SOFTWARE IS EVERYWHERE

James P. Catty

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 required 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.

Background

The software craze is 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 manufacturers independent of waterpower; 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 system. 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 thinking 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 well-informed shareholders.

Factors of Production

The three traditional factors of production are Capital, Labor and Land (resources). The "Digital Age" has added "Information" to the 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 such 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 order to survive."

In our 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 product.

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 will want 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.

What Is Computer Software?

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, compressed the entire 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 written code, flow charts, sub-routines, objects, languages, procedures, documentation, data, etc. that are used to prepare it".

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 is right on track as evidently this recent press release:

On November 5, 2003, the Wall Street Journal reported that Intel Corporation plans to declare that it has removed the biggest roadblock to boosting the performance of chips over the next decade. The Company claims in a technical paper that it has successfully replaced materials that have been a vital part of chips for more than 30 years. The change will help prevent electrical current from leaking inside chips, a growing problem as more circuits get packed into the semiconductors. Electrical leakage generates heat and consumes power. Unless addressed, the problem is widely expected to slow progress in the electronics industry. Solving it could yield products with hundreds of times the calculating power of today’s computers and consumer devices.

Types of Software

The IRS to suit in 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 in-clude 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, spreadsheets and word-processing.

Application Software

Categories of Application Software

There are five main categories of application software. They are based on different technologies and each 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.
- 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. A few comments follow:

•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.
- Prior losses and the level of shareholders' equity have little impact on the value of a software company; this depends entirely on future prospects.
- One major cost is R&D, which is written off as incurred. For valuations, such amounts are capital expenditures 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. 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 a real company:

	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.

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; we 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 yet to publish 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.

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.

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.

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 successes 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 affect 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:

- Replacing an existing solution;
- Competing head to head with alternatives;
- Threatened by a novel approach and

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 coverage we know regarding "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 Bill is benevolent or a cruel dictator.
- Chimpanzees: Figuratively speaking, a threat to the Gorilla and a target for Monkeys, Chimpanzees secure their power bases by selling into niche markets, building up sufficient product advantages 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 more intense 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 itself around mainframes.
- Citizens: Related to the Old Guard, they fight a war of attrition and counterattack with new technology to preserve 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 a 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.

Software Is Everywhere

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 create 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 market. Privately owned companies have turned to M&A, since there has not been an IPO market since 2000. 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 grow 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 accommodate 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 service operations; the joint capabilities were expected to result in faster acceptance of Hewlett Packard's hardware in major corporations.

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, discussed previously; it correctly 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 enhancement.

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 chapter, 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 technology, 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 Relationship Management and e-Commerce.

Firm Lifecycle

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; but 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.

Significant Value Factors

Delivery Mechanism

This is the link between the company and the customers. It includes marketing, sales, distribution, and customer support. 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 Facilities, 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 creates 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 are not the norm. The team must include individuals with experience in computer science, sales, marketing and finance.

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 because they do not complete their workload during 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.

Valuing Software

Approaches to Valuation

As with the 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 by then 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 and corresponding 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" for 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 an amount 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 the following categories, we have listed some of the software you are likely to encounter.

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 Center

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" which assumes an exact replica 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 creates 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 number.

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 overhead, 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 located 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 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?
- The historical economic lives of similar software.

Income Approach

For the 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 to value software: Discounted Cash Flow or relief from royalty. 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 Approach.

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 Discounted 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 versions 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 amount 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. However, 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.

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" as of 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 then deducted to give the Net Income Value.

Discounted Cash Flow Method

The starting point, for the First Chicago Method, is the 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, since 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 for 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 redemption rights, conversion features, dividend and liquidation preferences and control attributes.

Once a venture capitalist has decided to make an investment, the 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 as 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 based upon this approach are valued between $5.2 million and $13.6 million.

$'million	**Discount Rate**		
Value in 5 Years	**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, biotechnology, 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 as of 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, which are 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 and
- Correlation to the stock market

CASE BACKGROUND

This Case Study is based on a valuation of Software being developed by an Internet Service Company in a major Eastern financial center undertaken for a bank as of September 30, 2000. The purpose of the valuation was to give comfort as to the collateral for an interim financing to be secured by all the Company's assets.

The objective was to obtain the Orderly Liquidation Value, which is defined as the total amount expected to be realized on the sale of an asset, or a group of assets, including related intangible items, on the winding-up of a business on an orderly basis. This refers to the amount after the deduction of required legal, accounting, marketing, auction, brokerage and other fees and expenses.

The Business Plan

According to the Company's Business Plan, dated September 2000, it was formed in September 1999 to create a business to serve virtual communities on the Internet, using proprietary Software, which would be an expansion of the well-established chat room for-mat. The initial communities were envisaged to be college and university students, and the Software was intended to supply users with an integrated suite of community tools, productivity aids and content.

A medium-sized college in California signed the initial purchase agreement on October 1, 2000. The Company had also reached an advanced stage of negotiations with a local university to launch the program. Management expected it would take from six to twelve weeks from the date of signing a contract to going live, meaning the service would be available to students at the beginning of 2001. They had informed the bank that the Software was complete, fully tested and saleable; additional content, e-concepts, computers and consulting features were in the process of being added to enhance the user experience.

Contents of the Software

According to the Business Plan, the Software consisted of the following:

- Communications Tools - the Company intended to combine proprietary communications products with what they considered to be the best existing tools and suppliers:
    - Chat Rooms (both private and conference)
    - Forums
    - Instant Messaging
    - E-mail
    - Media Player (audio & video, playing locally, downloading, or streaming)
    - Web Browser

Productivity Aids - the Software was to contain a "powerful personal productivity suite" to enhance an individual's efficiency and organization:

- Calendar
- To-do lists
- Personal home page
- Virtual hard drive (25 MB)
- File services

Content - The third component was to be content aggregated from external sources together with items being developed internally:

- Careers
- Entertainment
- Finance
- Health & Fitness
- Lifestyle
- Local
- Music
- National
- News and Information
- Research Resources
- School specific
- Sports
- Style
- World

The Software that had been supplied to the bank as of the Valuation Date only offered some of the above capabilities on a demonstration basis.

Analysis Of The Company

The Company had four anticipated revenue streams:

- Advertising - because the browser had an advertising window, the Company expected to be able to place banner ads wherever its users went on the Web.
- Sponsorship - as the number of subscribers grew, the Company envisioned sponsorship revenue from firms targeting college students.
- E-commerce - commissions from purchases made by the Software users.
- ISP revenue - from an alumni Internet Service Provider program.

Importance of the Software

Management contended that the Software gave the Company a distinct advantage over its competitors, because, although offering similar content, it would be a vehicle through which students could access all Web sites rather than merely a single destination. The Software would serve its users as a browser regardless of where they went on the Web. Consequently, once a student had completed his first transaction, he would never need to input any credit card information again; in spite of the fact that this practice lends itself to fraud and is severely discouraged by credit card issuers.

In addition to the Software serving as an "e-wallet", this techno-logy was intended to enable the Company to generate complete psychographic profiles of its users. Such information was to enable the Company to deliver a highly targeted and relevant audience to advertisers.

Business Model

The Company hoped to:

> "..... attack the higher education vertical first, primarily through a marketing alliance with PeopleSoft, a major Enter-prise Resource Planning (ERP) vendor in the higher education space ....".

The objective was to gain immediate market entry via PeopleSoft's (with whom it had a letter of intent) existing education customers. An initial contract, effectively a Beta test, was entered into on October 1, 2000.

The Marketing Approach contained in the Business Plan was:

> "First, sign on colleges and universities. Second, give away an online application and register users. Third, sell access to this targeted demographic audience."

Management was targeting further participants:

> "Other potential verticals include pre-college, the military, large corporations, or any other homogenous group that is interested in community building. The business model is primarily based on advertising revenue and commerce generated by the captive audience of the vertical".

By late 2000, the advertising based (TV oriented) model of Internet sites was experiencing problems; advertisers were gravitating to only those with very large numbers of visits, such as AOL and Yahoo. This created doubt that the Company's approach would generate the expected revenues.

Financial Position

As of the Valuation Date of September 30, 2000, the end of its first fiscal year, the Company's Balance Sheet, according to draft Financial Statements in the process of being audited by Arthur Andersen LLP, was as follows:

Assets	**$'000**
Cash & Equivalents	1,055
Prepaids	146
Property & Equipment – net	499
Due by Employees	57
Other	124
	1,881
Liabilities	
Payables & Accruals	188
Capital Lease	480
Redeemable Convertibles-Preferred Stock	2,020
	2,688
Stockholders' Investment	1,610
Accumulated Deficit	(2,417)
Stockholders' Deficiency	(807)
	1,881

Management stated that the current monthly burn rate was about $275,000, meaning it had sufficient funds only until early January 2001.

Financial Forecasts

The following Financial Forecasts for 2001 to 2005 were presented by Management in the Business Plan:

						$'000
Year to Sept 30	**2000**	**2001**	**2002**	**2003**	**2004**	**2005**
Schools	-	20	89	175	265	375
Students ('000)	-	69	294	560	747	1,238
Revenues	-					
Banners	-	4,896	33,626	52,589	82,598	125,784
Intstitutional	-	587	5,313	12,787	20,369	31,093
Alumni ISP	-	143	5,134	13,134	23,044	35,581
e-Commerce	-	166	1,134	2,776	4,901	8,684
Sponsorship	135	500	4,000	7,000	15,000	25,000
	135	6,292	49,207	88,286	145,912	226,142
Expenses						
Direct Costs	-	(2,848)	19,723	30,597	48,648	71,200
Commissions	-	386	1,153	1,412	1,318	2,070
Marketing	84	2,514	6,957	8,142	9,192	12,332
Staff	1,450	11,130	15,608	21,483	27,435	32,131
Other	970	3,275	4,772	7,580	10,171	12,623
	2,504	14,457	48,213	69,214	96,764	130,356
Profit (loss)	(2,369)	(8,165)	994	19,072	49,148	95,786
Margin %	n/a	-130%	2.0%	21.6%	33.7%	42.4%

Note: Interstitial means "ads that 'pop up' in a new window over the page the user is viewing."

As of the Valuation Date, the Company was seeking $10 million of third round financing. This would have been only sufficient to execute the Business Plan if the margins and growth shown in the Financial Forecasts would have been achieved. In 1999, the Company's three principle competitors: CampusCruiser, CampusPipeline and CollegeClub, all had revenues of under $3 million. Therefore, in 2001, revenues of the Company would likely be considerably lower than the forecast of $6.3 million.

The College Market

College students are an attractive market. At the end of 1999, there were approximately fifteen million college students in the United States; according to StudentMonitor. Each had a personal disposable income larger than that of the average US family of three, thanks to jobs and the availability of student loans.

Of this population, 50% owned their personal computers and a further 13% used those from school or home for a variety of the following activities:

- 84% studying with other students

- 83% e-mail
- 80% connecting to the Internet, compared with 48% of adults and 46% of teens
- 66% surfing the Web at least once a day
- 54% researching on the Internet
- 36% operating a personal Website or home page
- 32% buying something online
- 26% playing computer games
- 22% participating in Internet chatrooms

The interesting aspect of this list is the relatively small number participating in chatrooms, which the Company expected to become the center-piece of its "communities".

The online purchasing pattern of students in 1999, differed significantly from those of adults:

	Students	**Adults**
Anything	32%	36%
Books	26%	16%
Clothing	15%	8%
Music	13%	12%
Software	16%	11%
Videos	11%	6%

Source: Cyber Dialogue, an Internet research firm.

Forrester Research, Inc., a Boston computer consulting company, stated that in the year 2000, 70% of young consumers highlighted "quick loading" when describing the key features of their favorite Web sites, but they also wanted graphics that grabbed their attention. Striking a balance between cutting-edge technology and ease-of-use was seen as crucial. The sites most frequently visited tended to feature the latest in animation, videos and audio, while remaining easy to navigate and quick to download.

Competition

In the Business Plan, Management stated that, in September 2000, a number of companies were vying for the attention of college and university students through the Internet. Virtually all of these were destination sites rather than a sophisticated online vehicle, such as Yahoo!, which the Software intended to be. These sites included the sixteen listed below that catered exclusively to students. Those marked with an asterisk (*) were not mentioned in the Business Plan.

Software Is Everywhere

Aroundcampus.com/Egenda.net
Campuschat.com
CampusCruiser.com
CampusPipeline.com
*CollegeClub.com
Jenzabar.com
Mascot Network.com
Mybytes.com
Ontap.com
Powerstudents.com
Studentadvantage.com
*StudentAdventure
*Studentonline.com
*Uzone.com
Varsity.com
Webdorm.com

Evaluating the Software

In valuing any Intangible Asset, it is essential to define the elements of which it is comprised. Ownership of a computer software program normally involves the following thirteen elements:

1. Up-to-date comprehensive set of requirements, against which the design and functionality can be tested.
2. Design specifications, including initial analyses, flow charts and executable parameters.
3. Written Source Code files prepared by the programming staff.
4. Licensed Source Code files, that are owned or copyrighted by others, but which may legally be used for the program.
5. Libraries and executables built using the Source Code files.
6. Compiled executables, whose Object Code can be run on suitable platforms (computers).
7. Testing procedures, documentation, modifications and results.
8. Bug reports and fixes, patches etc., that list all problems observed in testing and set out the solutions adopted.
9. Utilities to allow installation of the integrated compiled executables on the appropriate computers.
10. Systems Documentation that summarizes and cross references all activities undertaken to guide the program from the development arena, through Alpha (internal) and Beta (external) testing, to a production version that can be handled by customers.
11. User manuals that explain all features of the software and give information as to how to operate it.
12. Maintenance records that set out in detail all changes to the program after the completion of testing, together with the reasons for them.
13. Enhancement plans that list the additional functionality or features intended to be included in future versions and an outline of how this is to be achieved.

Documentation

Eight of the elements (1, 2, 7, 8, 10, 11, 12 and 13) of a program are documentation. Experience has shown that software is of little use unless it can be operated and maintained by other than its developers; for that reason, scrupulous, ongoing documentation is an essential part of good software engineering. Documentation has three purposes:

1. State the system's objectives, why particular approaches and specifications were selected and how they relate to the requirements. This "design documentation" usually starts with an initial analysis of requirements and follows through the resulting specifications and designs. These will undoubtedly change during the program's gestation; therefore, requirements and designs should be constantly updated.

   Some programming tools, for instance CASE (Computer Aided Software Engineering), make tasks such as re-drawing flow charts and updating data dictionaries easier. Consequently, when they are used, the final documentation is more likely to be accurate.

2. Describe the software's internal composition, so that it can be maintained during its life. This, inherently technical, is known as "system documentation". One major component is "comments" in the Source Code. As it is essential that the Source Code of all programs be in a readable format so that they can be maintained, good practice uses: well-designed high level programming languages, extensive comment statements or annotations that explain how and why activities were under-taken, and guidelines that allow each module to be viewed as a coherent unit.

   Many software companies have rules for writing programs. These cover: organizing the Source Code on the page; naming conventions to distinguish titles of variables, constants, objects, classes, etc., and documentation formats to ensure that all elements are adequately notated. Such rules establish uniformity throughout a firm's software output, simplifying maintenance. It is essential for a valuation analyst to obtain a copy of such rules and confirm that the software adheres to them.

3. Explain all features of the software and give instructions on to how to operate them. As this is aimed at users, it is known as a "user manual" and tends to be non-technical. Good manuals, combined with a well-designed interface, make a software program more accessible and facilitates its acceptance. They can often be an important marketing tool.

   Recognizing this, many software firms hire technical writers to produce them, or supply preliminary versions of their products to independent authors, so that "how-to" books become available at the same time as the software. Traditional user manuals took the form of booklets, but in 1999 started to be added to the software itself in the "help" files. This allows easy reference to the documentation while operating the software.

   Testing successful software development typically includes significant effort (up to 50% of the total cost) devoted to traceable, repeatable testing and documenting results. Effective tests must be related to system requirements. When it is unclear what behavior is required, much effort may be wasted. Testing must verify that the system performs as designed, not only under nominal circumstances, but also as the boundaries of allowed inputs are approached. The system recognizes when such boundaries have been exceeded and terminates or restricts operations in the most orderly possible fashion.

State Of The Software

The bank received the Software on a CD-ROM containing three folders: "install.ssa" (96,909KB) from October 1, 2000; "Source.ssa" (284,287 KB) from October 2, 2000, and a "readme" file that said: "To install, you need MS Visual Sourcesafe 6.0. You can restore the above archives using the Visual Sourcesafe Admin tool." In appraising software, the valuation analyst needs to have an under-standing of the various Source Code archive systems of which Sourcesafe is one of the most common.

Size of the Source Code

The Software was reasonably large:

Archive	Install.ssa	Source.ssa
Items	752	5,300
VSS database properties	101 Megabytes	374 Megabytes
	2,078 Files	15,819 Files,
	123 Folders	612 Folders

And contained four types of files:

1. Original, Company developed code and data. These were originated by the Company's staff and are the largest contributor to the value of the Software.
2. Copyright, "Open Source" software and data. This is material from organizations or individuals who make their work freely available for reuse, provided their copyright is acknowledged. This code was integrated into the Software's architecture to reduce development costs. Unless subject to a rare and unique adaptation or use, it adds little to the value. At this time, it is not possible to be certain to what extent, if at all, such modifications were performed.
3. Original, Company developed sound and graphics data. These cause a PC to display pictures or produce sounds and were intended as an integral part of the user Software's experience, and their quality affects its value. There was neither information, nor was there any systems allowing their efficient modification when required.
4. Copyright sound and graphics data. Sound and picture information, to which the copyright is held by others. There was no evidence of the licenses necessary to allow those to be included. Under those circumstances, the presence of such material in any software installation represents a negative value and possible litigation.

The number of type 1 and 2 files in the Source Code was as follows:

	Lines	
Original Code	255,563	30.6%
Duplicate & Copyright Code	318,717	38.1%
Total Code	574,280	68.7%
Comments only	123,006	14.7%
Both Code & Comments	19,522	2.3%
Blank	118,990	14.2%
Total Code	835,798	100.0%

Note: The original code figure includes a small (5%) allowance for undocumented modifications to the copyright code, so that it would work efficiently with the Company generated material.

Other files in the Source Code archives included:

- Eighty-one Zip (compressed) files, for loading on to a student's PC in order to reduce the Servers transmitting graphic information. The ability of its browser to use locally stored graphics to shorten display delays is a "value added" feature of the Software. A problem with some of those graphic files was lack of information as to their origin, what software was used to create them, and, therefore, how they could be modified if required. No copyright information was embedded within the graphics, as found in the archives. Management stated that they were all usable but did not provide any documentation.
- Eighteen un-annotated "install" files.
- One thousand five hundred and seventy-five graphic files that generated displays on the students' monitors. No documents defined their "look and feel" criteria, or how they were generated and maintained. Some graphics are almost certainly copyrighted or included images of people or items unlikely to be in the public domain.
- Forty-three sound files generated audio prompts or background "ambience" while the Software was in operation. A number were checked and found to be currently copyrighted; Management was not aware of the problem.

Verification Required for the Source Code

The archives did not include any development documentation, printed or electronic; the absence of established administrative procedures may indicate organizational problems that could have negatively affected the development of the Software.

Any outside developer wishing to use the information in the archives would have to verify its integrity. Since there were no build instructions or test records, this will require definition and establishment of documented, configuration managed, development and test systems to ensure complete traceability. The cost of this would be of the order of $100,000, using an organization with suitably equipped and available facilities and a comprehensive development teamwork.

After the verification stage, testing of the Software would have to be undertaken by a significant number of computer literate individuals posing as "students". This is estimated to involve about 12 person weeks of programming staff and 75 person weeks of students, plus rental of suitable equipment at an additional cost of about $85,000. Only after this would a working edition of the Software be ready for sale.

Documentation in the Archives

The archives include a directory, "Docs", which contains only three documents, all incomplete or in a "preliminary" stage; there is no established documentation style. Some files with names, often identifying documents, were found scattered throughout the archives; most were empty. This suggests that, in designing the Software, Management had expected such information to be included, but that poor control resulted in this not being done. Certain programs, which automate the generation of application framework code, also generate boiler plate descriptive text; a number of examples were found.

Another problem with the documentation is that none of it is identified with a particular version; as a result, there is no explanation of changes that may have been made to the Source Code over time. Such information gaps make the development and test processes un-auditable, raising further questions about the Software's quality.

Comments in the code were informal, stylistically inconsistent and often insufficient. A few developers prepared well commented, easily readable code; the majority were much less disciplined. Spot checks did not reveal any files having been modified solely to correct documentation issues; it must therefore be assumed that the code was not reviewed for this purpose.

Operating Systems and Languages

The Software uses a three tier Client/Server architecture, with UNIX based application and database Servers. The Client is intended to run on a desktop or notebook using Microsoft Windows. To install and fully utilize the Software, the PC must have a CD drive as well as sound and color graphics capabilities.

The Server Software was developed in RedHat Linux, a popular open source UNIX variant. The code indicates that the Company planned to operate it on other UNIX variants, such as Solaris from Sun Microsystems. Major modifications would be needed to run it on HP-UX (from Hewlett-Packard) or AIX (IBM). Another significant issue is that no documentation showing whether or not the problems involved in Server scaling and porting to different UNIX variants had been considered. In addition to standard UNIX functionality, a PHP (see below) enabled Web server, an Oracle database and Java virtual machine software were required; all of these were easily available.

While the archives contain software to create empty Oracle databases for the Software, nothing was found which could transfer student information from the institution's records to them. This is essential to encourage initial use and build the "community", particularly should an organization wish to ensure that all users were registered. Only a limited portion of the administrative software required for efficient Server operation existed. This would have to be created together with that necessary to support advertising sales, which would require statistical information on items such as frequency of feature usage, time spent within features, also time-of-day usage patterns.

The following languages were used to create the Software:

"C"	The majority of the code; although the Client was developed using Microsoft's Visual C++ Integrated Development Environment, none of the Source Code is object-oriented.
Java	Used for some interactive chat functionality.
JavaScript	Used for interactive Web pages.
PHP	This open source product was applied to dynamically define the HTML (HyperText Markup Language) data displaying Web pages.
SQL	Structured Query Language for interfacing with the Oracle database.
UNIX Shell Script	Automated a number of important administrative functions.

All of the languages were well known and appropriate for a Client/ Server application, although by 2000, "C" was becoming obsolescent for new projects, although it had advantages, as it can run on numerous Operating Systems.

Stability

Given the resources available, stability assessments were only made for the Client. Several versions existed, as demonstrations had been prepared for various intended customers. All versions could be set up by the included automated installation process and subsequently be started successfully. However, at some point, they all crashed; unless corrected, this unstable behavior would have caused severe user distress in a production version.

Scalability

Based on test results, the Client would run satisfactorily on PCs that were then typically being purchased by college students; older machines or Macintoshes would have had difficulties.

Scalability of the Server Software was more important but difficult to assess. As a practical matter, given the numerous details to be considered, server software of this type can only be considered scalable when it has actually been built and run on equipment with different computing power and storage capacities. Procedures and utilities, which reliably allow the Server Software to be installed and tuned for other configurations, did not exist. While the Software was potentially scalable, the probability of this being performed without major difficulties was no better than 20%.

Scalability was significantly reduced by the languages chosen, as it was designed and implemented using a procedural rather than an object orientation. Both approaches can generate high quality software, but object orientation offers considerable advantages when an existing design is to be reused, as in a change of scale; although the newer of the two disciplines, it was a mainstream approach in 2000. For example, should a customized "chat" product be required for a particular institution, it could be more quickly developed from a library of objects rather than one of procedures.

As use of the chat or information browsing features, the highest potential value portions of the Software, will be discretionary for students, considerable effort has to be spent to ensure its reliability. If either system is subject to frequent or drastic failures, users, especially students, can be expected to quickly become disenchanted and "community building" will fail. Experience has shown that even offering generous incentives will not succeed in encouraging users to retry a system if their initial impression is unfavorable.

Testing

Tests must be repeated over and over. This is especially important after correcting failures or adding new features, to ensure that the changes have not introduced a problem in a previously functioning portion. When a failure occurs after software is in production, it is essential to rigorously repeat the related tests, so that reasons why the problem was not discovered sooner can be identified and the procedure revised.

Insufficient effort has been expended to make the Software testable by design, as this consumes substantial resources and is often, erroneously, perceived as not adding value. One way of reducing these costs is to consider how all elements will be tested as they are designed.

The test software found in the archives did not cover boundary or stress testing, but only assessed some basic functionalities; it did not seem to have been created as an integral part of the development process. When this is done, not only is automated testing possible, but modules and subsystems may be easily verified after each build, allowing problems to be detected and corrected early.

The Software contained almost no checks of data validity, neither for internal values nor for those from external sources. During development, it is desirable to take advantage of generally available code, which assesses the suitability of data that is transferred between modules. This allows early detection of any area that may be making miscalculations. Since such assessments are usually redundant by the time of sales, this software can easily be eliminated from the final version, but readily re-enabled for future development. No evidence of such code was found. During the development of the Software, little attention was paid to the logging of errors during testing, debugging and operating.

Maintainability

In the absence of suitable and sufficient documentation, maintenance of the Software by a purchaser would be difficult. Significant time and money will be required to analyze it and create the necessary additional material. The Company's practice seems to have been based on its staff's in-depth personal understandings. The lack of formal descriptions of the original development environment and of integral Software testing will add to the costs of maintaining it by an outside party.

State of Development

The software development process consists of five main stages: (a) design and specification; (b) coding, building and interim testing of systems and subsystems; (c) freezing the design for Alpha (internal) testing and modification; (d) freezing the design again for Beta (external) testing and modification; and (e) finalizing the design for a production version to generate revenues.

The California college contract was entered into just after the Valuation Date and in effect, was to be the Beta test; this is an integral part of the software development process by customers, at their premises, using their own staff and equipment. Therefore, the latest stage at which the Software could have been at that time was at the second freeze, after complete internal testing.

The programming team identified the version in the archives as "Beta 1.1, while the Client is identified as "version 1.4", a description typically applied to a deliverable product. These identifications seem self-serving. Various Clients experienced nasty failures during simple operations; Beta, much less deliverable software, should not have failed in these circumstances.

Bug Fixes

No clear definitions were available on product features, so that it was difficult to determine if any particular change was made to correct an identified problem or to add a feature. Hardly any alterations were explicitly listed as bug fixes; there was no indication of a formal "bug tracking" system.

Software Conclusions

The following conclusions were reached regarding the Software:

1. Available documentation is insufficient. Usability, maintainability and hence its value was compromised as a result;
2. Languages used for development are appropriate and allow for its cost-effective maintenance;
3. Programs sufficient to produce a running system were present, although significant costs and effort would have to be expended by a buyer to re-establish the baseline configuration;
4. Further work is needed to establish that both the Client and Server code will work on the range of Operating Systems likely to be found at various customers' sites;
5. Insufficient software has been developed to allow effective administration of Server databases;
6. Code will have to be created to provide information required by advertiser;
7. Data to define enhancements necessary for improved community building will have to be obtained;
8. The Software has been insufficiently tested; further and better tests must be defined and properly recorded;
9. The Software appears to be at the Alpha stage of development. It is ready to enter comprehensive laboratory testing and be demonstrated to potential customers;
10. It has insufficient stability and reliability to begin use at a customer's site.

Valuations of the Software

Orderly Liquidation Value, defined earlier, will nearly always be lower than Fair Market Value, as the seller is usually under a compulsion to deal and the assets are normally sold on a piecemeal basis rather than as a going concern. Orderly Liquidation Value assumes that the sales occur over a sufficient period to allow normal exposure to appropriate secondary markets. A still lower amount is given by the Forced Liquidation Value, which is based on less exposure over a shorter period, probably simply an auction.

There are two methods of obtaining an Orderly Liquidation Value: one is to deduct various discounts, fees and costs from Fair Market Value, the other is to look at actual liquidation transactions. For software, the first is usually applied, as little satisfactory information with regard to actual transactions in comparable assets is available.

Approaches Selected

Traditionally, three approaches are used to obtain Fair Market Value: Cost, Income and Market. In the case of the Software, the Income Approach is not applicable, as the only available Financial

Forecasts, those in the Business Plan, are not credible. Therefore only the Cost and Market Approaches were applied.

For the Cost Approach, two methods were selected: Historic Cost and Replacement Cost, while for the Market Approach, the implied value of the Software from a financing in June 2000 was taken into account as well as a value suggested by the bankruptcy sale of the assets of CollegeClub in October 2000.

Cost Based Values

Historic Cost

The most common application of the Cost Approach is to assume that the actual costs incurred to create an asset is its value.

From the Company's records, the costs to create the 255,000 lines of original code were:

	$'000
Programmer Salaries	767
Supervision	182
	949
Benefits & Taxes (20%)	190
	1,139
Contractors	549
Rent etc.	42
Equipment	240
	1,970

This is about 80% of the first year's deficit, a normal level for a development stage company and equivalent to $7.54 per line of original code.

Replacement Cost

A well accepted method of estimating the development cost of a software project is to multiply by a cost per line the expected number of lines of Source Code needed. The Company's Vice President of Development stated that the Software contained "400,000 to 500,000 lines of debugged code". He added that this was based on an inventory of running scripts, subtracting comments and non-native languages. A detailed analysis of the Source Code indicated a total of 574,280 lines, of which approximately 255,000 were original.

A large contract programming firm indicated that an average contract programmer in 2000 cost $76 an hour, including all benefits, overhead and equipment in New York, $65 an hour for

Philadelphia, and $50 in North Carolina, where the development group was located. Based on a seven-hour day generating 32.6 lines of fully commented code at a cost would be $10.74 a line and $2,739,000 for the Software.

With a total of 234 files having no comments at all, the Source Code would have to be extensively revised before the Software could be sold. Based on experience, well-functioning, suitably commented Source Code requires a minimum of one line of comment for each 3.1 Lines of Code. To reach this level for all original and totally uncommented files from other sources would require an estimated 18,500 additional lines of comments, at a cost of about $200,000, as well as the $185,000 expenditures for testing.

The replacement cost of the Software in its existing state is $2,350,000, composed as follows:

	$'000
Recreating Code	2,739
less Testing	(185)
Commodity	(200)
	2,354
Rounded	2,350

Market Based Values

Previous Offerings

The following table sets out the amounts, number of shares and prices of the Company's financing:

Date	**Amount**	**Shares**	**Price**	
		$	**$**	
09/99	150	475,000	0.0003	Founders
11/99	1,425,000	8,636,363	0.1650	Insiders
01/00	150,000	263,157	0.5700	Investor
05/00	7,000	6,250	1.1200	Investor
	1,582,150	9,380,770		
06/00	2,000,000	1,791,244	1.1165	Venture Capital
	3,582,150	11,172,014		

As of the Valuation Date, there were also 2,406,520 stock options outstanding, as follows:

	Number	Price
	$	
Employees	2,339,500	0.34
Investor	11,500	1.12
Consultant	55,520	0.155 to 1.604
	2,406,520	

Stock Market Activity in 2000

During 2000, the stock prices of traded Internet and software companies rose to a peak in March and then declined sharply. The well-established Goldman Sachs Software Index, which is published on a daily basis, is a useful proxy for changes, over time, in the worth of unlisted software firms for comparisons.

The following table sets out, for each month in 2000, the high, low and close for the Index, together with the monthly mean. To provide perspective, the differences between the monthly highs and lows as a percentage of the mean, and the percentage change of the monthly means from that of June 2000, when the Company obtained its Venture Capital financing, have been added. During the period, there was a high degree of volatility (over 20%) in each month, except September.

Goldman Sachs Software Index

	High	Low	Close	Mean	Variation +/-	Closing Change
January	534.0	433.6	454.8	483.8	10.4%	8.1%
February	559.7	450.3	558.6	505.0	10.8%	4.4%
March	644.6	514.5	534.8	579.6	11.2%	14.8%
April	535.3	363.0	454.4	449.2	19.2%	-22.5%
May	471.5	354.5	402.8	413.0	14.2%	-8.0%
June	492.5	402.8	463.7	447.7	10.0%	8.4%
July	481.7	389.7	410.6	435.7	10.6%	-2.7%
August	481.4	390.0	481.0	435.7	10.5%	0.0%
Septembe	491.8	433.5	449.9	462.7	6.3%	6.2%

Value of the Company's Common Stock in June 2000

In June, a Venture Capital investor purchased 1,791,244 preferred shares convertible into an equal number of common for $2,000,000, at $1.1165425 a share. In most circumstances, a buyer of convertible preferred shares accepts a conversion price above the Fair Market Value of the underlying common in exchange for priority features. Such premiums vary from 15% to as high as 30%, although they are usually in the 20% to 25% range.

The table below sets out the implied Fair Market Values of the Company's common stock at various conversion premiums, also the related apparent increase in value from the last independent sale of common shares. For this, we chose the weighted average price ($0.5828 a share) of the last two sales, in January and May; on its own, the May sale is too small to be meaningful.

Assumed Premium	**Conversion Price**	**Implicit FMV**	**Increase from Previous Sale**
%	$	$	%
5	1.1165	1.0633	82.45
10	1.1165	1.0150	74.16
15	1.1165	0.9709	66.59
20	1.1165	0.9304	59.64
25	1.1165	0.8932	53.26
30	1.1165	0.8588	47.36

Considering the reported improvements in the Company's situation, and the decrease in the Index between January and June, we believe that the Fair Market Value of the common stock in the June 2000 financing was approximately $0.90 a share; this represents a 24% conversion premium and a 54% increase in value from the previous sales. At this value, the Company had a market capitalization of $8,443,000, based on 9,380,770 issued common shares, and a Total Enterprise Value of $11,238,000, reflecting the $2,000,000 of preferred shares and $795,000 from the 2,339,500 in the money options.

Value of the Software as of the Valuation Date

Between June and the Valuation Date, the Index rose by 3.3%, but started to decline immediately thereafter. Therefore no adjustment was made to the Market Capitalization. The Balance Sheet at the Valuation Date shows a deficit of $807,000, which ascribes a value of $12,045,000 to Intangible Assets, some of which today would qualify under SFAS 141, and more that would not. The table below, based on contemporaneous information from a specialized New York investment bank, shows the typical allocations of Internet company intangibles in 2000:

	Contribution %			
	Start-Up		Operating	
	Minimum	Maximum	Minimum	Maximum
Computer Software	22	27	15	19
Management Team *	15	20	10	13
In-place Workforce *	8	10	5	6
Sales and Marketing Plan *	25	30	10	15
Customers Relationships	-	-	2	4
Sales Channels	-	-	10	15
Revenue Generation *	-	-	25	25
Market Opportunity *	31	13	24	4
	100	100	100	100

* Goodwill under SFAS 141

As the Company in the Business Plan stated that it had progressed from a start-up (development stage) entity to an operating (revenue generating) firm, these allocations give a range of $1,807,000 to $2,228,000 for the Software, with a mean of $2,020,000 (rounded).

Value Based on CollegeClub Sale

On October 19, 2000, the assets and business of CollegeClub, with revenues of about $2.9 million, were sold by the bankruptcy court for $12.5 million in cash and shares. CollegeClub had financial and physical assets of $440,000, fully operating software, a number of advertising customers and a significant audience. Its value was therefore substantially greater than that of the Company, whose intangibles were considerably less.

The value of CollegeClub's intangibles in the bankruptcy sale were $12,060,000, almost the same as those of the Company. That amount gives a similar value of between $1,809,000 and $2,331,000, with a mean of $2,070,000, for the Software. Because it was not yet complete and operational, the Software had a Going Concern Value of at least 20% below that of CollegeClub, even at a bankruptcy sale; this suggests a value of about $1,660,000 for the Software.

Indication of Orderly Liquidation Value

The two previous sections have established four possibilities for the Going Concern Value of the Software. These have to be reconciled before a conclusion can be reached as to its Orderly Liquidation Value.

Reconciliation of Going Concern Values

The four methods discussed, as applied by us, give a reasonably close range ($1,013,000 to $1,289,000) of Going Concern Values; they are set out below with the comparable figures from the NYV Report:

	$'000
Cost Based	
Historic Cost	1,970
Replacement Cost	2,350
Market Based (means)	
Financing Value	2,020
CollegeClub Sale	1,660
Average	2,000

The average is only 1.5% above the Historic Cost, which normally represents the minimum amount a buyer would pay for the Software on a Going Concern basis. Therefore this amount, which involves the least assumptions, was selected.

Orderly Liquidation Value

The values for Intangible Assets, such as software, are much more user specific than those of financial or physical items. In particular, the discount for sale will vary, depending on the synergies a buyer expects. It has to be deducted together with the selling expenses and costs of verification and documentation to establish the Orderly Liquidation Value of the Software.

Sale Discounts and Costs

Information on databases indicates discounts from Book Value on the sales of various assets in bankruptcy proceeds. As internally developed software is not normally carried separately on Financial Statements, no useful information with respect to the discounts of Orderly Liquidation Value from Going Concern Value is available for it. Those for other assets range from 30% for commodity type items to over 80% for specialized equipment. We would expect a discount for sale of between 50% and 60% from the Going Concern Value for the Software, with selling costs in the 8% to 10% range.

Verification and Documentation

Before the Source Code could be sold, the Company would have to spend $185,000 for verification and $200,000 for additional commenting. A further $35,000 is estimated to be needed for the rights to use copyrighted image and sound files.

Valuation Conclusion

The table below sets out the results of these adjustments and indicates an Orderly Liquidation Value of between $135,000 and $210,000:

$'000	**High**	**Low**
Going Concern Value	1,970	1,970
Discount for Sale	(990)	(1,180)
Sale Receipts	980	790
Verification	(185)	(185)
Documentation	(200)	(200)
Selling Expenses	(80)	(80)
For Copyrights	(35)	(35)
Orderly Liquidation Value	480	290

With the further decline in software industry stocks, the value of both, the Company and the Software, had declined sharply by the end of 2000. The bank had made a loan of $600,000 and took a total write-off.

Ten Rules For Valuing Technology Companies

1. Product cycles are the only ones that matter. Each has an upswing and a downswing.
2. Do not fall in love with the technology.
3. Make sure you can understand and use the product.
4. Favor items that are bought rather than have to be sold. High-volume businesses are generally less subject to end-of-quarter fluctuations.
5. There are over 900 public technology companies, so be selective in choosing comparables.
6. Do not rely on input from management alone. They are often the last to know, or admit, a firm's shortcomings.
7. Talk to competitors, customers and suppliers. Ask them about problems.
8. Insight is precious; information is a commodity. Become a participant in the industry, not just an observer.
9. Make sure research insights are balanced with market opportunities. Participation and research establish what can be made, but the market determines what can be sold today.
10. If the products are successful, revenues will follow. Don't pay much attention to financial forecasts. Technology is not financial engineering, it's product development.