VALUING COMPUTER SOFTWARE AND SOFTWARE COMPANIES

NACVA Learning Institute, December 4, 2003, San Diego, California, USA
James P. Catty and Dita Vadron

PART I - INTRODUCTION TO SOFTWARE

Software Is Everywhere

To begin with the crassest possible announcement: There's gold in them thare chips - microprocessor chips, that is, and one can still make a darn good living not only by creating high tech companies, but also by valuing them. In both instances, you've got to know what you are doing and take into account the characteristics and possibilities of the product. It is a vast field, encompassing many different disciplines and philosophies.

The Western world, and quite a few of the Third World countries would find it very difficult to exist without all the sweeping electronic developments that have taken place in the last twenty years; we seem to be tied to them "for better or worse", a phrase that may ring a bell with many of us in a different context.

There's much in software that's good, but sometimes there seems to be just about as much that's bad. Software enables airlines to crowd us in like sardines calling it "yield management" and presenting it as something for everyone's benefit; it gives us mobile telephones, which makes it impossible to hide anywhere - be it from agitated clients or bill collectors, and it clears checks within 24 hours, no matter where they were issued; in the good old days, that process used to take two, sometimes three weeks or more. Long gone is the time when credit card charges from Europe or Asia needed at least six weeks to show up on your statement.

But software is also a key factor in many "medical miracles". We all know someone with a pacemaker, and have read or heard about chip-controlled artificial limbs or something equally astounding, which today is practically taken for granted.

On a lesser scale, software turns off the coffee maker when the brew is just right, gives us instant replays during sports events and keeps supermarket shelves stacked with the goods we want. It lets companies or individuals share documents and ideas around the globe, allows us to watch Marlene Dietrich in a tailor-made commercial years after she died, and lets pigs play video games; those were developed at the University of Guelph in Ontario and cause them to gain weight a lot faster. It's a revolution alright, and we've got to get used to it as rapidly as our little grey cells can manage.

Background

Actually, this will be the third major revolution since 1776, when Massachusetts and a few likeminded colonies declared their independence from Britain. It started with the "Steam Age", which lasted well over a hundred years; the first passenger steam railroad started service in Northern England during 1825, followed in 1826 by the first US line out of Baltimore.

Steam also made manufacture independent of water power; it allowed plants to be built away from the rivers, which, till the middle of the 19th century, had been the only economically feasible way to transport goods. The English economy soared, driven by a new technology rather than by the discovery of mines, as in South Africa, or by a new crop, as in the West Indies.

The new invention culminated in superb national railroad systems. Aided by Samuel Morse's invention of the telegraph in 1835, between 1850 and 1860, railroad tracks in the US increased from 9,000 to 30,000 miles, and - voilà - the Midwest was connected with the East coast. For the next fifty years, the railroads dominated the economy: freight ton miles rose 9,700% and employment expanded by 2,300%, a not unreasonable parallel to the explosive growth of the Internet.

The second, the "Machine Age", brought us two dominant innovations, the automobile and electricity. This led to the rail links being rapidly supplanted by the world's most extensive highway network, giving us a distribution system which has no equal anywhere in the world.

The third is happening now, the "Digital Age" of software and the Internet.

All this means that mankind has had to change and adjust its thin-king on a number of previous occasions. We made it then, and we will likely make it now. However, there is one vast difference: the two previous revolutions did not have to contend with a rapacious government, the IRS, aggressive regulatory authorities and far-flung, well-informed shareholders.

We do! That's where the savvy valuation analyst comes in, and that is what we will be talking about today.

Factors of Production

The three traditional factors of production are Capital, Labor and Land (resources). The "Digital Age" has added "Information" to those three pillars. According to Business Week, August 31, 1998:

> "The information revolution will continue to boost productivity across the economy. Over the next ten years, such information-dependent industries as finance, media, and wholesale and retail trade are expected to change the most. Increasing globalization will

> simultaneously provide much larger markets and tough foreign competitors. The result: companies must become even more innovative while cutting costs."

In my opinion, the human brain is still unmatched by anything called "ARTIFICIAL INTELLIGENCE", but processing large amounts of information is best done by computers. As no computer functions without software, this means an increasing demand for state-of-the-art products.

Although no longer as prevalent as a few years ago, some garden shed wunderkind could be turning out new and exciting stuff, which he believes to be worth a cool million or more to be shelled out by eager investors. In reality, his discovery may be worth a tenth of that, or even nothing at all; it may already exist, if not here, maybe elsewhere. And if it doesn't, there is the possibility that nobody wants it.

In this context we always mention a well-regarded real estate company that assured us their software had no competitors - none. We stopped our research after we had found 43, not in their country, but in Europe and Asia; as a result, their vast investment turned out to be a total write-off. We encountered that situation on not just one occasion, but on several.

To drive the point home, I would like to amend the phrase "caveat emptor" into "caveat valuator"; in other words, do your homework. In preparing this presentation, I have relied not only on my experience of sixteen years valuing software, but also on many published and some unpublished sources and studies.

What Is Computer Software?

That question brands you as neither uneducated, irrational or behind the times. Aside from ever increasing demand for speed supplied by constantly improved software, without such software, a computer is totally ignorant. The comment by Lady Ada Lovelace, Lord Byron's daughter, who financed Charles Babbage's mechanical predecessor of the computer, still applies:

> "The Analytical Engine has no pretensions whatever to originate anything. It can only do whatever we know how to order it to perform."

That was written in 1843, one hundred and two years before ENIAC, the first successful electronic computer, which, together with its elementary software, took up the whole gymnasium of the University of Pennsylvania.

According to Webster's Dictionary, software is:

> "Both the precise sequence of instructions that enable a computer to undertake a particular activity and the writ-ten code, flow charts, sub-routines, objects, languages, procedures, documentation, data, etc. that are used to prepare it".

Some of you may remember when spreadsheets and Financial Projections were done by hand; using a slide rule was considered progressive. Now, the Pentium on my desk, which takes care of all that and then some, has more power than the whole mainframe at the Wall Street investment bank where I used to work over four decades ago.

To put it simply, software instructs a computer what to do, how to do it, and how fast. Computers are expected to become even faster and cheaper in the future as they continue to adhere to "Moore's Law". In 1965, Gordon Moore, Co-founder of Intel, stated that:

> "The cost of computing power drops roughly 30% every year, and microprocessors double in power and speed every eighteen months."

At some time in the future, this Law is forecast to come up against the Laws of Physics, but so far, Moore's on track.

Types of Software

The IRS has its own views, perhaps interpreting Webster to suit its philosophy; Rev.Proc.69-21 defines computer software as:

> "All programs or routines used to cause a computer to perform a desired task or set of tasks, and the documentation required to describe and maintain those programs. Computer programs of all classes, for example, operating systems, monitors, compilers, and translator assembly routines, and utility programs, as well as application programs are included. 'Computer software' does not include procedures which are external computer operations, such as instructions to transcription operators and external control procedures."

There are two types of software: Systems and Application.

Systems Software allows a computer to function. It includes Operating Systems like Windows or UNIX, as well as service and utility functions that handle activities such as: managing, sorting, merging and converting data, system accounting, diagnostics, performance measurement, report generation and security control.

It is highly unlikely that you will ever have to value a systems software program on its own, as there are only about 70 varieties in general use; normally, except for depreciation purposes, it is grouped with the related hardware. Therefore, we will not discuss it further.

Application Software supplies instructions for computers to carry out specific functions related to the management, storage and pro-cessing of data. In our profession, we all use it, for accounting, spread sheets and word-processing.

This presentation deals with some of the approaches and problems of valuing application software and the organizations that create it.

PART II - APPLICATION SOFTWARE

Categories of Application Software

There are five main categories of application software; they are based on different technologies and serve distinct markets.

Enterprise: Products that control business processes and activities; they may serve a single vertical market (such as financial organizations), or supply a function (like accounting) to many industries.

Packaged: Programs that run on personal computers or servers; normally used to improve individual productivity, such as word processors, spread sheets and personal information managers.

Technical: Systems that assist the design and production of items ranging from food formulations to mechanical devices, computer chips and even other software.

Edutainment: Programs, usually running on PCs, that offer entertainment or education, mainly oriented to the under-twenty crowd. The software for the video game created to encourage pigs to eat more and gain weight faster certainly falls into that category.

Internet/E-Commerce: Software used for accessing the Internet, transmitting information between participants and entering into business transactions. This category is almost totally integrated with services, be they from a telephone company, an ISP (Internet Service Provider), a computer reservations system or a bank ATM (Automated Teller Machine).

Application Software is Different from Everything Else

Today, society in nearly every country is dependent on computers. It is unthinkable to function without software, which is effectively a capital item, but often expensed on purchase.

- The market life of software is limited; generally, investors and tax authorities will accept two years. However, established programs are often enhanced to prolong their lifespan with several "new" versions. Those obviously improve its value by increasing and extending the cash flow. In some cases, subsequent versions are almost a different product and may require an additional valuation.
- Software is the ultimate intellectual property; once it has been created, making and selling an infinite number of copies is easy and cheap, other than marketing expenses; therefore, it has a cost structure completely different from most goods and services.
- Barriers to entry are normally at the marketing, not the code creation level. Logically, one first must find out if anyone really wants the product, and what advantage it may have for any group or sector.
- In most industries there are standards: the world agrees on the layout of car pedals; North America accepts one electric plug and line voltage; Europe, where there has been much more dissention than in the US, managed to do the same - albeit different from ours. Of course, that excludes Great Britain, which uses plugs three times the size of ours... Standards have also been negotiated for software, but with change happening so rapidly, most are determined by the market, followed by the regulators.
- Past losses and the level of shareholders' equity have little impact on the value of a software company; this depends on future prospects.
- One major cost is R&D, which is written off as incurred. In valuations, such amounts are capital expenditure to the extent that an asset has been created.
- Significant contributors to the value of a software company, such as "distribution channels" and "installed user base" do not normally appear in the financial records and are often ignored.

Typical Software Economics

Most industries show declining economies of scale: the bigger an organization or plant gets, the more layers of management and infrastructure are needed, at least, are created - or should I say, metastasize? With software, the opposite is true. Once development is completed, the only large ongoing expenses are for advertising and marketing; manufacturing and distribution costs tend to be fixed and rather low. Therefore, the product has increasing economies of scale, which is demonstrated by the following example based on PC packaged products from real companies:

	Company M	Company C
Product Upgrade R&D	$250 million	$200 million
Software Selling Price/Unit	$350.00	$350.00
Variable Costs/Unit	$50.00	$50.00
Share of Installed Base	65%	28%
Share of Market	80%	13%
Units Sold	8.0 million	1.3 million
Revenue	$2.8 billion	$455 million
Gross Profit	$2.4 billion	$390 million
S G & A (40%)	$1.1 billion	$182 million
Operating Contribution	$1.3 billion	$208 million
Pre-tax Return on R&D	520%	104%

The leader ends up with a good profit, while the follower just gets back its development costs. Therefore, once a firm is in the lead, it stays there, unless it commits a strategic error. Temporary monopolies are quite normal, although none in the software field has been as long-lasting as that of Bill Gates.

If you have seen the Charlie Chaplin classic "Modern Times", you will remember that at some point, the machines simply run away from him and he can no longer cope. A similar situation from an old Lucille Ball show was used a few years ago in a commercial: Lucy and Ethel were stuffing candy into boxes on a conveyer belt that seemed to get faster and faster. Well, with software, speed is everything. Unless it's happening now, and unless you can reliably value it now, everything is bound to change, from any edge it may have over the competition to investors' enthusiasm for the industry.

PART III - SOFTWARE COMPANIES

Accounting For (Software) Twinkies

> "A dollar spent on a toaster doesn't reduce your wealth in the same way as one spent on a Twinkie. One lasts, the other doesn't. But where do toasters end and Twinkies begin in the information economy?"

That question was posed in 1998 by Peter Huber, a writer for Forbes; I don't know whether or not he is a CPA, but he thought out a highly applicable philosophy for recording software in Financial Statements, which, after all, is a CPA's job.

He writes that according to tax collectors and securities' regulators, land values last forever, brick and metal for ten or thirty years, and silicon chips for five. Software is all Twinkies.

Then he reasons that Washington may possibly have it backwards: only half the cost of a software program is to improve current productivity; the other half forms the base for a subsequent version. The software's useful life depends to a great extent on a firm's ability to hang on to its trained employees, which will likely increase productivity year after year. If this is reflected in a firm's accounts, the P&L statement would be different.

Huber goes on to say that, when a company runs out of disk space or processing power, it keeps its old software and files and chucks the computers. With the Internet, a credit card company no longer needs offices in expensive Manhattan, but can locate anywhere cheap, even offshore. The Twinkie is eating the toaster.

FASB has not published any conclusion about this, except for Business Combinations, and neither has the SEC; no one can be certain whether or not something has a real future value until the future is here.

In customary fashion, tax collectors go the other way to maximize revenue; the IRS would like us to not only capitalize software, but practically everything else, from airline engine maintenance to advertising.

Could this by any chance mean that Washington is losing its grip on economic reality?

Products and Companies

With a few glaring exceptions, most software companies are small; therefore, in many cases,

THE PRODUCT **IS** THE COMPANY.

This has advantages as well as disadvantages. Products can be very lucrative while they flourish, but they are generally short lived. That means that, unless the company constantly updates, enhances, even replaces the software, your client should not count on gains on the shares sufficient to put his children through college, or use them to set up trust funds for the grandchildren.

One hundred and fifty years ago, this great state of California was built on mining. When I am talking to politicians who don't under-stand technology - by no means a rare breed - I sometimes use mining companies as an analogy to software firms. Both start with an idea: mining with a prospect, the other with a software concept. Substantial amounts have to be spent on developing the potential ore body - the computer program - before any cash flow is genera-ted. In many instances, there will be insufficient ore - or lack of interest in the software - to become commercially viable.

Once cash flow has started, much of it must be pumped back into further development; find more ore - enhance the software. Eventually, a mine runs out of ore, and a software concept, like DOS, has been passed by and ceases to generate cash flow.

For established entities, whether in mining or in software, the value of the company consists of the value of the products, plus its skilled staff, products (prospects) under development, the opportunity and intention to innovate, and its relationships with customers, distributors, suppliers etc.

Technology

Product Lifecycle

In assessing technology, the position of the product in its life cycle is fundamental; it may be driven by hardware capabilities or customer needs. In recent years, the economic lives of software have become shorter and shorter, but, as many organizations are laggards, adhering to the old adage "if it ain't broke, don't fix it", there are still firms that use long superseded products, so that the "tail" is becoming much longer.

Networks

In the 1990s, there was a trend to move information processing from mainframes to Client/Server systems; these require well designed networks, which increased in importance. At the same time, use of Microsoft's Windows grew much more rapidly than traditional UNIX. Now "free" Linux is gaining ground, as Intranets within organizations and the Internet linking them are having a fundamental impact on the types of software purchased, and how it is applied.

The Elegance of the Solution

The degree of elegance of the solution is important, as is the suitability of the architecture selected, because both have considerable influence on how easy it is to modify and enhance the Source Code. This is affected by the choice of Operating System and programming languages.

The hardware, Operating System and programming languages chosen for a software program are also major determinants of the markets it can serve. To be considered as well are the quality of the Source Code, particularly the amount and comprehensibility of the comments that explain the reasons for decisions, and the completeness of the documentation.

A crucial component is the programming team, the number of people, the languages they know, their experience and adaptability, as well as their ability to communicate internally. It is essential that the various elements of the programs easily fit together and are properly tested, with all bugs being recorded and corrected.

PART IV - THE MARKET

Market Dynamics

- Size: Value increases with size due to increasing returns, while, except for advertising, costs remain fairly constant;
- Growth: Value is enhanced by a rapidly growing market;
- Usage: The more people use or might use a software product, the more valuable it is.

In valuing a software company, it is of considerable importance to determine where its products are in the market cycle. As developed by Geoffrey Moore in his book, "Crossing the Chasm", the market cycle is based on the type of users rather than the period during which the product has been available. The categories are:

- Innovators
- Visionaries (early adopters)
- The CHASM
- Pragmatists
- The Second Gap
- Conservatives
- Laggards

The CHASM is the boundary between success and becoming a living dead; some companies or technologies are never able to cross it, but once a product is starting to be bought by the Pragmatists, its value jumps substantially. Basic differences are set out below.

Visionaries	**Pragmatists**
Intuitive	Analytic
Support revolution	Support evolution
Contrarian	Conformist
Break away from the pack	Stay with the herd
Follow their own dictates	Consult with their colleagues
Take risks	Manage risks
Motivated by future opportunities	Motivated by present problems
Seek what is possible	Pursue what is probable

The delivery mechanisms required to satisfy Pragmatists are very different from those needed for Visionaries.

Size and Growth of the Market and Competition

The potential size and possible growth of the market must be taken into consideration when valuing software. Rapidly changing technologies may permit a competitive product to take advantage of the existing program and add better features, something that may not be possible for the original; this will reduce its value.

The demonstrated size of the market for the competitive product and penetration by a specific technology are a guide to the probable market share of the product being valued.

Rates of growth vary widely, depending on the maturity of the technology and the market. A new solution presented to a stagnant market can totally change growth rates.

On the other hand, if there is no competition, there may be no market, and yet, enormous mass markets have been created for products nobody knew they wanted, such as hi-fis, condos, cruises or health food stores. In 1950, Thomas J. Watson Jr., Executive VP of IBM, approved creating their first general purpose scientific computer, believing they "could find customers for as many as 30 machines".

Today, almost every piece of software is:

1. Replacing an existing solution;
2. Competing head to head with alternatives;
3. Threatened by a novel approach.

All three situations may occur simultaneously.

Positioning

In valuing a software company, it is essential to understand the place each of its products occupies within two interrelated systems; first, the customers' alternative choices for a purchase; second, and more important as it determines the first, are the various companies interacting to make the "market".

The next section builds on "Crossing the Chasm", and "Inside the Tornado" by Geoffrey Moore, the best works I know on "Hi-Tech" marketing.

The Software Marketplace

New Market	Imperialists v Natives	Explorers & Forty-niners
Established Market	Old Guard: • Gorillas • Chimpanzees • Monkeys	Barbarians v Citizens
	Established Product	**New Product**

Understanding this situation is important to a valuation analyst, as a firm's future and the prospects for each product are influenced by management's perception. A firm that does not recognize itself in one of the industry's archetypes is likely to be considered just another 'no name' company, easily ignored by the market and not expected to be around for long. This can become a self-fulfilling prophecy, since survival requires a certain amount of industry support. Each role implies different power relationships, alliances, and competitors.

Industry Archetypes

- The $500 Billion Gorilla: The only question is whether he is a benevolent or a cruel dictator.
- Chimpanzees: A threat to the Gorilla and a target for Monkeys, Chimpanzees must secure their power bases by retrenching into niche markets, building up sufficient product advantage to ward off attacks, and telling everyone that while they are not interested in expansion, they are prepared to defend their turf to the death.
- Monkeys: Their goal is to be the low-cost supplier and the easiest to do business with.
- Imperialists: Members of the Old Guard who have extended established products into new markets, either geographically by deeper penetration, or through adoption of a new platform.
- Natives: The mirror image of the Imperialists; instead of new technology, they have access to the customer through superb distribution and communications channels.
- The Explorers: Oriented to new products and new markets, they are disquieting because they do not seek immediate profits and are in for the long haul.
- Forty-Niners: Farthest removed from other companies, they claim to have found gold and are recruiting partners to cross the CHASM and mine it.
- Barbarians: They attack a contested piece of the market with pincer movements, the way UNIX gradually wrapped it-self around mainframes.

- Citizens: Related to the Old Guard, they fight a war of attrition and counterattack with new technology to pre-serve their position.

Software Markets Are Maturing

In all segments of the software market, other than packaged from the five-and-ten, the structure is slowly becoming one of up to a half dozen leading vendors, plus scores of smaller firms typically providing highly specialized products, many of which are integrated, or used in conjunction with systems from the major suppliers. The exception, packaged software, is dominated by Microsoft's "Office" suite, which has over 80% market share.

Overall, the traditional software markets are now growing more slowly (less than 10% a year) than in the past, with the notable exception of online gaming and location-based Edutainment products. In addition to the obvious impact of the recession in the Asian economies on everybody, including Microsoft, sales are being influenced by the increasing importance of replacement/upgrade business (resulting only in maintaining market shares), and the rise of "good enough, cheap enough" solutions, which are attracting cost conscious new buyers.

Major software producers have a history of acquiring small companies to improve or expand their technology base with complementary products or to extend their markets. This consolidation trend is likely to continue, with purchases being made not only by the majors, but also by specialized vendors, as a company must achieve a critical mass if it is to remain independent. All buyers seek to extend their market, increase their customer base, expand distribution, integrate related product lines and, most of all, add to market share.

Numerous products have relatively high shares of small markets where in effect they created a strong "brand presence". Most have revenues of less than $50 million and address specific functions or applications. Such companies are acquisition candidates, especially as "small cap" software companies are no longer being given high multiples by the stock markets. Privately owned companies have turned to M&A, as since 2000 there has not been an IPO market. The requirements of SFAS 141 and SFAS 142 may result in this exit mechanism becoming more difficult.

Services - The Value Added Function

Outsourcing of a firm's information technology operations is surging, as organizations farm out the operations of their data centers and bring in outside help to assist IT operations. This action, together with systems integrating, contract programming and consulting form the elements needed to establish a leading edge data processing infrastructure. In this approach, hardware and software are interchangeable, upgradeable components, while services "make it happen".

In recent years, this trend has been accelerated by five key factors:

- Shortage of technically skilled personnel.
- The challenge of reprogramming for the Year 2000 and the introduction of the Euro.
- Increasing rate of technology change leading to useful lives of two years or less for some software.
- The requirements of implementing ERP systems.
- Avoiding finger-pointing by one-stop shopping with "prime contractor" responsibility.

As business operations grew more dependent on IT, the services have advanced to become better integrated with the installation and operation of hardware and software. Consulting has expanded from strategic IT planning to process re-engineering, while infrastructure building has moved from systems integration to complete process implementations. Management needs to handle not only data center operations, but also "help desks", desktop PC administration and network supervision.

Most computer hardware and software companies have realized that it is essential to be able to offer services, as customers are increasingly seeking "end-to-end vendor solutions". IBM established its global services group, which purchased the consulting division of PriceWaterhouseCoopers in 2002, and has been extremely successful in obtaining outsourcing contracts and supplying services in support of IBM's hardware and software businesses. Hewlett Packard merged with Compaq, which in the late 1990s had acquired Digital partly to obtain its services operations; the joint capabilities were expected to result in faster acceptance of Hewlett Packard's hardware in major corporations.

PART V - SOFTWARE TRENDS

Underlying Factors

The future growth of the software industry will continue to be driven by two fundamental laws. The first is Moore's Law, dis-cussed previously; it rightly forecast the doubling of microprocessor power every eighteen months for the same price. The second, and probably in the long run more important, is Metcalfe's Law, which states that the value of a network varies with the square of the number of users. No previous technology has led to values increasing at this kind of rate. Together these laws accurately predicted the exponential growth in value of Internet-based companies in the bubble era and resulted in valuation standards that were nonsense; this is shown by the huge declines from 2000 to 2002.

Company purchases of software are typically made on a "one-off" basis; they install a particular capability, such as: word-processing, accounting, inventory control, etc. throughout the organization. When a project is complete and the system installed, there is often a lull, with

employees being allocated to maintain and enhance existing programs until the next major deployment.

This means that enterprise and technical software firms tend to have a cyclical pattern in their sales, around a usually rapidly rising trend. Packaged software suppliers with much lower unit prices show less variation. Edutainment developers have great similarity to movie studios in that all their profits are earned from a few hit products. Internet software producers serve a market that grew rapidly and then declined, but is constantly looking for a "better mousetrap".

In addition to the software spending trends discussed in this part, two vertical markets have "come out of the dark ages" in their use of new software products. The overriding worldwide trend in healthcare is cost containment in the face of aging populations and rising medical expenses; automation is envisaged to become part of any economic solution. Combined with advances in medical techno-logy, this has led to a growing demand for software. Retail, traditionally a low-tech industry, is also increasing its use of software as it adopts Supply Chain Management, Customer Relation-ship Management and e-Commerce.

Firm Lifecycle

A software firm typically goes through six stages before it reaches sufficient maturity to either go public or be sold. One of our cases today deals with increasing value at a later stage of a soft-ware company's "private life".

Stage	Key Events	Required Return	Source of Funds
Start-up	Business Plan	40% - 50%	Seed Capital
			Angel Investors
Early State I	Software Development	35% - 40%	Venture Capital
			First/Second Rounds
Early Stage II	Initial Sales	30% - 35%	VC Third Round
Expansion I	New Products New Markets	25% - 30%	Bank Loans
			Mezzanine Debt
Expansion II	Increased Market Share	20% - 25%	Bank Loans
			Bridge Loans
Exit	IPO/Sale	15% - 20%	

The period from Start-Up to Exit at one time was relatively short, three years or so; now it is at least five. However, it should result in an increase in value of over ten times. At the Start-Up stage, the Business Plan is the most important single factor. Unless it is easily understood and credible and the management team complete and experienced, it will be difficult, if not impossible, to get financing other than from family and friends.

Other factors that enhance the ability to raise funds are:

- A large and growing market with few competitors.
- Proprietary technology.
- Significant "first mover benefits" from being the initial supplier and notable barriers to entry by others.

PART VI - SIGNIFICANT VALUE FACTORS

Delivery Mechanism

This is the link between the company and the customers. It includes marketing, sales, distribution, and customer support. As I said before, marketing is the key; very few things sell them-selves. Unless the valuation analyst is satisfied that management understands the market and how to reach it, he can have little or no confidence in its future. Keep in mind that marketing is rarely taught in Engineering, Science or Math Faculties, which produce most software company managers.

Selling

There are many successful methods of selling software, from a dedicated direct-sales force to packages on the shelves of the local retailer. The length of the sales cycle determines the appropriate approach. PC packaged products, which have relatively low prices and can be sold in a few minutes, are normally handled through retailers. A sale of enterprise software to a government agency can take as long as two years and requires dedicated, experienced sales "engineers".

Distribution:

- The distribution method must be geared to the needs of the sector. While direct selling is effective and gives good margins, it is very expensive, as it requires trained staff which must continue to be well paid.
- Channel marketing through Systems Integrators or VARs (Value Added Resellers) is less costly, but results in lower margins and re-quires totally different pricing, cost and management structures.
- The Internet offers low-cost distribution, usually packaged pro-ducts at reduced prices; it can be regarded as a software five-and-dime with sales being downloaded after payment by credit card.

Customer Support

One key factor that is often overlooked: the quickest way for a software company to lose customers is the infernal voice mail, or keeping them on hold on telephone support lines.

Management

Range of Talents

A wide range of skills is of greater importance in a software company than in most businesses. When you value a software company, you must investigate the team that runs it; one-man shows simply don't fly. The team must include individuals with experience in computer science, sales, marketing and finance. It is very helpful if they have backgrounds in seemingly unrelated qualities, as this gives them an edge over the programmed automaton. Make sure the bankers and lawyers are not the ones to make decisions. Beware of sharp dressers; "grunge is good"!

Track Records

It is very difficult to analyze how much any success or failure is due to the individual and how much to the team and circumstances. Also, keep in mind that a failure, or even two, does not necessarily mean bad management; it may well be a benefit if it becomes part of the learning curve. Do they really know how much capital is needed to see the company through its product and sales cycles?

Enthusiasm and Tenacity

Check if they are strictly nine-to-fivers, or if they are willing to hang around when there's work to be done. Are they really prepared to put in the hours, all night sometimes, and to accept the risks necessary to make a software company grow? A useful test is to see who is there on a Saturday or Sunday, and if it's for real jobs or because they can't hack their work load in the week.

Realism

Does everyone really know what they are doing? Is there some wishful thinking? Do the revenue projections look like a hockey stick? A $5 million company can grow by more than 100% for a couple of years, but not a $100 million business. Very few firms go from nothing to $50 million in two years.

Ownership

How much of the company does management own? Some investor owner-ship and outside directors are essential for monitoring, to avoid complacency and ensure responsiveness to the market, but if management holds too small a position, it reduces their incentives.

PART VII - VALUING SOFTWARE

Approaches to Valuation

As with more conventional businesses, the three traditional approaches to valuation, cost, income and transaction, also apply to software. The original investment to create the product is usually

high, as it often involves many blind alleys. On the other hand, the reproduction cost is normally lower, because the methodology has been established.

The replacement cost of the software covers not only recreation of the Source Code and Documentation, but also a factor for the "time-to-market" and the expense of re-establishing dealers and customer base. As none of those costs are reflected on the Financial Statements, they have to be taken into account in determining the Net Worth/Goodwill Value.

Traditionally, the Income Approach obtains a Net Income Value by capitalizing after-tax profit. Most software companies do not make a profit for many years, as their "capital expenditures" on R&D are expensed as incurred. Therefore, income based values are reached by Capitalization of EBITDA, or, more commonly, EBITRAD (Earnings Before Interest, Taxes, R&D, Amortization and Depreciation).

Another frequently used approach is the Discounted Cash Flow (Adjusted Present) Value. This should be applied separately to each managerially relevant segment of the existing operation, so that each of them is valued separately. The tax-shield should be segregated and valued with a lower Discount Rate. Part of the DCF value is a figure for each identified potential opportunity.

As software companies often incur losses, Transaction Based Values are normally established based on multiples of revenues. In selecting multiples, care must be taken to choose suitable comparables to identify expected trends in revenues and the possibility of substantial variations.

Software In Use

It would take a several hundred page book to describe all the types of application software one might find in a corporation. Using our categories, I have listed some of the software you are likely to encounter and will discuss the approaches to valuation. Finally, we look at one type of enterprise software, such as ERP products that run on a wide variety of machines, from legacy mainframes to the latest desktop unit using a range of Operating Systems.

Enterprise

- Accounting/Databases
- E-Mail
- EDI (Electronic Data Interface)
- Business Intelligence/Data Warehouse
- ERP/SCM (Supply Chain Management)
- CRM (Customer Relationship Management)
- Customer Care/Call Centre

Packaged

- Word Processing
- Spreadsheet
- Presentation/Desktop Publishing
- Databases
- Text translation

Engineering

- CAD (Computer Assisted Design)/CAM (Computer Assisted Manufacturing)
- GIS (Geographical Information System)
- Visualization
- EDA (Electronic Design Automation)
- MDA (Mechanical Design Automation)

Edutainment

- Training

Internet/E-Commerce

- Browsers
- Servers
- Intranet

The Cost Approach

As with most tangible assets, the Cost Approach includes two common types. The first is "Reproduction Cost" that covers an exact re-plica of the asset. The second is "Replacement Cost" which recreates the functionality or utility of the asset but may have a different form or appearance. "Replacement Cost - new" typically establishes the maximum amount an investor would pay for an asset. However, specially developed software may be unique and less useful than a replacement offering up-to-date technology. Therefore, the value of the asset must reflect the decrease in value due to functional, technological and economic obsolescence.

Two primary methods are used to estimate the reproduction and Re-placement Costs of computer software; they are: Adjusted Historical Costs and Software Engineering Models. The first method is based on what actually happened, adjusted for inflation and "time-to-market"; it gives the Reproduction Cost. In many cases, Re-placement Cost is significantly lower; this is especially common for older software or programs which were developed or enhanced over an extended period. Software Engineering Models are the most common way of obtaining this figure.

To apply the adjusted historic cost method, all development or acquisition activity is identified and the costs are quantified. Care must be taken to include every applicable item, such as:

- Wages, benefits and bonuses, including options.
- Supervision.
- Management contributions.
- All overheads, including facilities.
- Equipment and materials.
- Testing, including travel to Beta sites.
- Allowance for profit if software is to be resold.
- Entrepreneurial incentive if software for internal use.
- Time-to-market factor.
- Adjustment for inflation.

Software Engineering Models were not created for valuation purposes, but are intended to assist developers in estimating the effort, time and human resources needed for a software project. Three models are in general use: Constructive Cost Model (COCOMO) from the Center for Software Engineering (CSE), University of Southern California; Software Lifecycle Model (SLIM) from Quantitative Software Management, Inc. (QSM); and Checkpoint from Software Productivity Research, Inc. (SPR), all in California.

Remaining Life Analysis

Software developed for sale usually has a relatively short life; many well-known organizations introduce a new version of their major products every eighteen months or so. However, many businesses, when they replace their hardware, transfer the software to the new Operating System and extend its life far beyond that of resale products.

The theory of remaining life analysis was developed in the early 1900s for the railroads. In the various applicable analytical methods, survivor curves are used to estimate the decay rate of a group of similar data points (such as computer programs) as time passes. The theory is very much like the mortality concept used by insurance companies to estimate human life spans.

In most cases, there is insufficient information to calculate Survivor Curves and Probable Life Curves for the lines of code in a computer program. Therefore, our approach is to assume a maximum life of eight years, as after that period, the costs of enhancement and maintenance are usually of the same order of magnitude as the benefits being generated; in such a case, it is often cheaper to replace it with a new program.

In estimating the remaining economic life of software, the valuation analyst should consider at least the following:

- When the program was created.
- The Operating System for which it was designed.
- To what extent it has been ported to another Operating System.
- Maintenance/enhancement practices.
- How well it satisfies the user's current needs.
- Quality of the documentation.
- The degree of comments in the Source Code.
- Does it comply with industry standards and regulatory requirements?
- Are its speed and efficiency suitable for current demands?
- Historical economic lives of similar software.

Income Approach

In an Income Approach, the value of an intangible asset is considered to be the present value of the projected future economic benefits contributable to its ownership over its expected remaining useful life. Those benefits may result from royalty or license in-come, higher operating revenues or cost savings.

Two methods are commonly used in the value of software: Discounted Cash Flow or relief from royalty. You will all be familiar with the Discounted Cash Flow method, so there is no need for me to discuss it any further. However, the DCF value of software includes the present value at a relatively low Discount Rate of the tax savings incurred from its purchase in addition to the present value of the benefits.

The relief-from-royalty method is based on an estimate of the cost savings that accrue to the owner of an intangible asset who would otherwise have to pay royalties or license fees on revenues from its use. The royalty rate chosen is based on analyses of empirical, market derived royalty rates for comparable or guideline In-tangible Assets. As a result, the method is a hybrid between the Income and Market Approaches.

For computer software, revenues from its use are projected over the expected remaining economic life. A market derived royalty rate is then applied to give the estimated royalty savings. An after-tax amount is calculated for each year of the remaining economic life and discounted to a present value in a manner similar to the Dis-counted Cash Flow method.

Market Approaches

Market Approaches estimate the value of an intangible asset by reference to actual transactions involving comparable items. For three reasons, they are extremely difficult to apply in valuing "Software Created for Internal Use" or "Information Bases Generated Internally."

The first reason is that information about sales of this type of software is not readily available. Second, most such events form part of the acquisition of an entire business, and third, custom software is usually unlike other software for which transaction data might be obtained.

Two adaptations of the Market Approach are sometimes applied to computer software: the Market Transaction method, and the Market Replacement Cost method.

When arm's length market data is available for comparable software, the value is typically expressed in dollars per Line of Code or per Function Point (a particular repetitive activity), just like land values are indicated as per acre or per square foot. The unit value can then be applied to the number of Lines of Code or Function Points of the software being valued. The figure is most useful as a reality check, as the necessary adjustments needed for the differences from the guideline software are hard to determine.

The Market Replacement Cost method assumes that, if a commercial off-the-shelf software package can be found that supplies most of the functionality of the software, the costs to purchase or license this package gives an estimate of the its Replacement Cost. How-ever, as the software could be licensed or sold to others, this method will usually understate its value.

In some cases, the valuation analyst may request proposals from independent developers for the creation of programs comparable to the software. Such estimates may be based on originating a complete custom system or modifying an existing package. From an independent source, these objective arm's length estimates give a good indication of the Market Replacement Cost of the software.

PART VIII - VALUING SOFTWARE COMPANIES - VALUATION APPROACHES

All three traditional approaches are also applicable to software companies, but each needs some modification. Most of them do not show any significant Net Income in their first few years of operations; therefore, we generally apply the First Chicago Method for a Net Income Value.

First Chicago Method

When a company is in an early stage of development and a valuation is mainly dependent on a Business Plan and Financial Projections, the "First Chicago Method" of determining the Net

Income Value is often used. Popularised in the 1970s by the Equity Group of the First Chicago National Bank, this looks forward from three to five years, and establishes a future value by capitalising the projected net income at that time.

Usually, three different "Outcome Scenarios" are considered: "Success", "Survival" and "Failure". The Success Scenario is normally the Business Plan, with Survival based on modest growth. As these values are calculated at a date in the future, they must be adjusted to their "present value" at the Valuation Date.

The Discount Rate used is normally the rate of return required by a venture capital investor. The three values are then weighted by the probability of each Scenario and added together. The required additional equity capital is subsequently deducted to give the Net Income Value.

Discounted Cash Flow Method

The starting point, as for the First Chicago Method, is a Business Plan. The Financial Projections are nearly always optimistic and require rigorous trimming; therefore, we frequently apply the DCF approach to the First Chicago Scenarios. The Discount Rate is based on the return required by investors and therefore varies with the client for whom the valuation is being prepared. The Terminal Value is critical, as often little profit will be earned during the projected period. A multiple of EBITDA or sales is a good way to establish terminal value.

Venture Capital Method

Software companies often sell shares to venture capitalists or other third party investors. Such transactions can be useful support to a valuation, but it is essential to recognize that there may have been one or more significant value-creating events between the investment and the Valuation Date.

A further factor is that venture capitalists often purchase convertible preferred shares, which are not directly comparable with the common shares due to retraction or redemption rights, conversion features, dividend and liquidation preferences and control attributes.

Once a venture capitalist has decided to make an investment, its pricing is based on three factors: the required rate of return, the time involved and the expected value under the anticipated exit strategy (IPO? Sale? Buy-back?).

Assume that a venture capitalist is willing to supply $5 million to a new software company and that the Exit Value is expected to be between $70 million and $100 million on an IPO in five years. Based on the company's state of development, the required Rate of Return is between 40% and 50%.

The following matrix shows that after the financing, the current value is between $9.2 million and $18.6 million. Before the financing, the existing shares on this approach are worth between $5.2 million and $13.6 million.

Value of $'million in 5 Years at Discount Rate

	40%	45%	50%
100	18.6	15.6	13.2
90	16.7	14.0	11.9
80	14.9	12.5	10.5
70	13.0	10.9	9.2

Acquired In-Process R&D

Many industries, such as software, electronics, computer hardware, semi-conductors, bio-technology, pharmaceuticals and medical devices have significant on-going in-process R&D. Under SFAS 141, this R&D is to be written off when the firm is acquired.

The key is to:

- Describe the nature of the in-process R&D and determine the stage of development; the write-off is largest immediately before the "Project" becomes a "Product".
- Confirm that at the acquisition date, its technological feasibility had not yet been established and no future alter-native use was known.
- Value the Projects.

The acquirer should apply the same policies in determining the stage of completion of the acquired In-Process R&D ("IPR&D") as to internally developed software. Some factors to be considered are: the nature, amount and timing of the remaining expenditures necessary to develop any project into a commercially viable product.

The SEC has stated that IPR&D cannot be valued as a "residual", such as Goodwill; it rejected the argument that the nature and stage of the acquired company's development implies all excess value beyond identifiable financial, physical and intangible assets to be IPR&D. It believes that every technology company has Good-will to the extent of its trained workforce, and that many other items that do not qualify as Intangible Assets, exist in Business Combinations which involve significant technology. While FASB Interpretation No. 4 suggests that the cost of IPR&D is not a suitable approach to establish its value, the SEC has accepted the replacement costs of the project to the purchaser.

Our preferred approach is the Adjusted Discounted Cash Flow method; this uses projected cash flows segregated into current, pipeline (IPR&D) and future products. These Cash Flows should be

discounted at the suitable rates for the specific risks. The DCF Value must also reflect the tax shield from the deductibility of all costs; the economic life should not exceed seven years without any residual.

Guidelines

Suitable Guidelines are essential in applying the Market Approach; for software companies, those are is often difficult to find. For Internet firms, it may be almost impossible. The traditional price multiples are:

- Price/Earnings
- Price/Revenue
- Enterprise Value (Debt plus Market Capitalization)/EBITDA (Earnings Before Interest, Taxes, Depreciation & Amortization).

However, except for Price/Revenue, they are not suitable for software companies; we therefore prefer the relationship: Enterprise Value/EBITRAD.

The definition of comparability for Guidelines may need to be expanded from just the five types of software to also include other high-tech companies with similar:

- Market potential
- Growth prospects
- Lifecycle stage
- Cost structure
- Correlation to the stock market