The Product Flow Model

Gio Wiederhold
Stanford University
Oct 2004
Current approaches in SW engineering
You know them, but they are rarely compared

Maintenance
The major cost of software, ignored by creators

Product Flow Approach
Make maintenance profitable

Modeling income
Value of software today derives from future income

Business effects
A sustainable SW business
Customer and marketing in the loop
Varied Software Models

Guide the process of SW creation

• By individuals for themselves
  – Dykstra model

• By IT departments for their enterprise
  * Waterfall model / Spiral model / Watersluice

• By software supplier for individuals
  – Shrinkwrapped, ~ biannual versions

• By software suppliers for enterprises
  ❖ Product Line model [Boehm:00]

• Start-up enterprises
  – See if it sticks. If it does, convert
Risk minimizing approach

• Develop highest risk modules first
  – avoid problem of spiral approach

* Water Sluice [Burback 2000]

Modularization

Design Validate Implement Test

Design Validate Implement Test

Design Validt Impl’t Test

↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
Assess risk, reassign if needed

integrate . . .

Time →
Product Line Approach

Strategy for a software supplier
- Specialize in some domains
- Invest extra in each product delivered

Value embodied
- in reusable software modules
- in staff knowledge about them

Recognized by consulting / integration firms
Long-term Software Models

Guide the process of SW creation

• By individuals for themselves

• By IT departments for their enterprise
  – Waterfall model / Spiral model / Watersluice *
    If common objective, share costs

• By software supplier for individuals
  – Shrinkwrapped, ~ biannual versions

• By software suppliers for enterprises
  – Product Line model [Boehm:00] **

• Startup Enterprises
SW creation ≈ Business

In-house control by customer

\[ \leftarrow \text{schedule} \rightarrow \]
\[ \leftarrow \text{feasible size} \rightarrow \]
\[ \leftarrow \text{in house external} \rightarrow \]
\[ \leftarrow \text{short life long life} \rightarrow \]
\[ \leftarrow \text{responsiveness contracts} \rightarrow \]

Outsourcing to remote service

\[ \leftarrow \text{choices} \rightarrow \]
Outcome: IP

Delivery of Intellectual Property (IP)

• Some SW embodies company competence.
  – Don’t cede control: best kept in house!

• Remaining commodity SW
  – Purchased for in-house use or service?

Over time the inventory of SW grows
  – Internal development + purchased software

• Has to be maintained → growth
Problem

As the software inventory grows

- The maintenance load increases
  - Internally developed software by enterprise staff
  - Externally obtained software using external and enterprise staff
- Increasing load relative to new work
  - Often not highly valued
  - Assigned to least competent staff?
Effects on software

• Long life
  – new releases/versions must be compatible
  – new releases/versions retain IP
  – experience: base life $\approx 15$ years

• Steady growth
  – less than exponential because of complexity
  – $\text{Size}(V_n) \approx n \times \text{Size}(V_1)$
  – Occasional module rewrite
Impact

Effort \approx SWsize^2 \quad \text{[Brooks:95]}

Time →
Quantifying SW Growth

Rules: $S_{n+1} = 2 \text{ to } 1.5 \times S_n \text{ per year} \ [\text{HennesseyP:90}]

$V_{n+1} \leq 1.30\% \times V_n \ [\text{Bernstein:03}]

$V_{n+1} = V_n + V_1 \ [\text{Anonymous:97}]

Deletion of prior code = 5\% \text{ per year} \ [\text{W:04}]
What is Maintenance?

Definition: *Unscheduled* tasks to

1. fix errors in software
2. adapt to externally imposed changes
3. perfect to users’ desires

Crucial leverage:

70-90% of system costs are SW-maintenance

If maintenance costs can decrease 25% we double our capability to develop innovative products

If maintenance costs increase by 25% we lose any capability for innovation

Who should do the work?
Types of Maintenance

- Corrective ~ 40% - over time → 5%
  - Bug fixing,

- Adaptive ~ 40%
  - Compliance with external needs
    - New hardware, associated software
    - Changing standards
    - Government regulations

- Perfective ~ 20% → 55%
  - User / customer expectations
    - Ease of use
    - Scale up of performance, capacity
    - Improved functionality
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*

Ratios differ in various settings
Input for Maintenance

- **Corrective** from customer:
  - Complaints → Log → Triage → Fix in next release
  - Patch now
  - Ignore

- **Adaptive** from Infrastructure suppliers
  - from standards orgs
  - from governments
  - Ignore
  - Fix in next release
  - Patch now

- **Perfective** from customer via marketing
  - Evaluate cost/benefit → Assign
  - next version
  - next release
  - never / future
Work flow methodology

Consider inputs, outputs that link processes

and assignments to staff & business units
Product Flow Approach

Extend software product line to
• include maintenance (the high cost portion)
• the feedback that motivates maintenance

Move maintenance to software supplier
• exploit development staff knowledge
  – suppliers’ value
• develop tools for maintenance
  – tools reduce reuse costs, also for customer

Product Flow Model (PFM)
Maintenance increases life

- Automobile: 5 years, 17% maintenance, 20% depreciation/year
- Hardware: 3 years, 9% maintenance, 33% depreciation/year
- Software: 15 years, 80% maintenance, 7% depreciation/year

Relative annual maintenance cost:

- Total automobile: 23.4% per year
- Total hardware: 16% per year
- Total software: 12.3% per year

Depreciation/year = 1 / lifetime
Value at the enterprise

• Maintenance for purchased SW is delegated
  – Avoid work for which no expertise exists
  – A larger percentage of IT staff works on tasks that are valuable for the enterprise (may not be more people)

• Retain control of enterprise specific SW
  – Protects IP
  – Still need staff for internal SW & interfaces

• But tied to supplier -- high cost to change
Income to supplier

1. from new sales $S_i$ in year $i$
   - improved products at stable price -- as *hardware*
   - supports sales cost and engineering: 50/50?

2. from maintenance fees 15% $\sum_{i}^{n} S_i$
   - supports basic service and engineering: 50/50?
   - steady (95% of customers renew annually)

![Graph showing income from sales and maintenance over 16 years of life.](chart.png)
Total income

Total income = price \times 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 \times \#customers
  ➢ above F= 50\% \ monopolistic \ aberration
Sales curves

For 50,000 units over 9 years
Income in a software company is used for:

- **Cost of capital**
  - Dividends and interest  \( \approx 10\% \)

- **Routine operations** -- not requiring IP
  - Distribution, administration, management  \( \approx 40\% \)

- **IP Generating Expenses (IGE)**
  - Research and development, i.e., SW  \( \approx 25\% \)
  - Advertising and marketing  \( \approx 25\% \)

These numbers are available in annual reports or 10Ks
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

\textit{NPV of$}\$1.- \textit{in 9 years at 15% is$}\$0.28$

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

• Selling 50,000 SW units at $500 \approx $1M
  not $25M

Once it's in a spreadsheet, the effect of the many assumptions made can be checked. When assumptions later prove unwarranted then management can make corrections.

To be wise, don't spend more than \approx $500,000 to develop this example software product.
Alternate business model

Consider maintenance and its income
"Service model"

• More assumptions – now include cost
  1. Original cost $516 000 (used to estimate 2.)
  2. Maintenance cost 15%/year of original cost
  3. Maintenance fee 15%/year of original price
  4. Lag = \( \Delta (t \text{ cost, } t \text{ income}) = 2 \) years
  5. Stop maintenance when cost > income
### Effect of service model

<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>Org.cost $K</td>
<td>516</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maint.cost</td>
<td>0</td>
<td>77</td>
<td>129</td>
<td>181</td>
<td>232</td>
<td>284</td>
<td>339</td>
<td>387</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Aggregate</td>
<td>0</td>
<td>77</td>
<td>207</td>
<td>387</td>
<td>619</td>
<td>903</td>
<td>1239</td>
<td>1626</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disc.(lag)</td>
<td>0</td>
<td>102</td>
<td>171</td>
<td>239</td>
<td>307</td>
<td>376</td>
<td>444</td>
<td>512</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>income</td>
<td>0</td>
<td>240</td>
<td>946</td>
<td>1413</td>
<td>1424</td>
<td>1081</td>
<td>661</td>
<td>343</td>
<td>103</td>
<td>21</td>
</tr>
<tr>
<td>Net income</td>
<td>0</td>
<td>137</td>
<td>776</td>
<td>1174</td>
<td>1117</td>
<td>705</td>
<td>218</td>
<td>-170</td>
<td>103</td>
<td>211</td>
</tr>
<tr>
<td>Contribute</td>
<td>0</td>
<td>119</td>
<td>586</td>
<td>772</td>
<td>639</td>
<td>351</td>
<td>94</td>
<td>-64</td>
<td>32</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>2537</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Assume designed for maintenance**

**Cost of maintenance = 1626/(516+1626) = 61% of total**

**Reduce income 1/3 each year**

**Good time to quit**

but $1626 for maintenance

$\approx 2.5$ million
Service model

Analysis shows » profitability in service model

- To achieve such a beneficial model
  1. Management must value maintenance
  2. Marketing and sales must provide feedback
  3. Education and training must recognize the value of maintenance and maintainability
    - Often ignored today
      1. Academics don't teach it  (3/850 pages [Pressman:01])
      2. Companies give maintenance tasks to novices
         - Experienced programmers should maintain their work
Supplier staff roles change

- SW Engineers stay with product
  - increasing application knowledge
  - more specialized
- Marketing and sales personnel provide feedback for perfecting
  - must learn to listen to desires
  - not just talk to impress customers
  - inform engineers
- Management supports IP flow
Related technologies

• Shrink-wrapped software
  – income from periodic version sales
  – similar growth pattern

• Extreme programming (XP)
  – fast turnaround cycle - here for releases/versions
  – customer presence replaced by marketing
  – optional:  
    • egoless programming 
    • shared work -- good backup
    • regular work hours
  – but
    • documentation
Summary

Product Flow model

• Difference after SW initial creation
• Convert maintenance from a liability to an opportunity
• Stable relationship supplier ⇔ customer
• Supports mix of IP at enterprise
  – high cost if change is needed vs 15% annual fee
• Best if deeply ingrained in supplier’s culture
Backup slides
Maintenance as a Strategy

Good BENEFIT/COST \( \text{wrt customer} \)
- Benefit is maximal relevant information, requires constant updating
- Manage cost of change, don’t change system architecture, interfaces

*We expect SW to be adaptable*
- enables change, growth
- long lifetimes
- regular, high maintenance

• Costs
  - initial
  - maintenance (70-90% for SW)

\[ \text{long lifetime} \]
\[ \text{low depreciation} \approx \frac{1}{\text{lifetime}} \]
\[ \text{high maintenance cost} \]
Knowing what software is worth

• Allows rational design decisions, as
  • Limiting development efforts
  • Programming investment for maintenance

• Allows rational business decisions, as
  • Choice of business model
  • Where and when to invest
  • How to assign programming talent

• Improve focus of education in software
  • Consider quality, not just quantity in assignments
  • Effectiveness of curriculum
Observations

• **Software cannot grow exponentially**
  
  no Moore's Law

  **Because**

  1. Cost of maintaining software grows exponentially
     [Brooks:95]
  2. Can't afford to hire staff at exponential $^{*2}$
  3. Cannot have large fraction of changes in a version
  4. Cannot impose version changes on users $< 1 / \text{year}$
  5. Deleting code is risky and of little benefit
     except in game / embedded code
Price $IP = f(income)$

- **Price stays $\approx$ fixed over time**

  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

- **Effect**

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

Unit value

\[ v_1 \]
\[ 0.75 \times v_1 \]
\[ 0.50 \times v_1 \]
\[ 0.25 \times v_1 \]

Assumptions:
\[ IP \approx \text{codesize} \]
deletion \( \approx \text{codesize} \)

Note:
less steep if start with \( V > 1 \), if \( V > 2 \) obeys rule B

Y \( v_1 \) \( Y_{v2} \) \( Y_{v2} \) \( Y_{v3} \) \( Y_{v4} \) \( Y_{v5} \) \( Y_{v6} \)

Years →

Versions →
Growing Systems: \( n \) modules

Federated: deal with many servers and clients

Fast build by resource reuse

changes (x) are difficult affect many clients
Systems with Mediators

Applications . . . .

Mediators . . . . .

Data Resources . . .
Integration originally performed for large systems (> 5Mbyte)
• by system integration companies:
  – Honeywell, IBM federal, Fujitsu, Lockheed, SAIC, Andersen.
Integration now performed for most systems (> 5Mbyte)
• by application clients and services:
  – *too many to name, including you* . . .
New Role for Consultants

Old
• Used at Design Time and
• To Explain Failures

Future
• Available as a Service
• Responsible for Interoperation Maintenance
Technology Transition

- Economic drivers have to be considered.
- Three party model
  - Industry: need-based invention
  - academia: formalization
  - innovators: new technology
- New Service models provide new Opportunities
  - supply innovative tools to industry
  - supply specialized information to industry
Motivation is profit and loss avoidance of

• **Industry**: *investment* --
  – payoff to stockholders / retain value / stable

• **Academia**: *prestige* -- (leads to continuing funding)
  – visibility, not stability or reliability

• **Innovative businesses**: *leverage* -- not sustainable
  – low downside cost, high upside risk,
  – change expected and needed

• **Government research**:
  – technology dissemination & shelving service ?
Notes from Stanford OTL

• SoE inventions
  – 31% licensed / 29% waiting, 40% dropped
  – Large companies poor adopters, best are exclusive to startups
  – Center for Networking has an overall license, pay once + small annual

• New Licensed Field of Use: EPIC program
  – excludes software - but same pricing scheme
  – excluded are already active inventions
  – inventions co-developed outside of SoE
  – inventions not pursued by Stanford
  – base cost for membership 400K or 100K for 5 years
  – 100K non-exclusive license per inventions before patent is granted
  – 200K non-exclusive license per invention after patent is granted
  – easy access for 6-months
  – SOE gets base, can distribute to depts. Inventor gets license fees.
  – objective is better relationships to large companies
  – not suitable for startups, small companies
Research economy transfer paths

Tool suppliers (TS) versus Product suppliers (PS)

- High-value
- Modest volume

Customers
- High volume
- Taxes

Research
- People
- Results

Teaching

Government
Summary

- Maintenance is a high cost item
- Cost is incurred over many years
  - if the product is successful
  - often by different people, org’s
- Should be planned for
  - in architectural decisions
  - in responsibility assignment

Good maintenance has a high value and high leverage