CS207 #3, 10 Oct. 2014

Gio Wiederhold & Bob Zeidman

Hewlett 103

Sign in

Mail title of topic you have chosen chosen to Gio@cs before 13 Oct.
1. Why should software be valued? Cost versus value.
4. Income from Sales and Service
5. Sales expectations and discounting of future income.
7. Software growth.
8. Legal & forensics
9. The role of patents, copyrights, and trade secrets.
10. Life and lag of software innovation.
11. How to grow a software company: organic or by acquisitions
12. Open source software; theory and reality. Freemium.
13. Separation of use rights from the property itself.
15. Role of Government
16. Risks when outsourcing and offshoring development.
17. Effects of using taxhavens to house IP. Abolish Corporate taxation?
Reports

Write an initial statement on the issue you are addressing.
Having it written down will help you focus. (;-) need help ?

Then make a list of one or more candidate documents.
You could Google for likely documents and use the files cited on the CS207 wiki page.
Read the ones that seem significant.
Write a one or two sentence summary of the relevant points for your topic
Keep the citations [Author: title; publication, [vol.no.], date, page numbers].
copy Internet files so they won’t get lost

Make notes of their assumptions and results, be critical.

This is your contribution !
Folk that advocate a specific point-of-view often forget or ignore important factors.

Add a brief conclusion.
Relate it to your initial the intro.
The conclusion will tell me -- and the world on the web -- what you have learned.

The value of your work is the clarity of your point.
Don’t worry about insufficient length; it is harder to be brief and clear than voluble.
Methods of valuation

• Value is based on future income
  ➢ Looking into the future is risky
• Having multiple methods match gives confidence
  ➢ There is no best method
Various Approaches to assess IP

Rapid summary only

1. Income Prediction for other products (similar to SW)
2. R&D roll-over
3. Market capitalization (Market Cap)
4. Comparisons with prior acquisitions with IP
5. Comparisons with existing businesses
Continuously updated

1. Corrective maintenance
   *bugfixing reduces for good SW*

2. Adaptive maintenance
   *externally mandated*

3. Perfective maintenance
   *satisfy customers' growing expectations*

[IEEE definitions]

Ratios differ in various settings
Spending to keep going

- SW engineers
  - maintenance tasks
    - bugs ⇐ triage
      - bad for all, bad for some, hold for next release, ignore
    - compatibility
    - virus defenses (subcontract)
    - interfaces with infrastructure
    - interfaces with customers
- SW architects
  - fundamental problems
  - monitor interface changes
- Marketing
- Advertising ➤ to potential customers

• Management
Current value

Prior investment has created what you have now

“a bunch of software”

- That’s what’s to be valued
  - Based on reasonable expectations
    - future maintenance will be needed to earn income
    - future maintenance represents future investments

More “software code”

- not promises of new innovations ← new IP

Later we look at other valuation/business models
Technical Parameters needed

IP is to be valued as of some specific date

1. **Life** of the IP in the product from that time on
   The interval from completion until little of the original *stuff* is left

2. **Diminution of the IP** over the Life
   A bit like a depreciation schedule, but based on content replacement, until little IP is left. 10% is a reasonable limit.

3. **Lag period***, interval from transfer to start of IP diminution
   • also called “Gestation Period
   **Effective Lag** = the average time before an investment earns revenue

4. **Relative allocation**, if there are multiple contributors to income.
Crucial assumption for a quantitative valuation

- **IP content is proportional to SW size**
  - Not the value, that depends on the income
  - Pro: Programmers’ efforts create code
    - An efficient organization will spend money wisely
  - Counter: not all code contributes equally
    - early code defines the product, is most valuable
    - new versions are purchased because of new features

- **Arguments balance out**
  - it is the best metric we can obtain
Maintenance causes SW Growth

Rules:
\[ S_{n+1} = 2 \text{ to } 1.5 \times S_n \text{ per year} \] [HennesseyP:90]
\[ V_{n+1} \leq 1.30\% \times V_n \] [Bernstein:03]
\[ V_{n+1} = V_n + V_1 \] [Roux:97] ([BeladyL72], [Tamai:92,02] indications)  [Blum:98]
Deletion of prior code = 5% per year  [W:04], more for embedded code

Codesize

Growth per rule HP @1.5

Growth per rule A

New code per rules A and D for V4
\[ 1 \times v_1 + \times D * Y v3 \]

Replacement rate per rule W
\[ D = 5\% \text{ / year} \]

Deletion of prior code = 5% per year  [W:04], more for embedded code

10/11/2014 Gio CS207 Fall 2014
Observations

• Linear growth has been observed, is reasonable
• Software cannot grow exponentially

Because

no Moore's Law

1. Cost of maintaining software grows exponentially with size
   ▶ the number of interactions among code segments grow fast [Brooks:95]
2. Can't afford to hire staff at exponential *^2
3. Cannot have large fraction of changes in a version
   ▶ and get it to be reliable
4. Cannot impose version changes on users < 1 / year
5. Deleting code is risky and of little benefit
   ▶ except in game / embedded code
Price \qquad \text{remember IP} = f(\text{income})

• But --- Price stays \approx \text{fixed over time} \\
  \text{like hardware Moore's Law}

Because

1. Customers expect to pay same for same functionality
2. Keep new competitors out
3. Enterprise contracts are set at 15% of base price
4. Shrink-wrapped versions can be skipped

• \textbf{Effect}

The income per unit of code reduces by \(1/\text{size}\) \rightarrow
Growth diminishes IP

For constant unit price

Assumptions:
IP \approx \text{codesize}
deletion \approx \text{codesize}

Note:
less steep if start with \( V > 1 \),
if \( V > 2 \) obeys rule B
The Revised Technology Adoption Life Cycle

Know where you are in the valuation
Predicted product sales for 5 versions, stable rate of product sales 3 year inter-version interval, first-to-last product 12 years, life ~15 years
Fraction of income for SW

Income in a software company is used for

- Cost of capital
  - Dividends and interest $\approx 5\%$
- Routine operations -- not requiring IP
  - Distribution, administration, management $\approx 45\%$
- IP Generating Expenses (IGE)
  - Research and development, i.e., SW $\approx 25\%$
  - Advertising and marketing $\approx 25\%$
  - Joint distributor & creator
  - These numbers are available in annual reports or 10Ks
Recall: Discounting to NPV

Standard business procedure

• Net present Value (NPV) of getting funds 1 year later = $F \times (1 - \text{discount} \%)$

Standard values are available for many businesses based on risk ($\beta$) of business, typical 15%

Discounting strongly reduces effect of the far future

NPV of €1.- in 9 years at 15% is €0.28

Also means that bad long-term assumptions have less effect
Example

Software product

- Sells for €500/copy
- Market size 200 000
- Market penetration 25%

- Expected sales 50 000 units
- Expected income €500 x 50 000 = €25M

What is the result?
Total income

Total income = price \times volume \text{ (year of life)}

• Hence must estimate volume, lifetime

Best predictors are Previous comparables
  ➢ Erlang curve fitting (m=6 to 20, 12 is typical)

and apply common sense limit = Penetration
  ➢ estimate total possible sales F \times \#customers
  ➢ above F = 50\% monopolistic aberration
Combining it all

<table>
<thead>
<tr>
<th>factor</th>
<th>today</th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
<th>y5</th>
<th>y6</th>
<th>y7</th>
<th>y8</th>
<th>y9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>6.0</td>
<td>7.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unit price</td>
<td>€500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Rel.size</td>
<td>1.00</td>
<td>1.67</td>
<td>2.33</td>
<td>3.00</td>
<td>3.67</td>
<td>4.33</td>
<td>5.00</td>
<td>5.67</td>
<td>6.33</td>
<td>7.00</td>
</tr>
<tr>
<td>New grth</td>
<td>0.00</td>
<td>0.67</td>
<td>1.33</td>
<td>2.00</td>
<td>2.67</td>
<td>3.33</td>
<td>4.00</td>
<td>4.67</td>
<td>5.33</td>
<td>6.00</td>
</tr>
<tr>
<td>replaced</td>
<td>0.00</td>
<td>0.05</td>
<td>0.08</td>
<td>0.12</td>
<td>0.15</td>
<td>0.18</td>
<td>0.22</td>
<td>0.25</td>
<td>0.28</td>
<td>0.32</td>
</tr>
<tr>
<td>old left</td>
<td>1.00</td>
<td>0.95</td>
<td>0.92</td>
<td>0.88</td>
<td>0.85</td>
<td>0.82</td>
<td>0.78</td>
<td>0.75</td>
<td>0.72</td>
<td>0.68</td>
</tr>
<tr>
<td>Fraction</td>
<td>100%</td>
<td>57%</td>
<td>39%</td>
<td>29%</td>
<td>23%</td>
<td>19%</td>
<td>16%</td>
<td>13%</td>
<td>11%</td>
<td>10%</td>
</tr>
<tr>
<td>Annual €K</td>
<td>0</td>
<td>1911</td>
<td>7569</td>
<td>11306</td>
<td>11395</td>
<td>8644</td>
<td>2646</td>
<td>1370</td>
<td>1241</td>
<td>503</td>
</tr>
<tr>
<td>Rev, €K</td>
<td>0</td>
<td>956</td>
<td>3785</td>
<td>5652</td>
<td>5698</td>
<td>4322</td>
<td>2646</td>
<td>1370</td>
<td>621</td>
<td>252</td>
</tr>
<tr>
<td>SWIP25%</td>
<td>0</td>
<td>239</td>
<td>946</td>
<td>1413</td>
<td>1424</td>
<td>1081</td>
<td>661</td>
<td>343</td>
<td>155</td>
<td>63</td>
</tr>
<tr>
<td>Due old</td>
<td>0</td>
<td>136</td>
<td>371</td>
<td>416</td>
<td>320</td>
<td>204</td>
<td>104</td>
<td>45</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>Disct 15%</td>
<td>1.00</td>
<td>0.87</td>
<td>0.76</td>
<td>0.66</td>
<td>0.57</td>
<td>0.50</td>
<td>0.43</td>
<td>0.38</td>
<td>0.33</td>
<td>0.28</td>
</tr>
<tr>
<td>Contribute</td>
<td>0</td>
<td>118</td>
<td>281</td>
<td>274</td>
<td>189</td>
<td>101</td>
<td>45</td>
<td>17</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>1032</td>
<td>≈ €1 million</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simple Example
summary

Software product

- 7 versions over 9 years

- Sells for $500/copy
- Market size 200 000
- Market penetration 25%
  - Expected sales 50 000
  - Expected income $25M
  - Discounted gross income $14.7M
  - Available for SW maintenance $3.7M

• Profit $1 M – earlier investment

Will present alternate business models at later dates
Measuring Software Growth

• In the model we projected software growth
  o Using Roux’ rule and fixed release intervals

• After the fact one can measure the actual growth
  • Get a code expert
CS207: Software Forensics

Bob Zeidman
About Bob Zeidman

• President of Software Analysis & Forensic Engineering Corp.
• President of Zeidman Consulting
• Developer of CodeSuite®
• Clients include Apple Computer, Cisco Systems, Mentor Graphics, and Texas Instruments
• Law firms include Orrick Herrington, Wilson Sonsini, Jones Day, Baker & McKenzie
• Author of The Software IP Detective’s Handbook
• Degrees from Cornell and Stanford
Agenda

• Defining Source Code
• Software Correlation
• Software Intellectual Property
• Forensics: Detecting Copyright Infringement
• Stories from the Trenches
• Q & A
Defining Source Code

Source Code (human readable) → Compiler (program) → Machine Code (1s and 0s)
Defining Source Code Elements

• Statements: Cause actions, sequential
  ➢ Instructions: Signify the actions to take place.
    ▪ Control words: Control the program flow
    ▪ Specifiers: Specify data allocations or compiler directives
    ▪ Operators: Manipulate data (e.g., +, -, *, /)
  ➢ Identifiers: Reference code or data
    ▪ Variables: Identify data
    ▪ Constants: Identify constants
    ▪ Functions: Identify code
    ▪ Labels: Specify locations in the program

• Comments: Documentation

• Strings: User messages
// Skip null lines
if (InputLine != NULL)
{
    printf("Store the input line so we can tear it up");

    strcpy(TempLine, InLine);

    InLine[0] = '\0';

    InputIdentifier = strtok(TempLine, SepString);

    while (InputIdentifier != NULL)
    {
        // Put a single space between identifiers
        if (!FirstIdentifier)
            strcat(InLine, " ");
        else
        {
            // Eliminate leading whitespace
            FC = 0;
            while (strchr(SepString, InLine[FC]) != NULL)
                FirstChar++;

            for (i = FC; i <= strlen(InputLine); i++)
                InLine[i-FC] = InputLine[FC];

            FirstIdentifier = FALSE;
        }
    }
}
SOURCE CODE CORRELATION
Define Correlation

- 0 for unrelated source code
- 1 for perfectly related source code
- Exact match (reducing whitespace)
- Partial match
- Functional match
- Transformational match
Source Code Correlation

- $\rho_s$  Statement correlation
- $\rho_c$  Comment/string correlation
- $\rho_i$  Identifier correlation
- $\rho_q$  Instruction sequence correlation
- $\rho$  Overall source code correlation

- $\mu$  Match score (unnormalized correlation)
Axioms

• 1. Commutivity

\[ u_X(F^n, F^m) = u_X(F^m, F^n) \]

• 2. Identity

\[ u_X(F^n) = u_X(F^n, F^n) \]

• 3. Correlation

\[ \rho_X(F^n, F^m) = \frac{u_X(F^n, F^m)}{u_X^{\text{max}}(F^n, F^m)} \]
Lemma

• 4. Maximum match score

\[ u^\text{max}_X (F^n, F^m) = \min (u_X (F^n), u_X (F^m)) \]

• Otherwise axiom 2 is violated
Correlations

• Statement Correlation
  ➢ The result of comparing functional lines of source code

• Comment/String Correlation
  ➢ The result of comparing non-functional lines of source code

• Identifier Correlation
  ➢ The result of comparing identifiers in the source code

• Instruction Sequence Correlation
  ➢ The result of comparing the sequence of instructions in the source code

• Overall Source Code Correlation
  ➢ Each element correlation can be considered a single dimension that can be used to calculate a multi-dimensional overall correlation.
Correlation Equations

• **W-Correlation**

\[ \rho = \frac{\kappa_S \rho_S + \kappa_C \rho_C + \kappa_I \rho_I + \kappa_Q \rho_Q}{\kappa_S + \kappa_C + \kappa_I + \kappa_Q} \]

• **A-Correlation**

\[ \rho = \frac{1}{4} \left( \rho_s + \rho_c + \rho_I + \rho_Q \right) \]

• **M-Correlation**

\[ \rho = \max \left( \rho_s, \rho_c, \rho_I, \rho_Q \right) \]

• **S-Correlation**

\[ \rho = \frac{1}{2} \sqrt{\left( \rho_s \right)^2 + \left( \rho_c \right)^2 + \left( \rho_I \right)^2 + \left( \rho_Q \right)^2} \]
SOFTWARE INTELLECTUAL PROPERTY
• Trademarks
• Copyrights
• Trade Secrets
• Patents
Copyrights

- U.S. Copyright Office: Copyright is a **form of protection** provided by the laws of the United States (title 17, U. S. Code) to the authors of “original works of authorship,” including literary, dramatic, musical, artistic, and certain other intellectual works. This protection is available to both **published and unpublished** works. Section 106 of the 1976 Copyright Act generally gives the owner of copyright the **exclusive right** to do and to authorize others to do the following:
  - To **reproduce** the work in copies or phonorecords;
  - To **prepare derivative works** based upon the work;
  - To **distribute copies** or phonorecords of the work to the public by sale or other transfer of ownership, or by rental, lease, or lending;
  - To **perform the work publicly**, in the case of literary, musical, dramatic, and choreographic works, pantomimes, and motion pictures and other audiovisual works;
  - To **display the work publicly**, in the case of literary, musical, dramatic, and choreographic works, pantomimes, and pictorial, graphic, or sculptural works, including the individual images of a motion picture or other audiovisual work; and
  - In the case of sound recordings, to perform the work publicly by means of a digital audio transmission.
Trade Secret

• The precise language by which a trade secret is defined varies by jurisdiction (as do the particular types of information that are subject to trade secret protection). However, there are three factors that, although subject to differing interpretations, are common to all such definitions: a trade secret is information that:
  
  ➢ is **not generally known** to the public;
  
  ➢ confers some sort of **economic benefit** on its holder (where this benefit must derive specifically from its not being generally known, not just from the value of the information itself);
  
  ➢ is the subject of **reasonable efforts to maintain its secrecy**.
Patent

• Wikipedia: A patent is a set of exclusive rights granted by a state to an inventor or his assignee for a fixed period of time in exchange for a disclosure of an invention.

• The procedure for granting patents, the requirements placed on the patentee and the extent of the exclusive rights vary widely between countries according to national laws and international agreements. Typically, however, a patent application must include one or more claims defining the invention which must be new, inventive, and useful or industrially applicable.
Patent

• Constitutional right: Article I, section 8
  ➢ Congress shall have power . . . To promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries.

• Utility patent
  ➢ Apparatus
  ➢ Method

• Design patent

• Plant patent
Patent or Trade Secret?

- **Patent**
  - Public
  - Easier to defend
  - Easier to steal
  - Limited time
  - Only one inventor

- **Trade secret**
  - Private
  - Harder to defend
  - Harder to steal
  - Unlimited time
  - Many inventors possible
FORENSICS: DETECTING COPYRIGHT INFRINGEMENT
• Software Analysis & Forensic Engineering
• Available online (www.safe-corp.biz)
• CodeMatch®
  ➢ Measures full correlation
  ➢ Produces detailed reports
  ➢ Allows filtering
  ➢ Produces statistics spreadsheets
Detecting Copyright Theft

Source Code 1 \rightarrow Measure Correlation \rightarrow Correlation

Source Code 2

Danger!

October 10, 2014

Bob Zeidman
Source Code Correlation

- Identifier correlation ($\rho_I$)
- Statement correlation ($\rho_S$)
- Comment/string correlation ($\rho_C$)
- Instruction sequence correlation ($\rho_Q$)
- Overall source code correlation ($\rho$)

$$\rho = \frac{1}{2} \sqrt{\left(\rho_S\right)^2 + \left(\rho_C\right)^2 + \left(\rho_I\right)^2 + \left(\rho_Q\right)^2}$$
Reasons for Correlation

• Third-Party Source Code
• Code Generation Tools
• Commonly Used Identifier Names
• Common Algorithms
• Common Author
• Copying (Plagiarism, Copyright Infringement)
Third-Party Code?
- Check search engine

Code Generation Tools?
- Identifying comments
- Identifier names
- Check search engine
- Human generated comments

Common Elements?
- Personal experience
- Check search engine

Common Algorithms?
- Personal experience

Common Author?
- Identifying comments
- Regularly misspelled words
- Unique phrases and identifier names

Copying?
- None of the above

Finding
Correlation
Reason
Stories From the Trenches

• The Case of the Overconfident Defendant
• The Case of the Gullible(?) Expert
• The Case of the Honest Thief
• The Case of the Insane Expert
• The Case of the Sloppy Defendant
• The Case of the Proud Expert
• The Case of the Obfuscating Expert
Summary

• Defining Source Code
• Software Correlation
• Software Intellectual Property
• Forensics: Detecting Copyright Infringement
• Stories from the Trenches
References

Thank You

Bob Zeidman
bob@SAFE-corp.biz