Master copy on Varese

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.

Syllabus:

The order and coverage is flexible

1. Why should software be valued? Cost versus value.
2. Economic Flow. Market value of software companies.
3. Spending. Intellectual capital and property (IP).
4. Income from Sales and Service
5. Sales expectations and discounting of future income.
6. Principles of valuation.
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.
14. Setting licensing rates.
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.

mail topic chosen to Gio@cs before 13 Oct.

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.

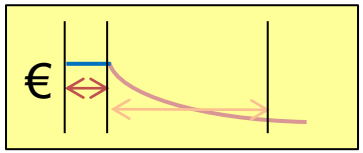
Don't worry about insufficient length; it is harder to be brief and clear than voluble.

Methods of valuation

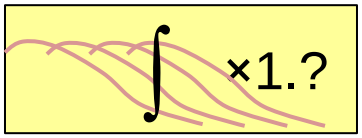
- Value is based on future income
  - Looking into the future is risky
- Having multiple methods match gives confidence
  - There is no best method

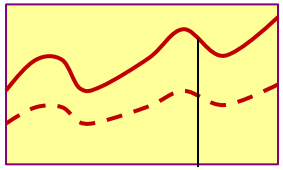
Various Approaches to assess IP

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

Software is slithery !

Continuously updated

1. Corrective maintenance

   bugfixing reduces for good SW

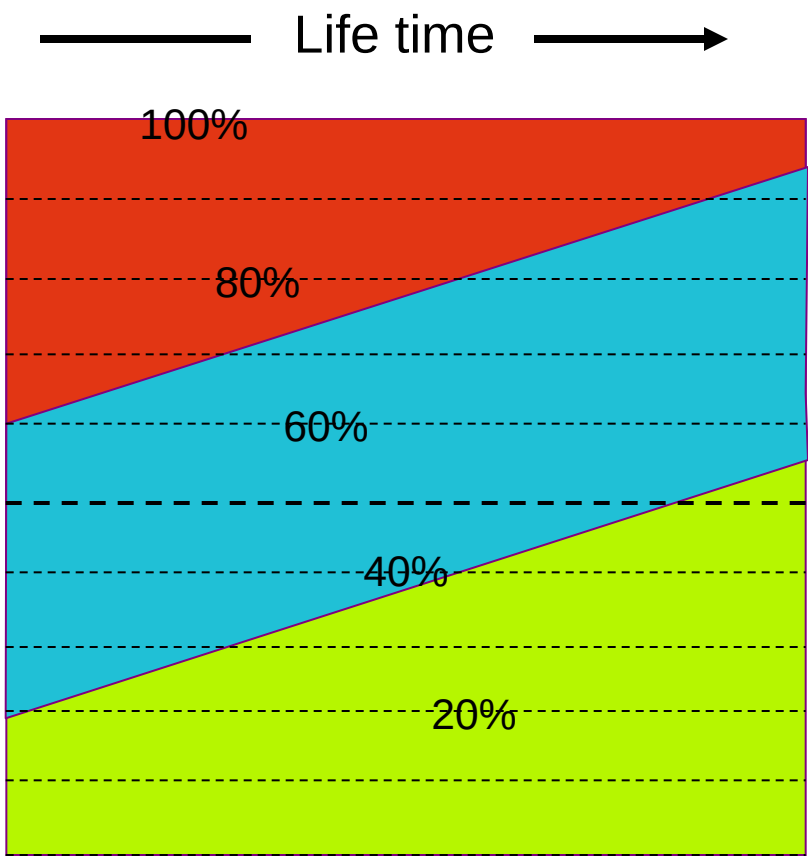
2. Adaptive maintenance

   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

design, code, ...

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

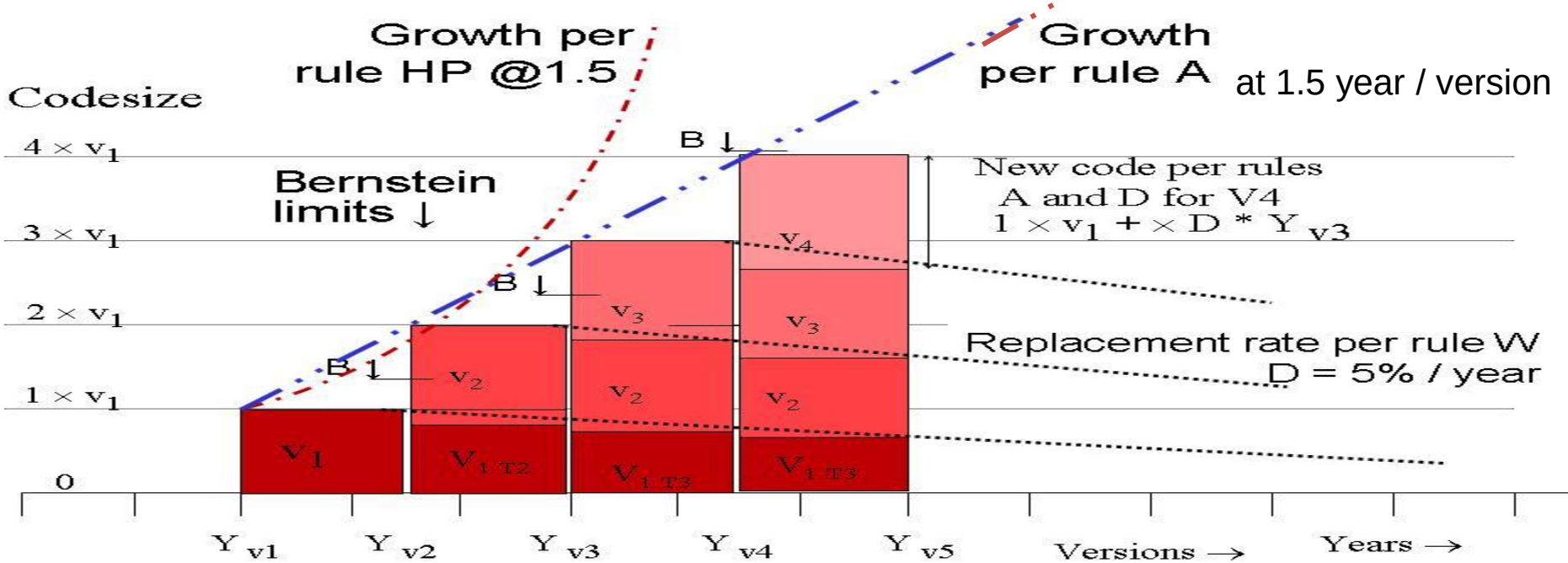
Maintenance causes SW Growth

Rules: S_{n+1} = 2 to 1.5 × S_n 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



Observations

- <u>Linear growth has been observed, is reasonable</u>
- <u>Software cannot grow exponentially</u>

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

remember IP = f(income)

- <u>But --- Price stays ≈ fixed over time</u>

like hardware Moore's Law

Because

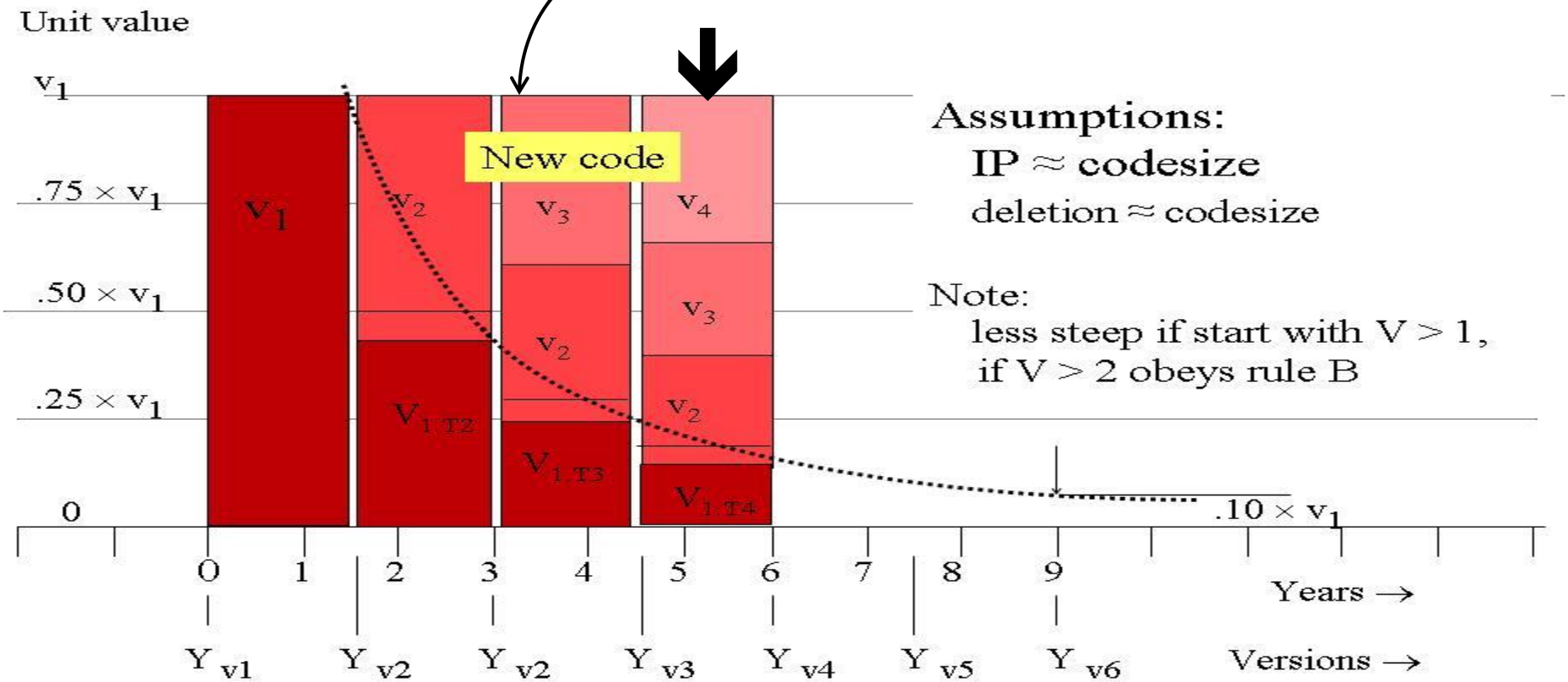
1. Customers expect to pay same for same functionality
2. Keep new competitors out
3. Enterprise contracts are set at 15% of base price
4. Shrink-wrapped versions can be skipped

- <u>Effect</u>

The income per unit of code reduces by 1 / size →

Growth diminishes IP

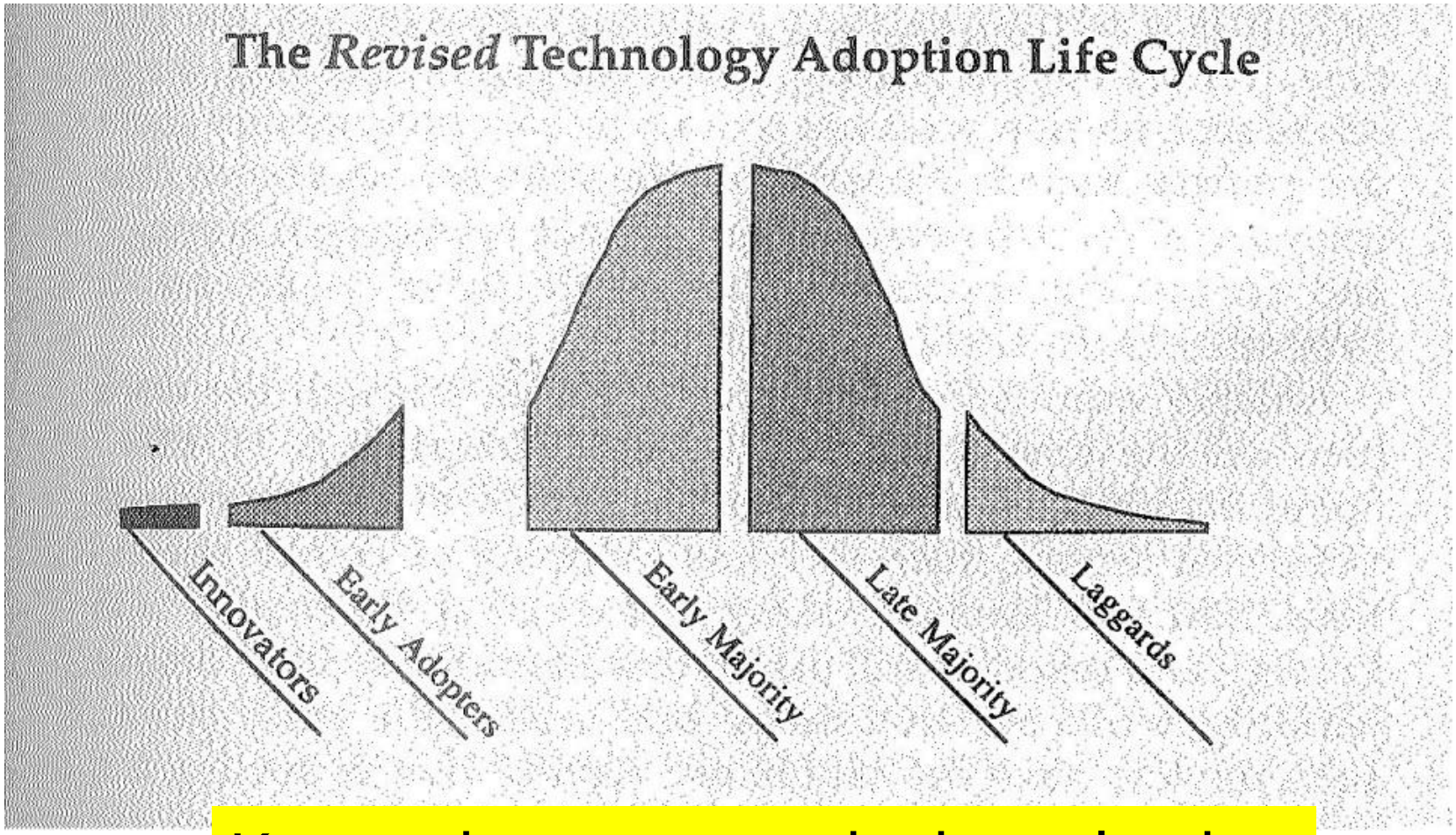
For constant unit price



Assumptions:
IP ≈ codesize
deletion ≈ codesize

Note:
less steep if start with $V > 1$,
if $V > 2$ obeys rule B

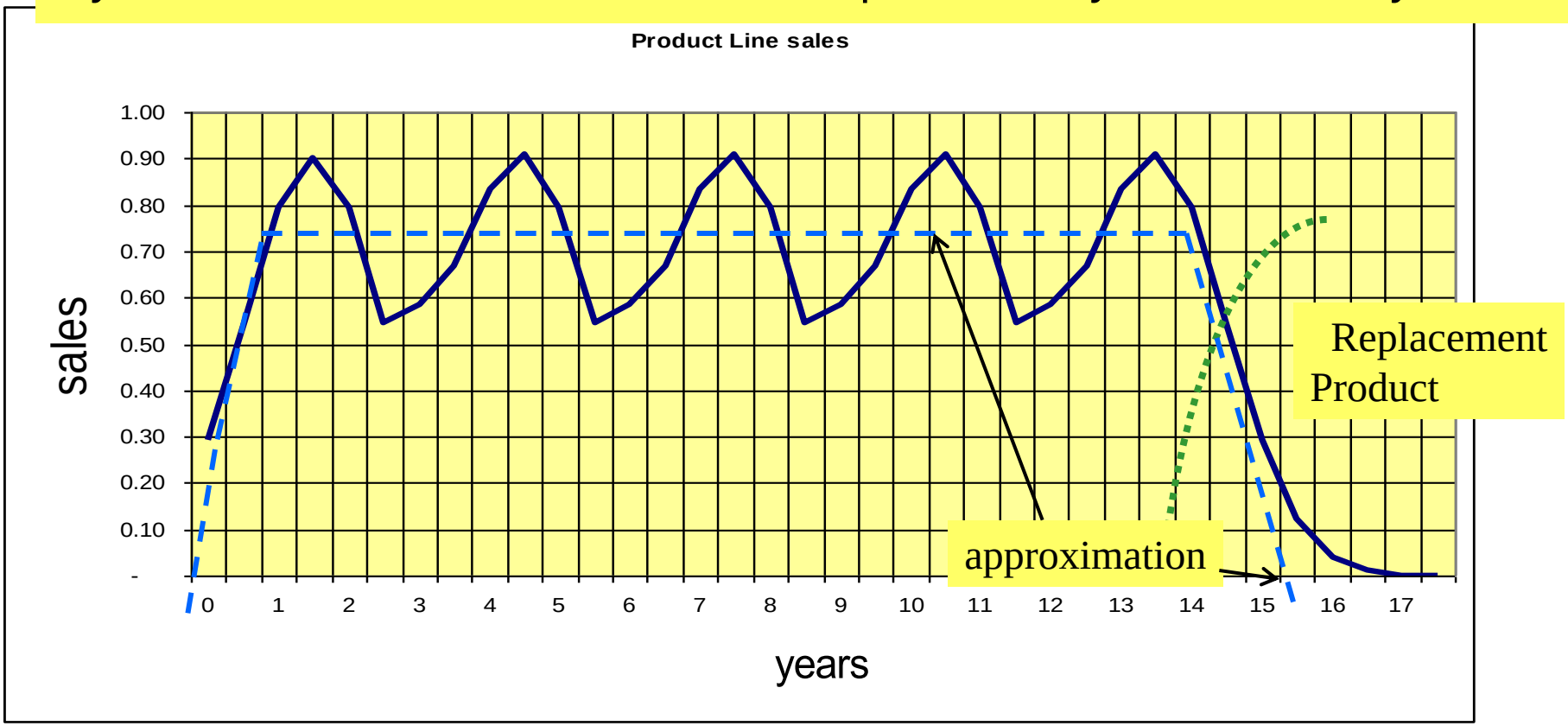
from Moore: Crossing the Chasm



Know where you are in the valuation

Ongoing Version Sales

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

	typical
• Cost of capital	
➢ Dividends and interest	≈ 5%
• Routine operations -- not requiring IP	
➢ Distribution, administration, management	≈ 45%
• IP Generating Expenses (IGE)	
➢ Research and development, i.e., **SW**	**≈ 25%**
➢ Advertising and marketing	≈ 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×(1 – discount %)

Standard values are available for many businesses

  based on risk (β) 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 × volume (year of life)

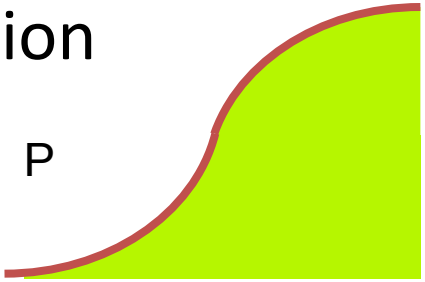
- Hence must estimate volume, lifetime

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 × #customers
- above F= 50% monopolistic aberration



Combining it all

factor	**today**	y1	y2	y3	y4	y5	y6	y7	y8	y9
Version	1.0	2.0		3.0	4.0		5.0	6.0		7.0
unit price	€500	500	500	500	500	500	500	500	500	500
Rel.size	1.00	1.67	2.33	3.00	3.67	4.33	5.00	5.67	6.33	7.00
New grth	0.00	0.67	1.33	2.00	2.67	3.33	4.00	4.67	5.33	6.00
replaced	0.00	0.05	0.08	0.12	0.15	0.18	0.22	0.25	0.28	0.32
old left	1.00	0.95	0.92	0.88	0.85	0.82	0.78	0.75	0.72	0.68
Fraction	100%	57%	39%	29%	23%	19%	16%	13%	11%	10%
Annual €K	0	1911	7569	11306	11395	8644	2646	1370	1241	503
Rev, €K	0	956	3785	5652	5698	4322	2646	1370	621	252
SWIP25%	0	239	946	1413	1424	1081	661	343	155	63
Due old	0	136	371	416	320	204	104	45	18	6
Disct 15%	1.00	0.87	0.76	0.66	0.57	0.50	0.43	0.38	0.33	0.28
Contribute	0	118	281	274	189	101	45	17	6	2
Total	1 032	≈ € 1 million								

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 50 785 V1-V7
- 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

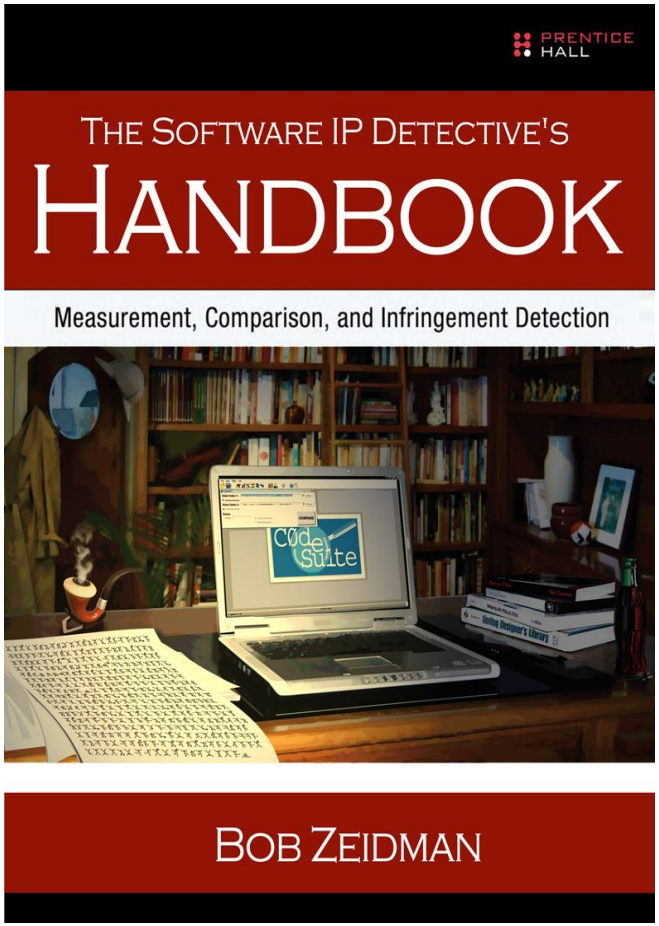
- In the model we projected software growth
  - Using Roux' rule and fixed release intervals

- 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

Bob Zeidman 

Agenda

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

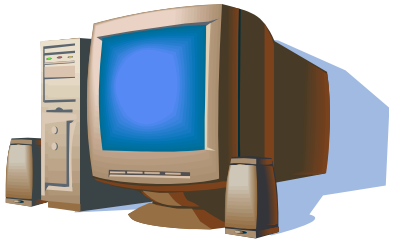
Bob Zeidman

DEFINING SOURCE CODE

Bob Zeidman

Defining Source Code

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



Bob Zeidman

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

Bob Zeidman

Defining Source Code

```
// 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++)
            InputLine[i-FC] = InputLine[FC];

         FirstIdentifier = FALSE;
      }
}
```

Source code elements
Statements
Instructions
Control words
Operators
Identifiers
Variables
Constants
Functions
Labels
Comments
Strings

Bob Zeidman

SOURCE CODE CORRELATION

Bob Zeidman

Define Correlation

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

Bob Zeidman

Source Code Correlation

- ρ_S Statement correlation
- ρ_C Comment/string correlation
- ρ_I Identifier correlation
- ρ_Q Instruction sequence correlation
- ρ Overall source code correlation

- μ Match score (unnormalized correlation)

Bob Zeidman

Axioms

- 1. Commutivity

$$u_X\left(F^n, F^m\right) = u_X\left(F^m, F^n\right)$$

- 2. Identity

$$u_X\left(F^n\right) = u_X\left(F^n, F^n\right)$$

- 3. Correlation

$$\rho_X\left(F^n, F^m\right) = \frac{u_X\left(F^n, F^m\right)}{u_X^{\max}\left(F^n, F^m\right)}$$

Bob Zeidman

Lemma

- 4. Maximum match score

$$u_X^{\max}\left(F^n, F^m\right) = \min\left(u_X\left(F^n\right), u_X\left(F^m\right)\right)$$

- Otherwise axiom 2 is violated

Bob Zeidman

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.

Bob Zeidman

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

$$\rho = \max\left(\rho_S\ ,\ \ \rho_C\ ,\ \ \rho_I\ \ ,\ \rho_Q\right)$$

best

- S-Correlation

$$\rho = \frac{1}{2}\sqrt{(\rho_S)^2 + (\rho_C)^2 + (\rho_I)^2 + (\rho_Q)^2}$$

Bob Zeidman

SOFTWARE INTELLECTUAL PROPERTY

Bob Zeidman

Software Intellectual Property

- Trademarks
- Copyrights
- Trade Secrets
- Patents

Bob Zeidman

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**.

Bob Zeidman

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.

Bob Zeidman

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

Bob Zeidman

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

Bob Zeidman

FORENSICS: DETECTING COPYRIGHT INFRINGEMENT

Bob Zeidman

CodeSuite®

- Software Analysis & Forensic Engineering
- Available online (www.safe-corp.biz)
- CodeMatch®
  - Measures full correlation
  - Produces detailed reports
  - Allows filtering
  - Produces statistics spreadsheets

Bob Zeidman

Detecting Copyright Theft

Source Code 1

Source Code 2

Measure Correlation

Correlation

Bob Zeidman

Source Code Correlation

- Identifier correlation (ρ_I)
- Statement correlation (ρ_S)
- Comment/string correlation (ρ_C)
- Instruction sequence correlation (ρ_Q)
- Overall source code correlation (ρ)

$$\rho = \frac{1}{2}\sqrt{(\rho_S)^2 + (\rho_C)^2 + (\rho_I)^2 + (\rho_Q)^2}$$

Bob Zeidman

Reasons for Correlation

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

Bob Zeidman

Finding Correlation Reason

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

Bob Zeidman

STORIES FROM THE TRENCHES

Bob Zeidman

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

Bob Zeidman

Summary

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

Bob Zeidman

References

1. Article I, Section 8. (1787, September 17). *United States Constitution* . Philadelphia, PA, USA.
2. Copyright Law of the United States and Related Laws Contained in Tıtle 17 of the United States Code. (2007, October). *Circular 92* .
3. Faidhi, J. A., & Robinson, S. K. (1987). An empirical approach for detecting program similarity and plagiarism within a university programming environment. *Computer Education Vol. 11.* , 11-19.
4. Halstead, M. H. (1977). *Elements of Software Science.* New York : Elsevier.
5. Jankowitz, H. T. (1988). Detecting plagiarism in student Pascal programs. *Computer Journal, vol. 31, no. 1* , 1-8.
6. Manual of Patent Examining Procedure (MPEP). (2007). *E8R6* . United States Patent and Trademark Office.
7. Parker, A., & Hamblen, J. O. (1989). Computer Algorithms for Plagiarism Detection. *IEEE Transactions on Education, Vol. 32, No. 2* , 94-99.
8. *Random House Unabridged Dictionary.* (2006). Random House, Inc.
9. Robinson, J. A. (1987). An empirical approach for detecting program similarity and plagiarism within a university programming environment. *Computer Education Vol. 11.* , 11-19.
10. Tysver, D. A. (n.d.). *BitLaw*. Retrieved from The History of Software Patents: http://www.bitlaw.com/software-patent/history.html
11. *Wikipedia*. (n.d.). Retrieved from Trade Secrets: http://www.wikipedia.org/wiki/Trade_secrets
12. *Wikipedia*. (n.d.). Retrieved from Patents: http://www.wikipedia.org/wiki/Patents
13. *Wikipedia*. (n.d.). Retrieved from Software patent debate: http://en.wikipedia.org/wiki/Software_patent_debate
14. Zeidman, R. (2008). Multidimensional Correlation of Software Source Code. *2008 Third International Workshop on Systematic Approaches to Digital Forensic Engineering* (pp. 144-156). Oakland: IEEE.
15. Zeidman, R. (2006). Software Source Code Correlation. *5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)* (pp. 383 - 392). Honolulu: IEEE.

Bob Zeidman

Thank You

Bob Zeidman

bob@SAFE-corp.biz