The Product Flow Model

Gio Wiederhold
Stanford University
Oct 2004

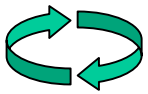
Presentation Layout

- **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**

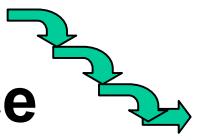
Varied Software Models

Guide the process of SW creation

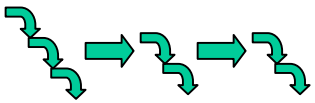
- **By individuals for themselves**
  - **Dykstra model**



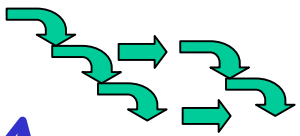
- **By IT departments for their enterprise**
  - **Waterfall model / Spiral model / Watersluice**



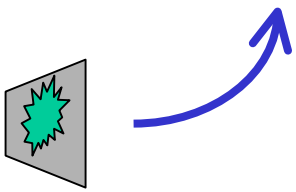
- **By software supplier for individuals**
  - **Shrinkwrapped, ~ biannual versions**



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



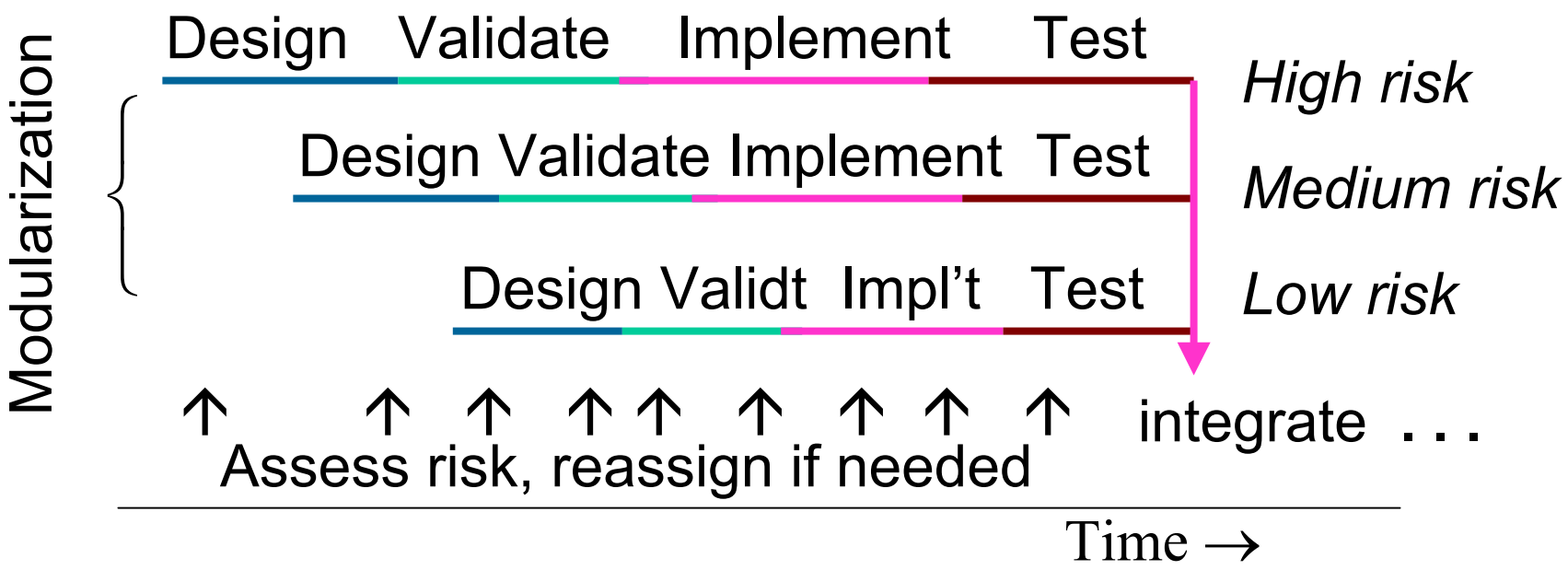
- **Start-up enterprises**
  - **See if it sticks. If it does, convert**



* Water Sluice [Burback 2000]

Risk minimizing approach

- **Develop highest risk modules first**
  - **avoid problem of spiral approach**



❖ Product Line Approach

Strategy for a software supplier

- Specialize in some domains
- Invest extra in each product delivered

Value embodied

❖ in reusable software modules

and

❖ 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

← *schedule* →

← *feasible size* →

← *in house* *external* →

← *short life* *long life* →

← *responsiveness* *contracts* →

Outsourcing to remote service

↑

← **choices** →

Outcome: IP

Delivery of Intellectual Property (IP)

- Some SW embodies company competence.
  - **Don't cede control: best kept in house!**
- Remaining *commodity* SW **Awkward mix**
  - **Purchased for in-house use or service?**

Over time the inventory of SW grows

- **Internal development + purchased software**

- Has to be maintained → growth

Problem

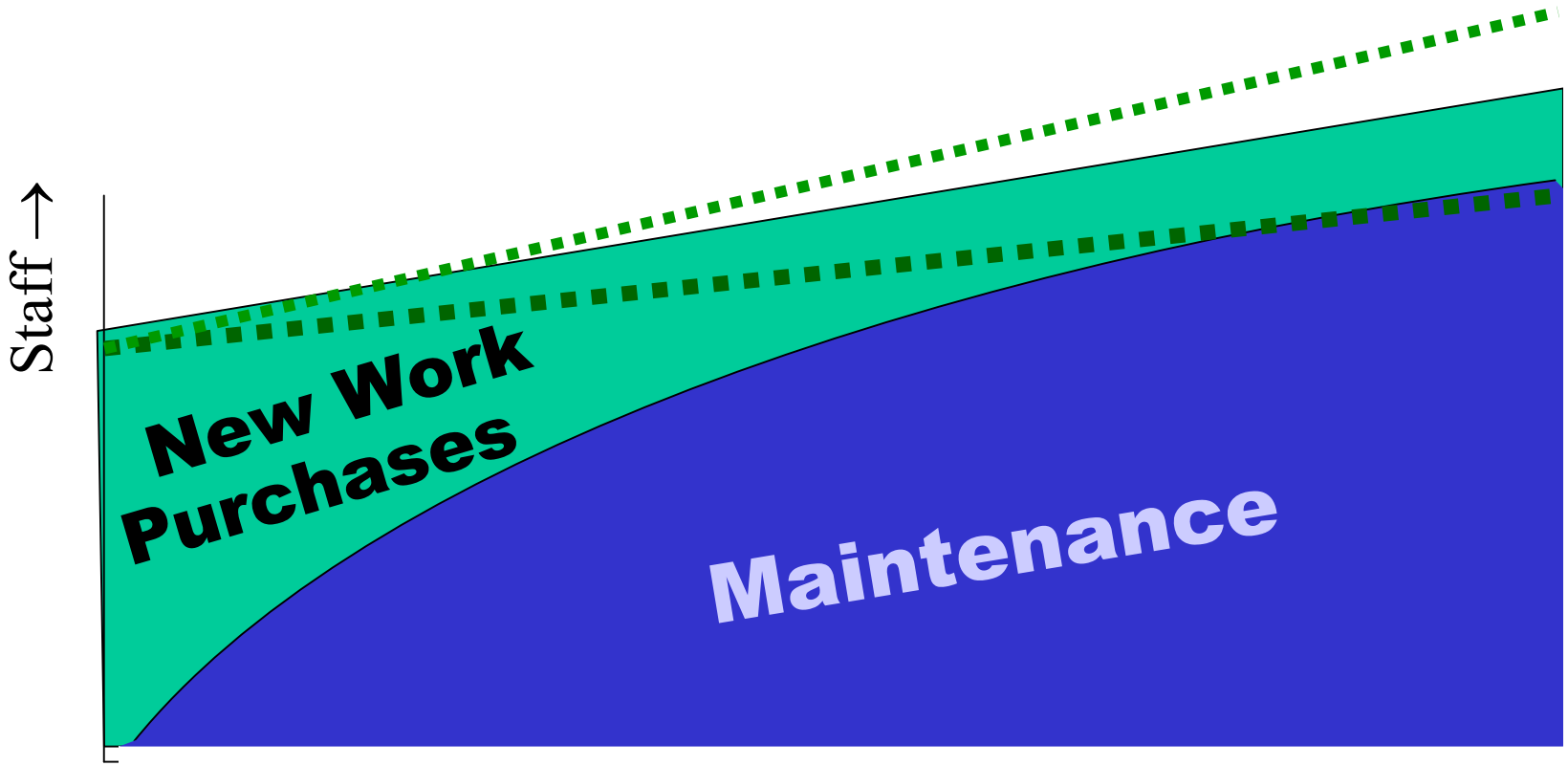
As the software inventory grows

- 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 ≈ SWsize ^{2} **[Brooks:95]**

Time →

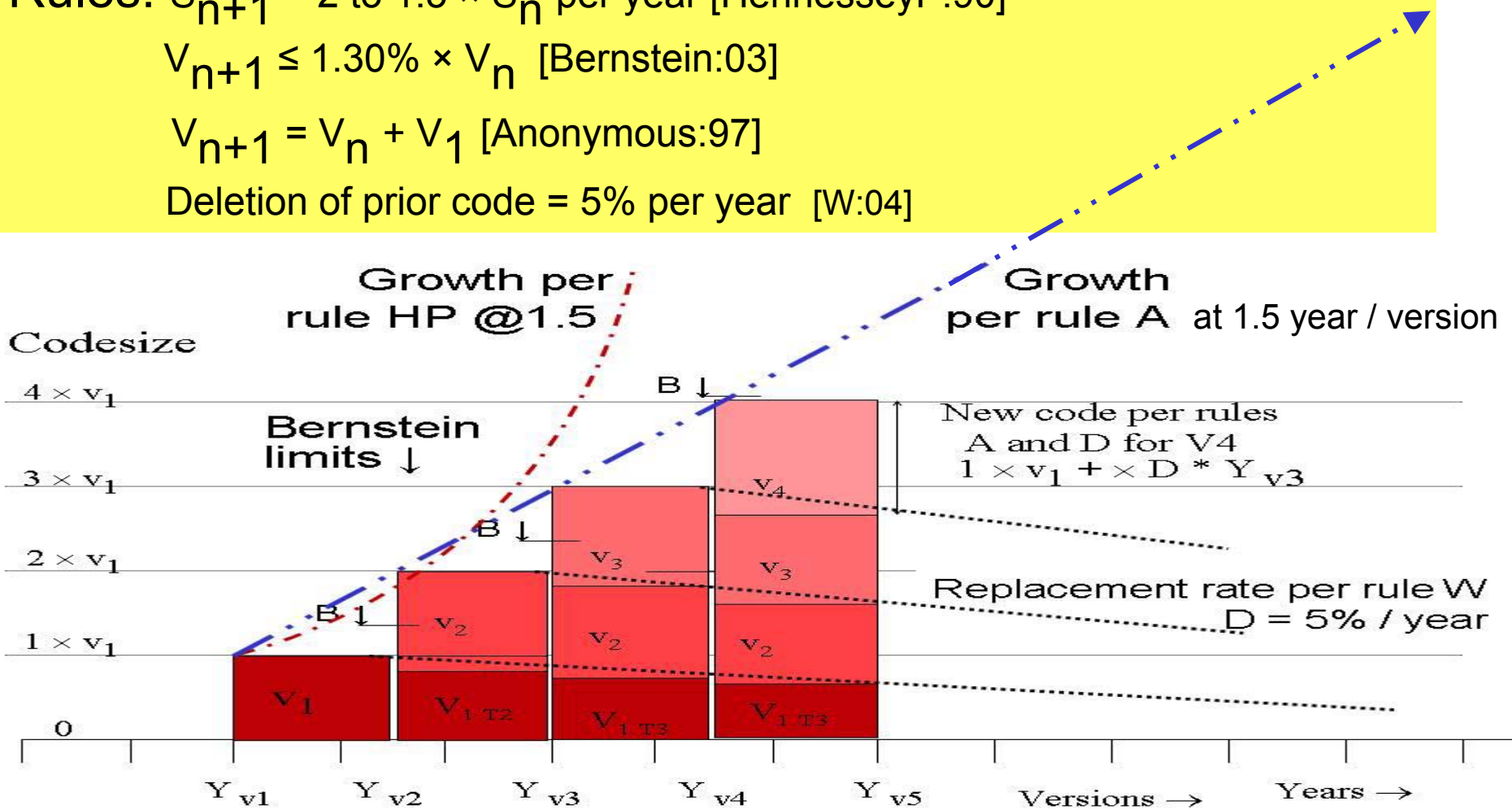
Quantifying SW Growth

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]

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

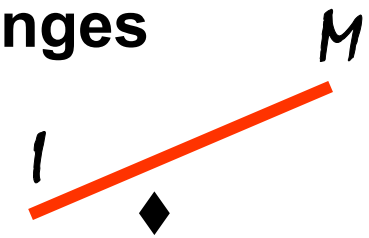
Deletion of prior code = 5% per year [W:04]



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 loose 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**

Long term Staff Effort

Software is slithery !

Continuously updated

1. Corrective maintenance

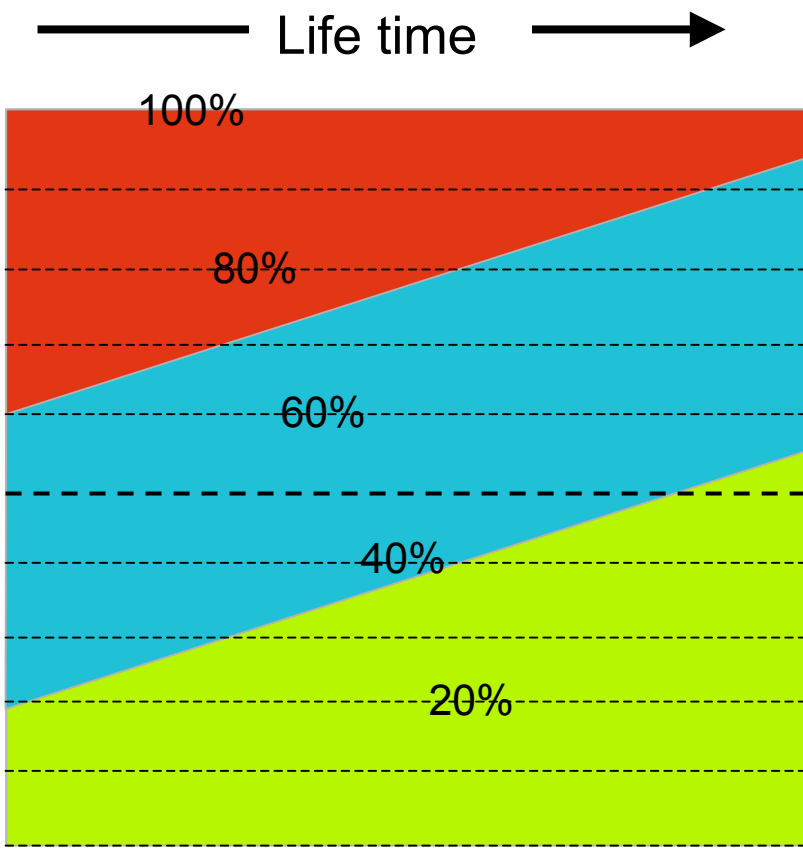
   bugfixing reduces for good SW

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 →
  - **Ignore**
  - **Fix in next release**
  - **Patch now**

- Adaptive **from Infrastucture suppliers**
  from standards orgs
  from governments

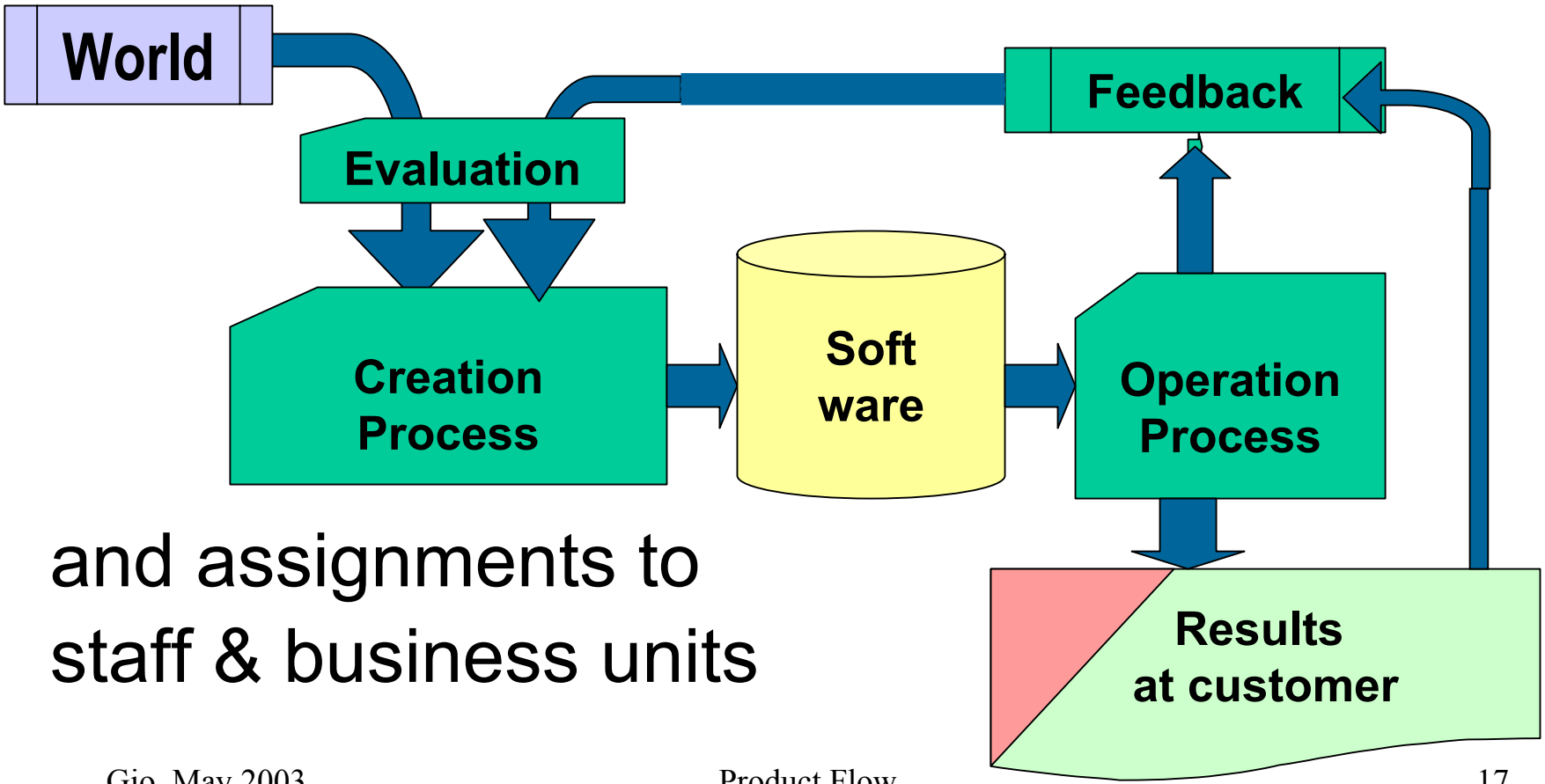
  Monitor → Validate / Classify →
  - **Ignore**
  - **Fix in next release**
  - **Patch now**

- Perfective **from customer via marketing**

  Evaluate cost/benefit → Assign →
  - **never / future**
  - **next version**
  - **next release**

Work flow methodology

Consider inputs, outputs that link processes



and assignments to staff & business units

Product Flow Approach

Extend software product line to

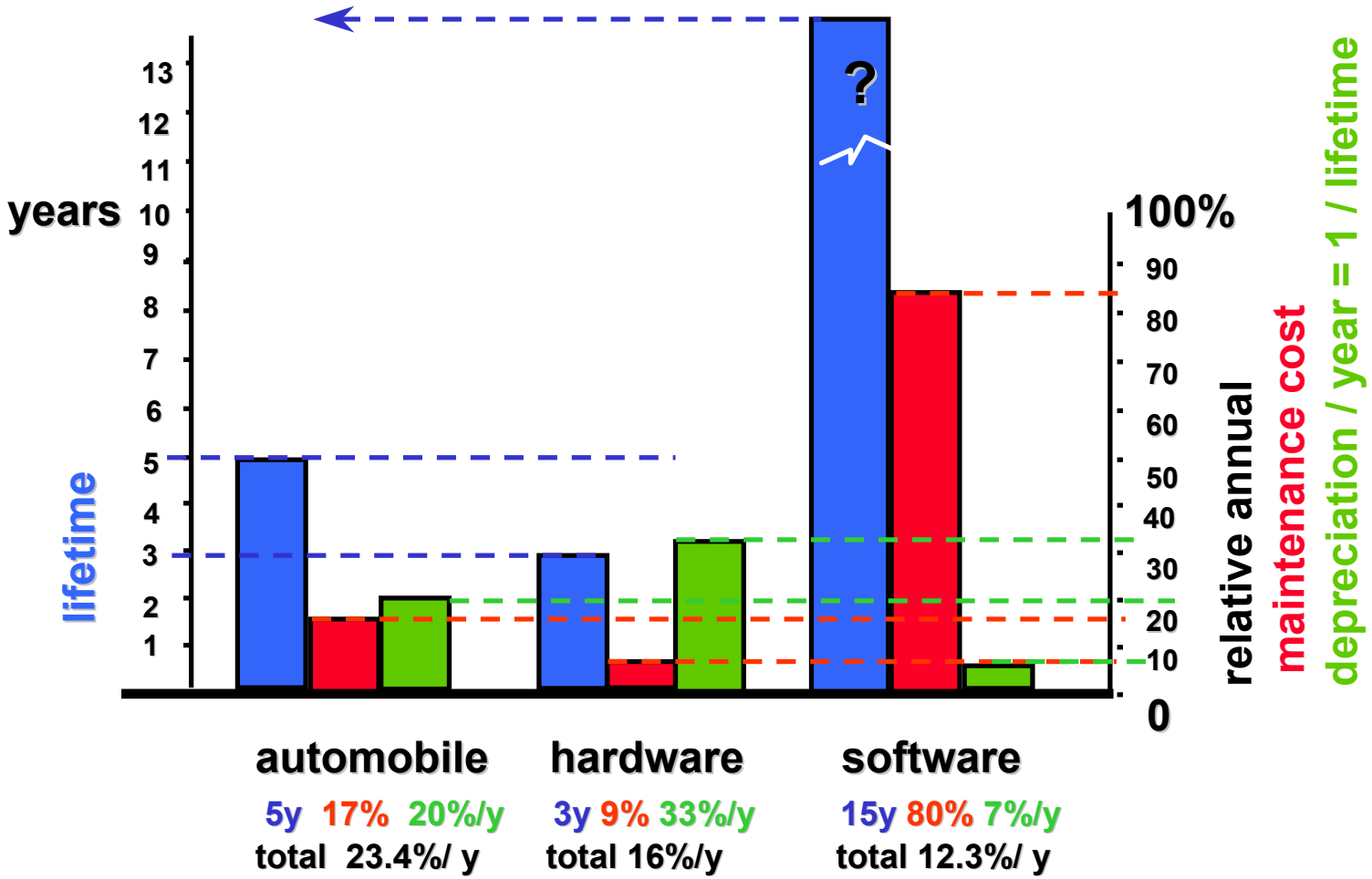
- **include maintenance (the high cost portion)**
- **the feedback that motivates maintenance**

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



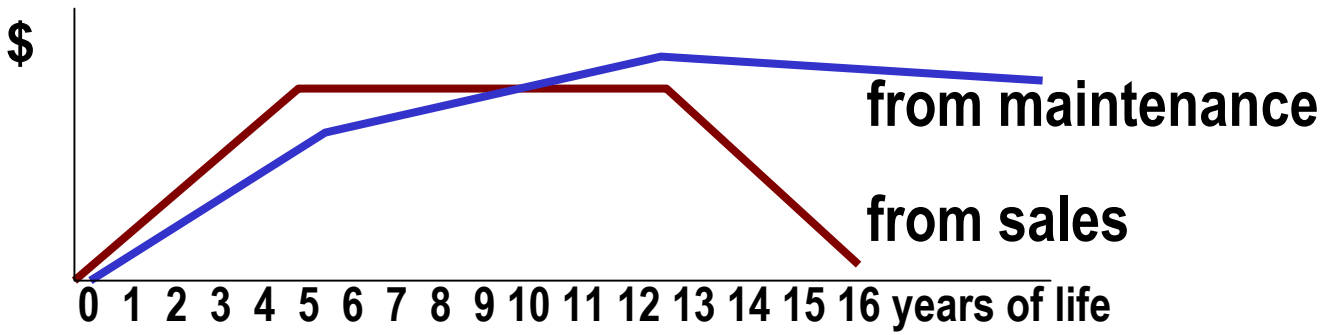
Value at the enterprise

- 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 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% $\sum^{n} S_i$
   - **supports basic service and engineering: 50/50?**
   - **steady (95%of customers renew annually)**



Total income

Total income = price × volume (year of life)

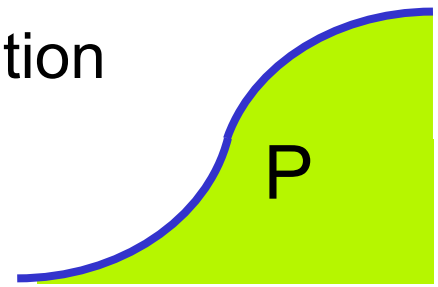
- Hence must estimate volume, lifetime

Best predictors are Previous comparables

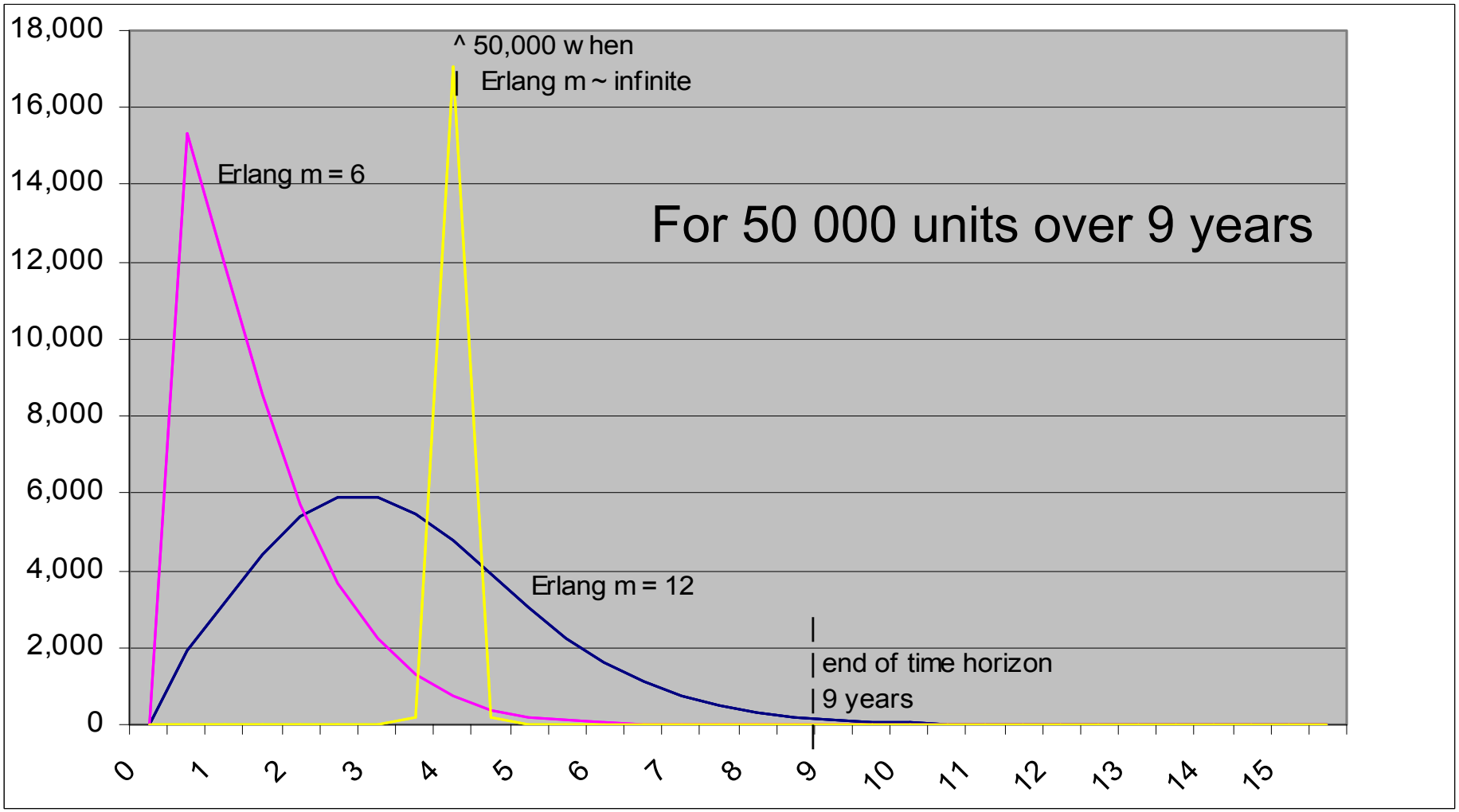
- Erlang curve fitting (m=6 to 20, 12 is typical)

and apply common sense limit = Penetration

- estimate total possible sales F × #customers
- above F= 50% monopolistic aberration



Sales curves



Fraction of income for SW

Income in a software company is used for

- Cost of capital *typical*
  - Dividends and interest ≈ 10%
- Routine operations -- not requiring IP
  - Distribution, administration, management ≈ 40%
- IP Generating Expenses (IGE)
  - Research and development, i.e., SW ≈ 25%
  - Advertising and marketing ≈ 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×(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

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	1032	≈ $ 1 million								

Result of Example

- Selling 50 000 SW units at $500 ≈ $ 1M
  not $ 25M

Once its 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 ≈ $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 = Δ (*t* cost , *t* income) = 2 years
5. Stop maintenance when cost > income

Effect of service model

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
Org.cost $K	516									
Maint.cost	0	77	129	181	232	284	339	387	0	0
Aggregate	0	77	207	387	619	903	1239	1626	*not discounted*	
Disc.(lag)	0	102	171	239	307	376	444	512	0	0
income	0	240	946	1413	1424	1081	661	343	103	21
Net income	0	137	776	1174	1117	705	218	-170	103	211
Contribute	0	119	586	772	639	351	94	-64	32	6
Total	2537	≈ $ 2.5 million					but $ 1626 for maintenance			

Assume designed for maintenance

Good time to quit

Reduce income 1/3 each year

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

typical

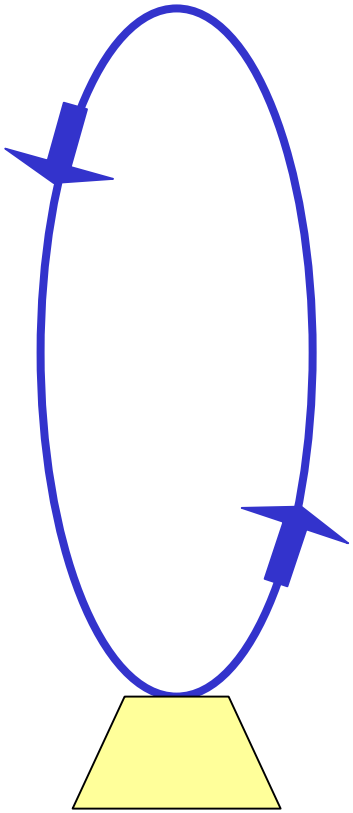
Service model

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 ***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 adapatable

- **enables change, growth**
- **long liftetimes**
- **regular, high maintenance**

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

long lifetime

low depreciation ≅ 1 / lifetime

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 assignements
  - Effectiveness of curriculum

Observations

- <u>Software cannot grow exponentially</u>

  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 / year
  5. Deleting code is risky and of little benefit

     except in game / embedded code

Price *remember IP =f(income)*

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

  like hardware Moore's Law

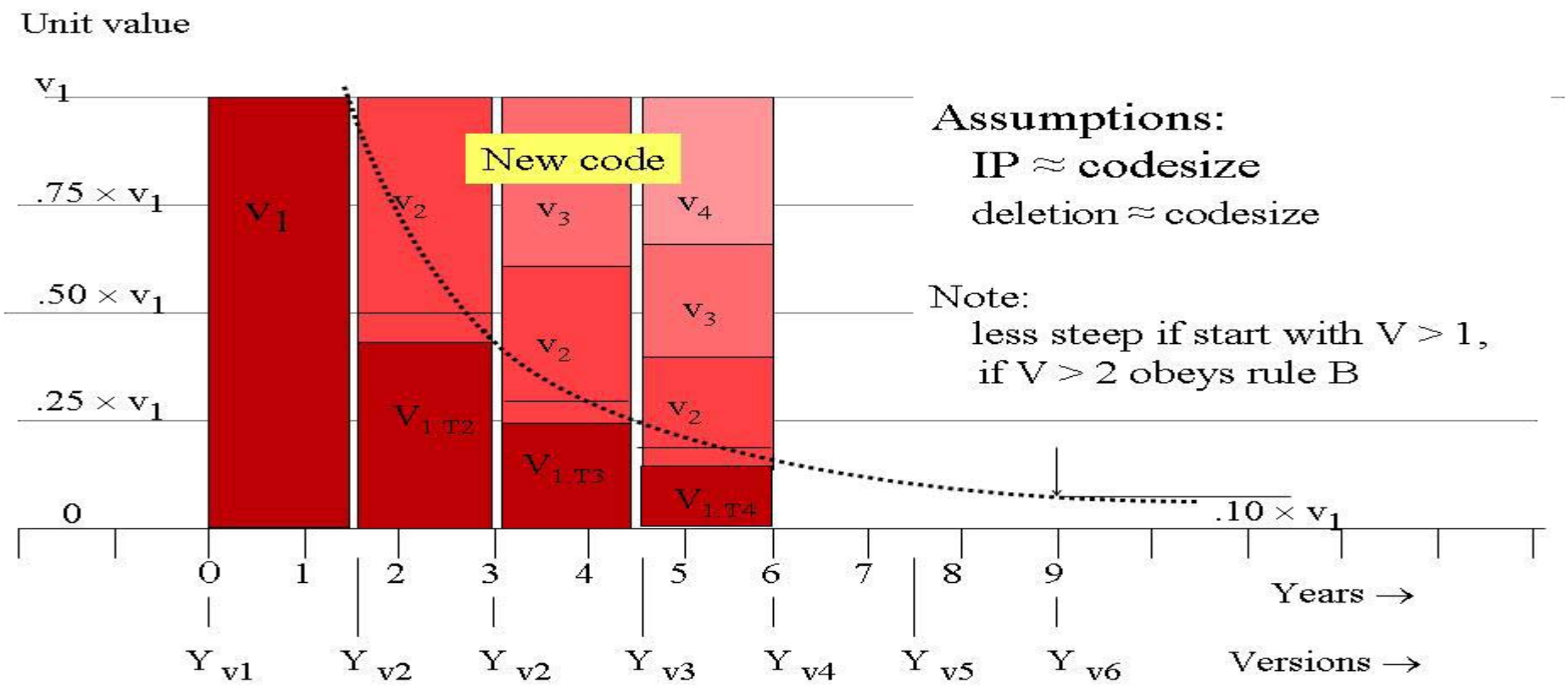
  Because

  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>

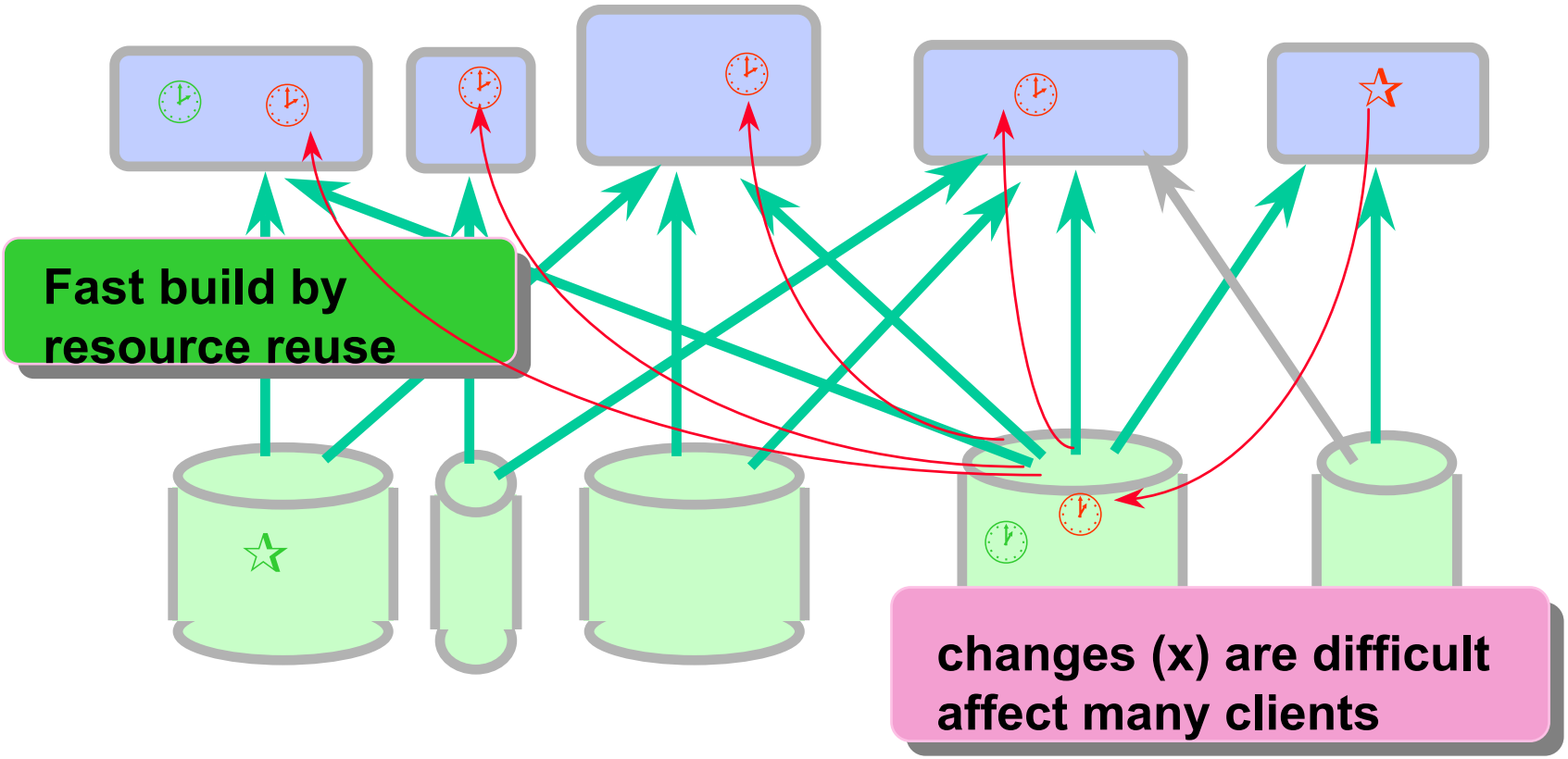
The income per unit of code reduces by 1 / size

Growth diminishes IP



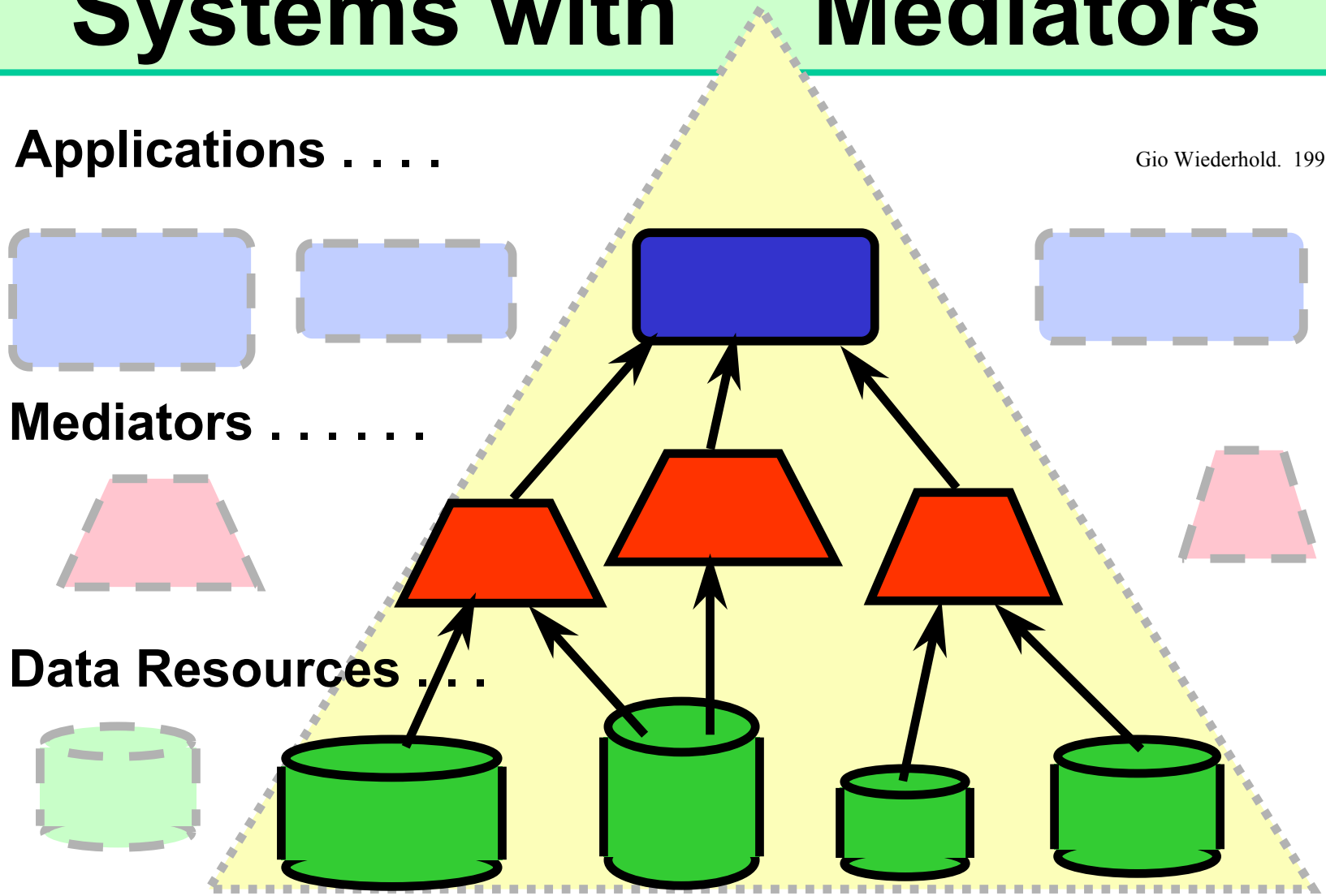
Growing Systems: *n* modules

Federated: deal with many servers and clients

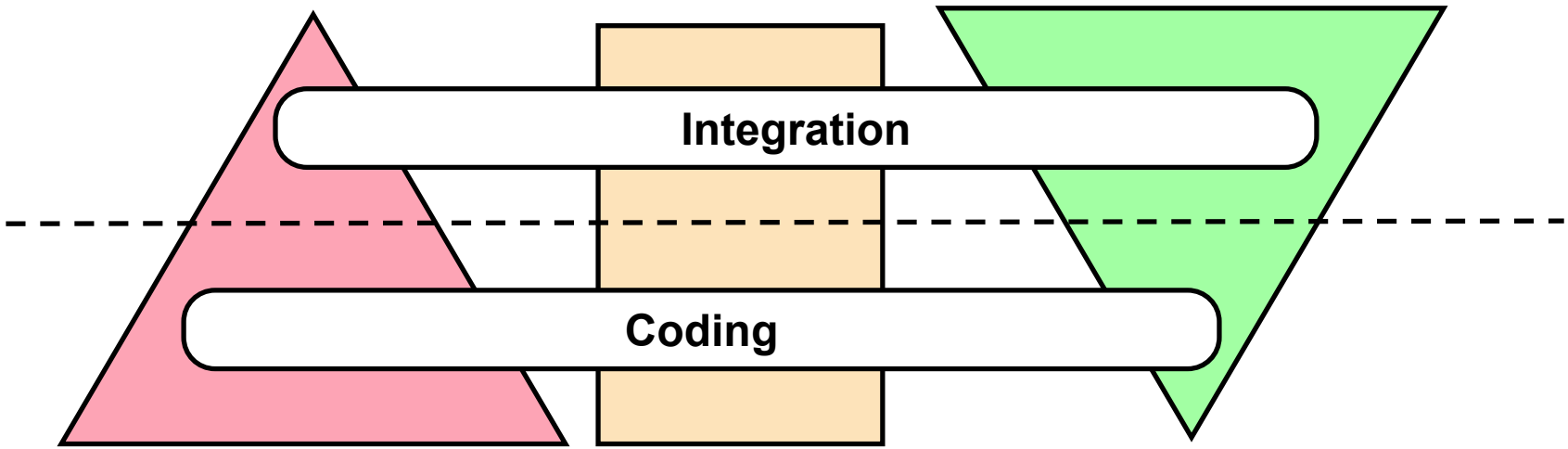


Systems with Mediators

Gio Wiederhold. 1995



Workforce reallocation



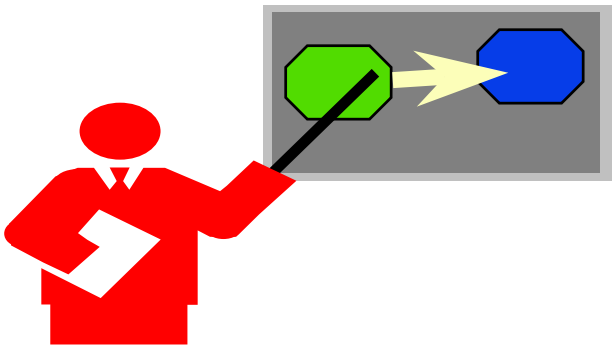
Integration originally performed for large systems (> 5Mbyte)

- 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

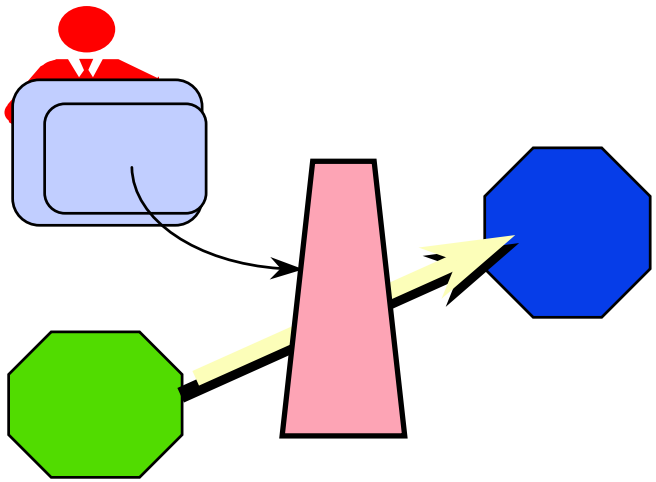


Old

- Used at Design Time

  and

- To Explain Failures

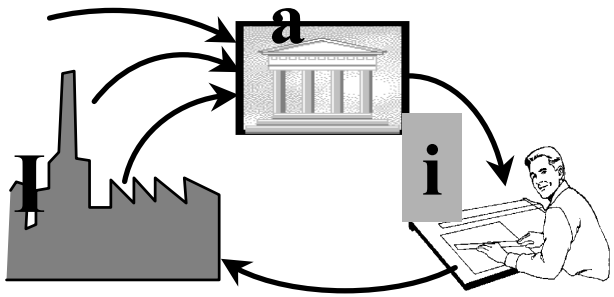


Future

- 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

Understanding the other parties

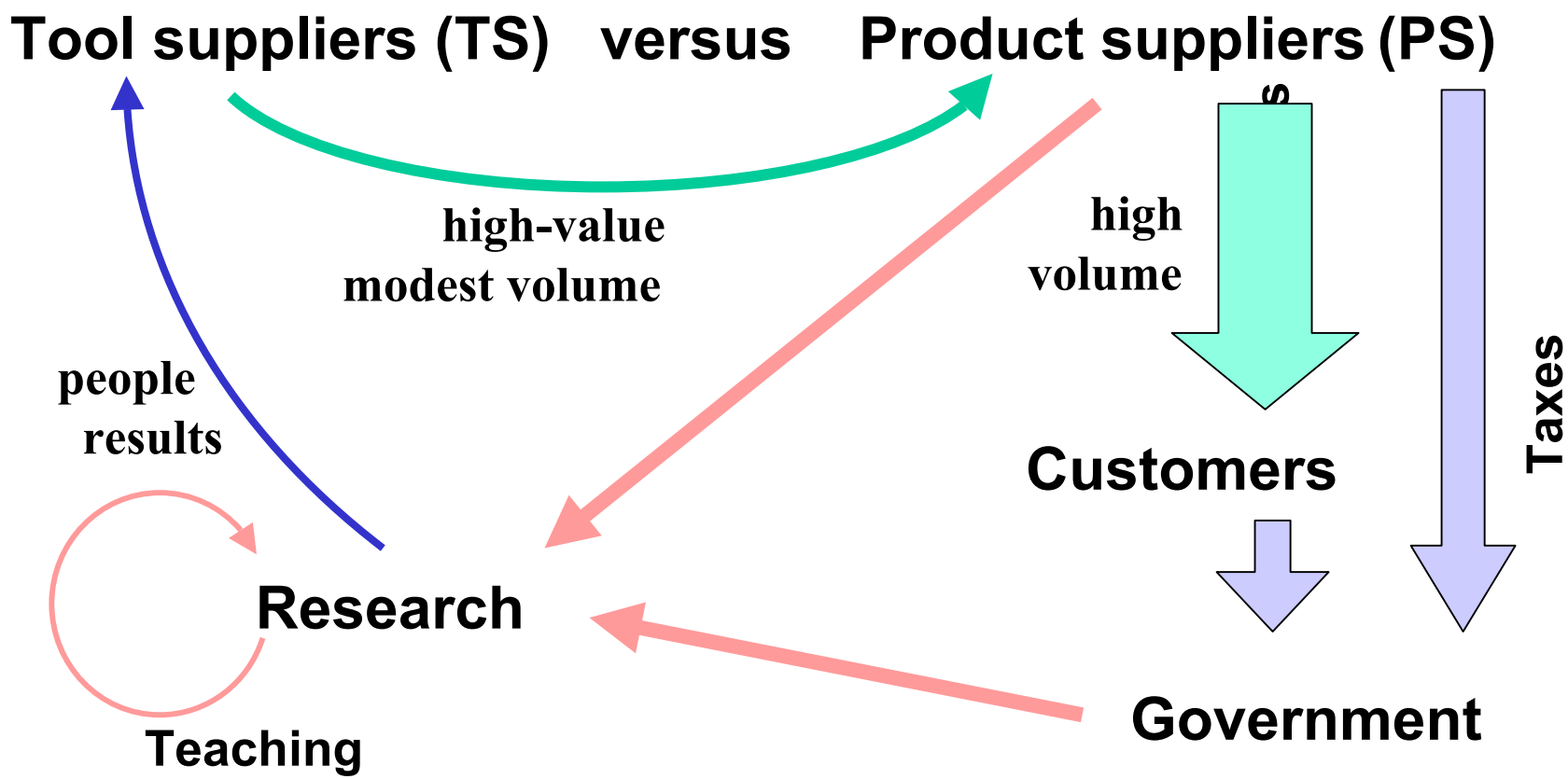
Motivation is profit and loss avoidance of

- 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 overll 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



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