VALUING COMPUTER SOFTWARE AND SOFTWARE COMPANIES CASE BACKGROUND

PART IX - 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 at 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 had been 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 first purchase agreement on October 1, 2000; the Company had also reached an advanced stage of negotiations with a local university to launch a 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 scalable; 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 elements.

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 at the Valuation Date could only offer some of those capabilities on a demonstration basis.

PART X - 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 aiming at 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 believed 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 Soft-ware 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; alas, 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 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

At the Valuation Date, 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."

At 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 $6.3 million.

PART XI - 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, a research company, 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 own personal computers and a further 13% used those from school or home for a variety of 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, the latest available data at the time, 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 2000, 70% of young consumers highlighted "quick loading" when describing the key features of their favourite 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 those were destinations 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.

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

PART XII - 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. It 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 as 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 therefore 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 "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 behaviour is required, much effort may be wasted. Testing must verify that the system behaves 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.

PART XIII - 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.

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 Soft-ware'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 was intended as an integral part of the user Soft-ware's experience, and their quality affects its value. There was neither information, nor were there any systems allowing their efficient modification when required.
4. Copyright sound and graphics data. Sound and picture in-formation, 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 unannotated "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 were 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 comprehensive development team work.

After the verification stage, testing of the Software would have to be undertaken by a significant number of computer literate individuals acting 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.

Documentation in the Archives

The archives include a directory, "Docs", which, however, 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 Soft-ware'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 in-tended 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 modification would be needed to run it on HP-UX (from Hewlett-Packard) or AIX (IBM). A 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 data-bases 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 behaviour 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 unfavourable.

Testing

Tests must be repeatable; 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 areas 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.

PART XIV - 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 Income 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, made up as follows:

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

Market Based Values

Previous Financings

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		

At 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 pro-vide 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 of**
%	**$**	**$**	**$ 0.5828**
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	59.65%
25	1.1165	0.8932	53.26%
30	1.1165	0.8588	47.37%

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.

Market-Based Value of the Software at 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 many 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.

Reconciliation of Going Concern Values

The two previous sections have established four figures 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.
The four methods discussed, as applied by us, give a reasonable range ($1,660,000 to $2,350,000) of Going Concern Values; they are set-out below with the comparable figures from the NYV Report:

Cost Based	**$'000**
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 as the Going Concern Value.

Orderly Liquidation Value

The values of 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 from the Going Concern Value to establish the Orderly Liquidation Value of the Software.

Sale Discounts and Costs

Information on data bases 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 listed software company shares, 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. Favour 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 is product development.