Year 2 | Year 3 | Year 4 | Year 5 |
|---|---|---|---|---|---|
| Software-Dependent Sales of Diagnostic Services | $ 45,000,000 | $ 47,250,000 | $ 49,612,500 | $ 52,093,125 | $ 54,697,781 |
| Multiplied by: Royalty Rate | 8.0% | 8.0% | 8.0% | 8.0% | 8.0% |
| Equals: Gross Royalty Savings | 3,600,000 | 3,780,000 | 3,969,000 | 4,167,450 | 4,375,823 |
| Less: Income Tax (at 40%) | (1,440,000) | (1,512,000) | (1,587,600) | (1,666,980) | (1,750,329) |
| Equals: Net Royalty Savings | 2,160,000 | 2,268,000 | 2,381,400 | 2,500,470 | 2,625,494 |
| Periods Discounted | 0.5 | 1.5 | 2.5 | 3.5 | 4.5 |
| Multiplied by: Present Value Interest Factor (at 15%) | 0.933 | 0.811 | 0.705 | 0.613 | 0.533 |
| Equals: Present Value of Net Royalty Savings | $ 2,015,280 | $ 1,839,348 | $ 1,678,887 | $ 1,532,788 | $ 1,399,388 |
| **Indicated Value of Subject Computer Software (rounded)** | **$ 8,470,000** | | | | |

## Exhibit 6
## AlphaMed Company
## Market Approach
## Market Transaction Method
## Computer Software Valuation Summary
## As of January 1, 2016

| Valuation Variables | Number of LOC | Sale Transaction Price | Sale Transaction Price per LOC |
|---|---|---|---|
| Comparable Software Sale/Licensing Transaction 1 | 408,700 | $ 7,560,950 | $ 18.50 |
| Comparable Software Sale/Licensing Transaction 2 | 587,020 | 8,394,386 | 14.30 |
| Comparable Software Sale/Licensing Transaction 3 | 362,892 | 4,572,439 | 12.60 |

| Valuation Analysis | Low End of Indicated Value Range | High End of Indicated Value Range |
|---|---|---|
| Subject Computer Software Total Number of LOC | 570,000 | 570,000 |
| Multiplied by: Market-Derived Price per LOC | $ 12.60 | $ 18.50 |
| Equals: Indicated Value of Subject Computer Software | $ 7,182,000 | $ 10,545,000 |
| **Indicated Value of Subject Computer Software (rounded) [a]** | | **$ 8,860,000** |

LOC = Line(s) of code
Note:
[a] Based on the average of the low and high end ranges.