

Foundations of Empirical Software Engineering: Legacy of Vic Basili – SEL, Experience Factory

Michael A. Cusumano

MIT Sloan School of Management, © 2005

Cambridge, MA USA

cusumano@mit.edu

1

Bit of a Personal Chronology

- **1984** article by Zelkowitz...Basili et al, "Software Engineering Practices in the US and Japan"
- **1985-91** – Japan's Software Factories (1991) study
- **1990-1991** – Motorola consulting (Vic – metrics, "Experience Factory," quality; MC comparisons with Japanese software factories; Motorola India in 1994)
- **1991** – Software Quality Improvement, Tokyo
- **1990-95** – part-time in D.C., visitor at Maryland (1994); Microsoft Secrets (1995, with R. Selby)
- **1995** – Basili & Caldiera *Sloan Management Review* article on Knowledge Reuse and Experience Factory

2

Particular Influences

- **Japan-US Comparisons:** Search for “best practices”
- **Metrics:** No metrics are perfect, but measure what you can (process, product, performance)
- **Experience:** Knowledge, like code and designs, can be packaged and reused; essence of “factory” idea
- **Strategy:** Matching process with product/project objectives (SEL-GQM, type and intensity of quality processes, iterative development, business models)
- **Import of Management:** Kindness to a novice, non-CS, from a business school... *Software engineering is as much about “management” as “technology”*

3

Recent Work

- Alan MacCormack, Chris Kemerer, Michael Cusumano, and Bill Crandall, “**Trade-offs between Productivity and Quality in Selecting Software Development Practices**”, IEEE Software, September-October 2003
- Michael Cusumano, Alan MacCormack, Chris Kemerer, and Bill Crandall, “**Software Development Worldwide: The State of the Practice**”, IEEE Software, November-December 2003
- M. Cusumano, **The Business of Software** (Free Press/Simon & Schuster, 2004)

4

Spectrum in Process Philosophies

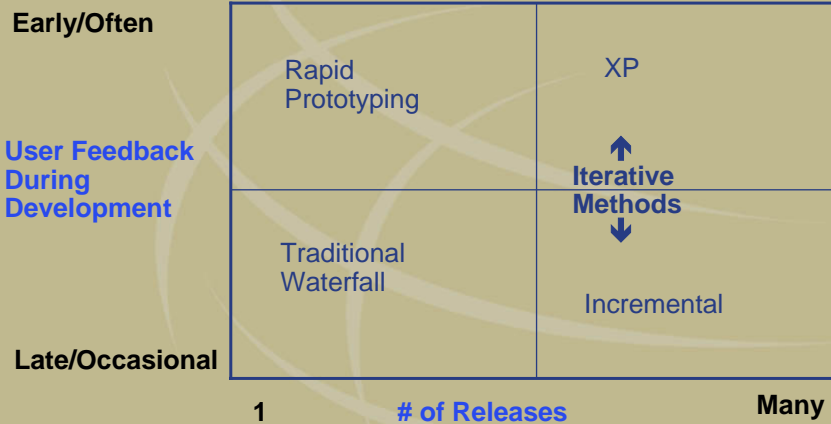
- **Waterfall-style (sequential, “stage-gate”)**

versus

- **Iterative-style (flexible, “agile”)**
 - Spiral
 - Rapid Prototyping
 - Synch-and-Stabilize (Microsoft)
 - Rational Unified Process & Tool-kit
 - HP’s Evo Process (short cycles of mini-waterfalls)
 - Agile and many other variations at companies
 - Extreme Programming (XP)

5

Spectrum of Process Approaches



Adapted from Bill Crandall (HP)

6

Tradeoffs between Productivity and Quality (HP pilot study)

- Case studies and pilot survey, in 2000-2001
- 35 project responses, 29 projects with complete data
- Median system size – 170K LOC, with 70K new code
- Median 9-person team, 14 month project
- 59% applications, 38% systems, 28% embedded
- Bias: Probably above-average projects due to data requirements of survey and self-reporting.
- But data seem more “reliable” than later international sample due to more precise data collection by research team member who is an HP manager.

7

HP Sample General Observations

- Median defects reported by customers in 12 months after shipment = 7.1/million LOC (or **.007/1000 LOC**)
- Median output per programmer 17.6 LOC/person-day (or **352/person-month**)
- Slightly more defects reported in systems projects
- **Fewer defects and higher programmer output in larger projects with more complete design specs before coding, and design reviews (i.e., waterfall-ish projects)**
- **Higher productivity and fewer defects with early prototypes (i.e., more iterative-style)**
- **Some reduction in defects and higher output associated with subcycles & regression testing at each check in (i.e., more iterative-style)**

8

HP Multivariate Regression Analysis

- **74% of variation in defects explained by early prototypes, design reviews, and integration/regression testing**
- Median project had 40% of functionality complete when first prototype released to customers and 35.6 defects per million (.04/1000) LOC, reported by customers in 12 months after release, and 18 LOC per person day (360/month)
 - Releasing prototype earlier with 20% of functionality = **27% reduction in defect rate (compared to median project)**
 - Integration/regression testing at each code check-in = **36% reduction in defect rate**
 - Design reviews = **55% reduction in defect rate**
 - Releasing prototype with 20% of functionality = **35% rise in LOC output/programmer**
 - Daily builds = **93% rise in LOC output/programmer**

9

Observations from the HP Survey

- **Best “nominal” quality** from traditional “waterfall” process (fewer cycles and late changes, regression tests on each build = less bugs)
- **Most flexibility** to adapt to changes during a project using a bundle of iterative techniques (early prototypes, subcycles, regression tests on each build)
- **Best balance** of speed, flexibility, and quality from combining conventional and iterative techniques:
 - Short development subcycles (subprojects/milestones)
 - Early prototypes to get customer feedback
 - Frequent builds to incorporate feedback, changes
 - Frequent design/code reviews (check quality continuously)
 - **Regression tests on each build (check for errors, late changes, integration problems)**

10

Software Development Worldwide

- **Survey:** Completed in 2001-2003, with Alan MacCormack (HBS), Chris Kemerer (Pittsburgh), and Bill Crandall (HP)
- **Objective:** Determine usage of Iterative (Synch-and-Stabilize) versus Waterfall-ish techniques, with performance comparisons
 - 104 projects plus 29 from HP-Agilent for pilot survey
- **Participants**
 - **India:** Motorola MEI, Infosys, Tata, Patni
 - **Japan:** Hitachi, NEC, IBM Japan, NTT Data, SRA, Matsushita, Omron, Fuji Xerox, Olympus
 - **US:** IBM, HP, Agilent, Microsoft, Siebel, AT&T, Fidelity, Merrill Lynch, Lockheed Martin, TRW, Micron Tech
 - **Europe:** Siemens, Nokia, Business Objects

11

International Sample Characteristics

- 104 of 118 projects with relatively complete data
- Type: 20 systems, 20 applications, 56 custom, 8 embedded
- Reliability: 32 High, 66 Medium, 6 Low
- Platform: 12 mainframe, 66 w/s, 15 PC, 10 other
- Customer: 5 Individual, 84 Enterprise, 14 In-house
- **Bias:** Probably above-average projects due to data requirements of survey and self-reporting. Data unclear, due to self-reporting...

12

	India	Japan	US	Europe et al	Total
# of projects	24	27	31	22	104
System	7	5	4	4	20
Application	4	4	7	5	20
Custom	11	16	19	10	56
Embedded	2	2	1	3	8
Rel. High	8	12	8	4	32
Medium	14	14	20	18	66
Low	2	1	3	0	6
Mainframe	2	6	3	1	12
Workstation	16	16	19	15	66
PC	3	4	7	1	15
Other	3	1	1	5	10
Individual	0	1	2	2	5
Enterprise	23	23	21	17	84 ¹³

“Conventional” Good Practices Used

	India	Japan	USA	Europe etc	Total
Number of Projects	24	27	31	22	104
Architectural Specs %	83%	70%	55%	73%	69%
Functional Specs %	96%	93%	74%	82%	86%
Detailed Design %	100%	85%	32%	68%	69%
Code Generators -- Yes	63%	41%	52%	55%	52%
Design Reviews -- Yes	100%	100%	77%	77%	88%
Code Reviews -- Yes	96%	74%	71%	82%	80%

“Newer” Iterative Practices Used

	India	Japan	USA	Europe etc	Total
No. of Projects	24	27	31	22	104
Subcycles -- Yes	79%	44%	55%	86%	64%
Beta tests -- Yes	67%	67%	77%	82%	73%
Pair Testing -- Yes	54%	44%	35%	32%	41%
Pair Programmer -- Yes	58%	22%	36%	27%	35%
Daily Builds at project start	17%	22%	36%	9%	22%
In the middle	13%	26%	29%	27%	24%
At the end	29%	37%	36%	41%	36%
Regression test each build	92%	96%	71%	77%	84%

15

“Crude” Output Comparisons

		India	Japan	USA	Europe & Other	TOTAL
Projects		24	27	31	22	104
LOC/ Month	median	209	469 cf. 389 in 1990	270 cf. 245 in 1990	436	374
Bugs/ 1000 LOC	median	.263	.020 cf. .20 in 1990	.400 cf. .80 in 1990	.225	.150

16

International Sample Observations

- **US:** Most PC, workstation, and in-house projects; “weakest” in conventional good techniques (documentation, design and code reviews); second-most waterfallish (one-cycle) projects; most use of daily builds; **most customer-reported defects**
- **Europe et al:** slightly more enterprise customers, and other platforms; more mix of “conventional” and “iterative” practices -- strong on specs and reviews, with subcycles and regression testing on each build; but less daily builds; **low defects but low programmer output**

17

International Sample Observations

- **Japan:** slightly more custom, high-reliability, mainframe, **most one-cycle (waterfall) projects**; strong on conventional good techniques (specs, reviews) and some iterative (regression testing on each build); **by far fewest customer defects, highest programmer output**
- **India:** slightly more enterprise customers, **more mix of “conventional” and newer “iterative” practices** -- strong on specs, reviews, subcycles, and regression testing on each build; pair testing and programming; **low defects but low programmer output**

18

Some Conclusions from Int'l Sample

Most projects (64%) not pure waterfall, but 46% were!

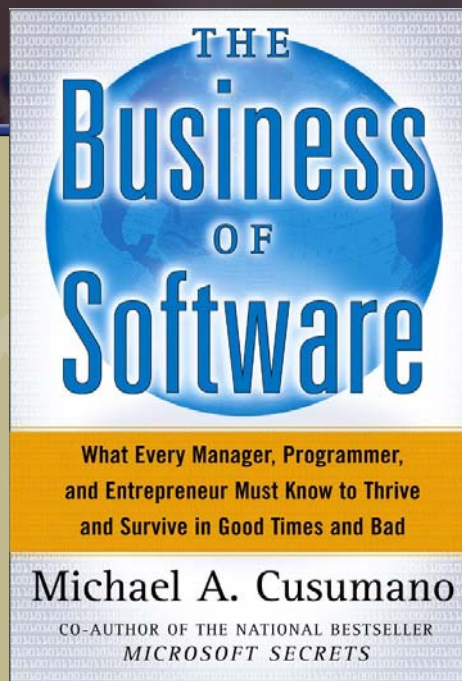
Mix of “conventional” and “iterative” common -- use of functional specs, design & code reviews, but with subcycles, regression tests on frequent builds

Customer-reported defects seem to have improved a lot in the past decade in US and Japan; LOC “productivity” may have improved a little, but unclear

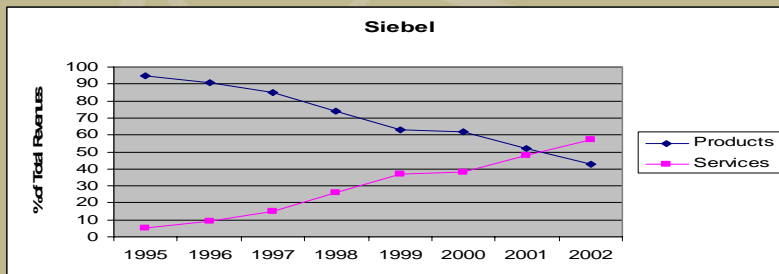
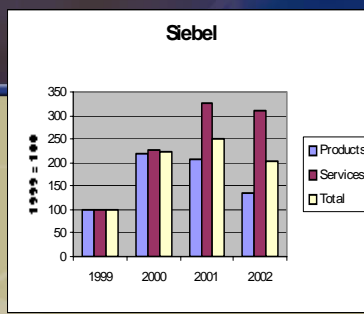
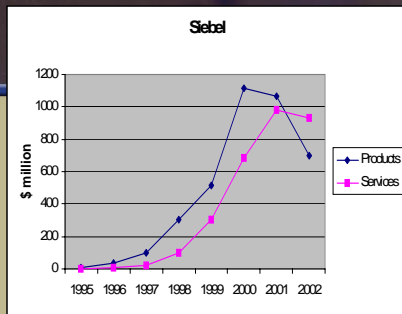
Japanese projects still report best quality, but what does this mean? Preoccupation with “zero defects”?

Indian projects look strong in process and quality, but not as strong as CMM Level 5 ratings suggest??

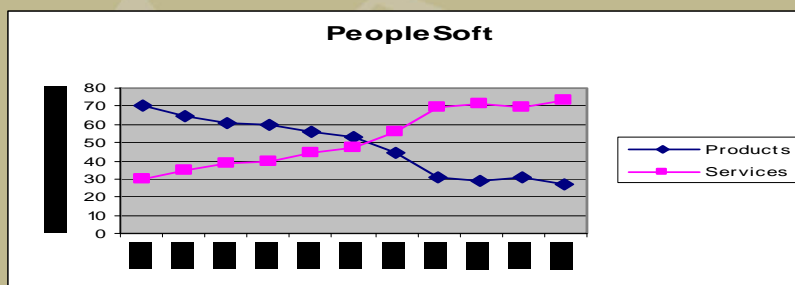
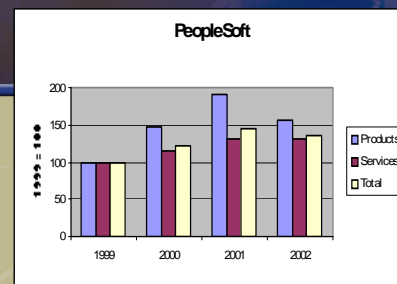
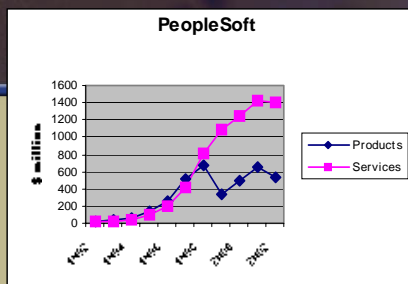
19



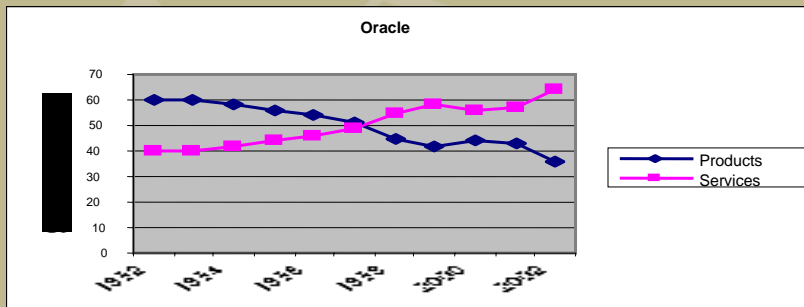
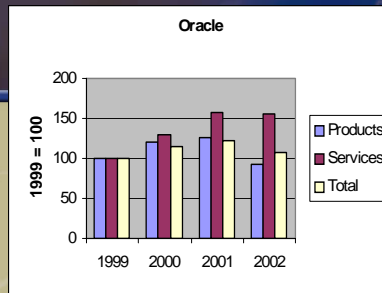
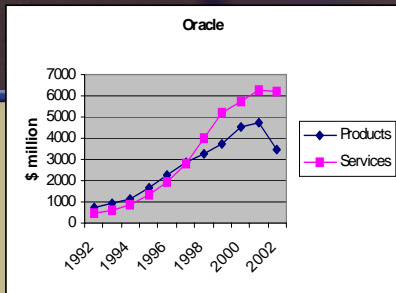
20



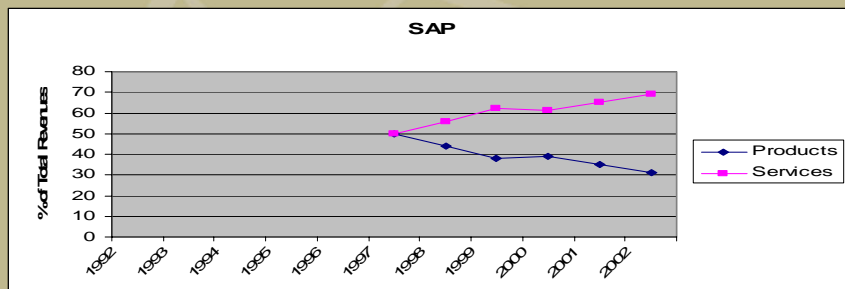
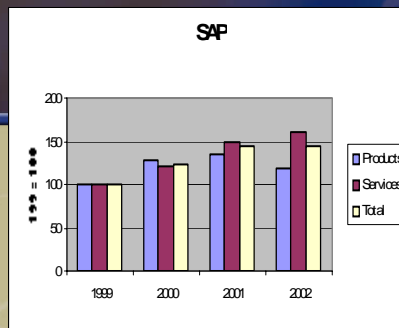
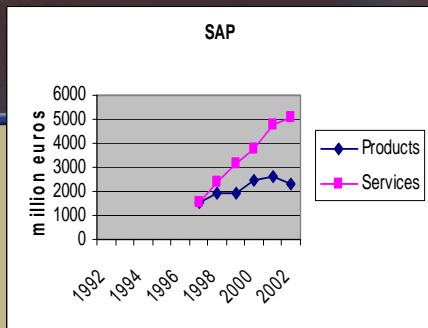
21



22

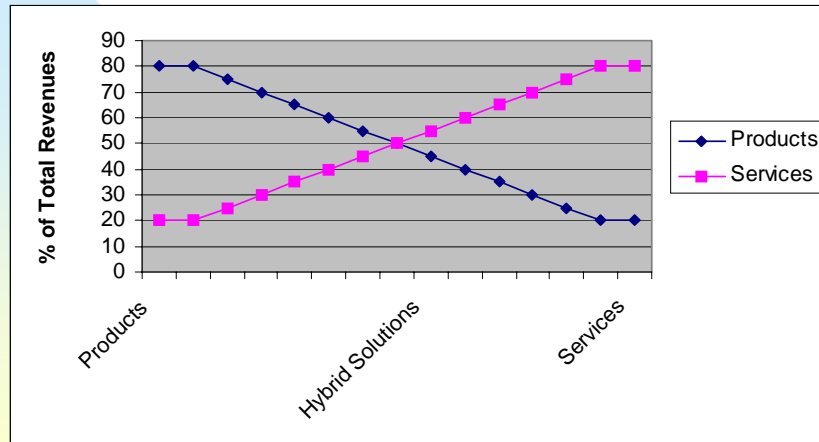


23



24

Three Business/Life Cycle Models



7

Software Business Models: Initial Research Questions

- **Products** not as good as previously thought?
 - “99% of 0 = 0,” especially in down economies?
 - “Platforms products” and some niche products do better?
 - **But new products drive services & maintenance?**
- **Services** not as bad as previously thought?
 - Can **double or triple** a firm’s sales and profits?
 - Services help create **stickier** product solutions?
 - **But maintenance much better than other services?**
- **Hybrid** the “best” business model?
 - Most stable performance & strategy re commoditization?
 - **But requires most complex combination of skills?**

26

Strategic & Operational Challenges

- **How Manage the Revenue “Crisscross”?**
 - Best balance of products, services & maintenance?
 - New products to generate services & maintenance?
- **How “Servitize” Products?**
 - Add special value and revenue opportunities?
 - Make products “stickier,” less commodity-like?
- **How “Productize” Services?**
 - Create two organizations within one?
 - Offer “one-off” solutions more efficiently -- standardizing components, leveraging knowledge, tools, or processes across customers (“scope”)?

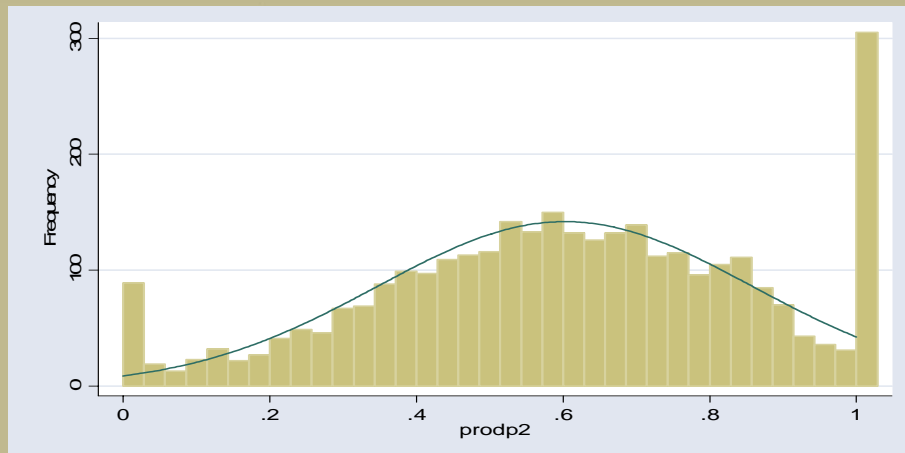
27

New Database Analysis

- Identified 463 public software “products firms” under SIC code 7372 – **PrePackaged Software**
- Financial information from Mergent Database & 10K reports. Avg. 9, maximum 15 years of detailed financial information, from firms listed in 1995 or later.
- 3386 total yearly observations (4198 including no-breakout firms).
- Now doing **exploratory** analysis
- Also starting database of **non-software** firms

28

Annual % Product Revenues by Firm (374 firms, 3386 yearly observations)



Notes: -- Excludes 89 firms with no sales breakout and unclear status.
 -- 1 (100%) includes some product firms that did not break out revenue mix (MSFT, Adobe, SPSS, Visio, Symantec, and Fair Issac, and game software firms).

29

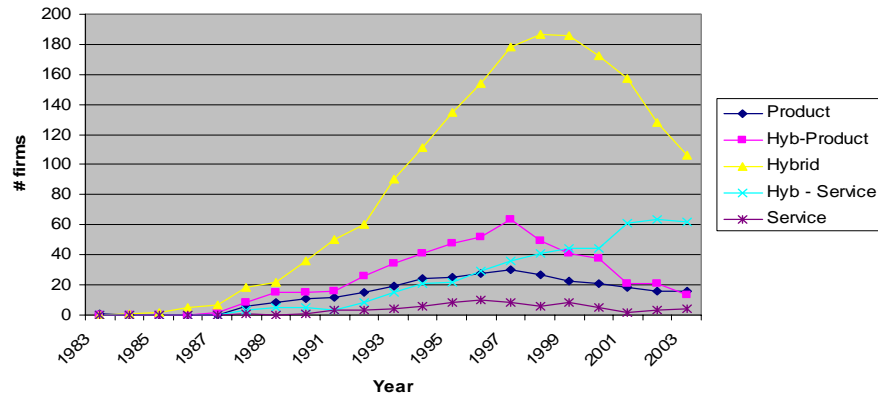
Preliminary Data Analysis

- Broke out hybrid using standard deviation. Distribution approximated normal. Used 1 standard deviation to calculate the middle group. The mean is .57 and standard deviation is .216.
 - **HybridServices** = product sales % > 0 but < .359
 - **HybridBalanced** = product sales \geq .359 but \leq .791
 - **HybridProducts** = product sales % > .791 but < 1
- Total observations for the 5 groups:

Services:	72
Product:	300
HybridS:	463
HybridB:	1805
HybridP:	<u>504</u>
Total:	3144

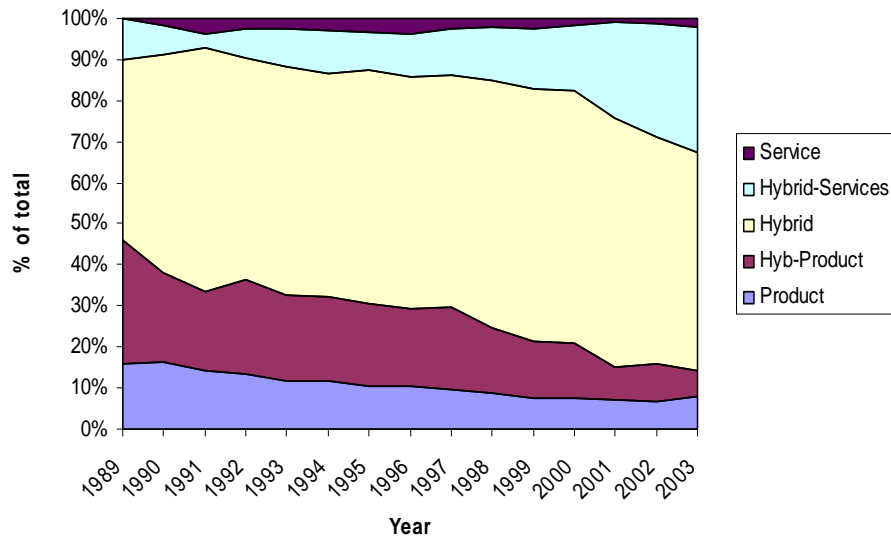
30

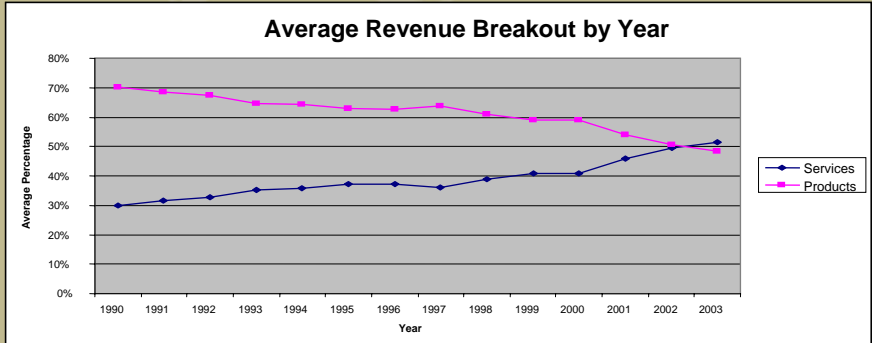
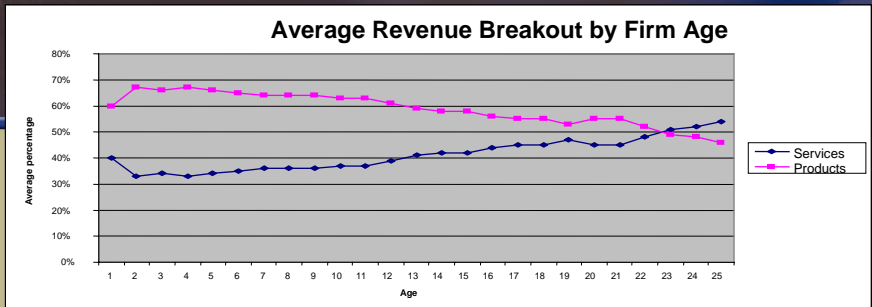
Business Model by Year



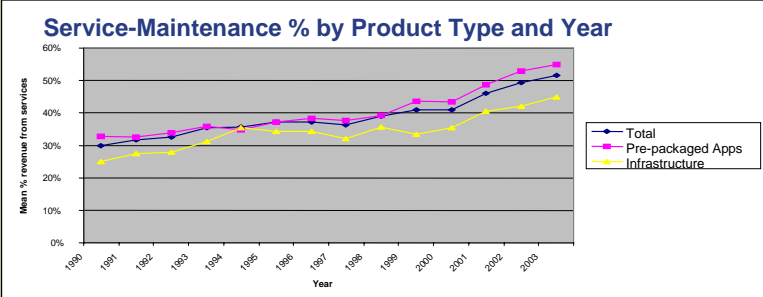
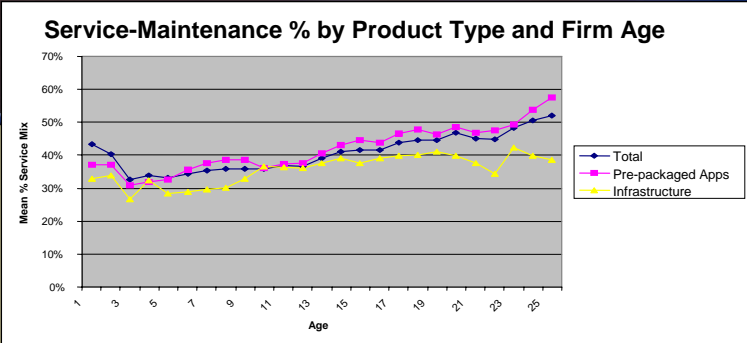
Products = 100% products Service = 0% product revenues
 Hybrid-Product = ca. 80-99% product revenues
 Hybrid-Balanced = ca. 36-79% product revenues
 Hybrid-Service = ca. 0-35% product revenues

Percentage Breakout by Year





33



34

Why Shift to Services-Maintenance?

- Associated with negative growth in sales overall & product sales
- Associated with higher firm age (life-cycle effect?)
- Associated with platform shift (Internet boom, bust?)

Random effects regression on product firms (> 50% product revenue) that shifted to > 50% services-maintenance
Dependent variable = percentage of sales from services-maintenance (continuous)

Explanatory variables = age, client/server shift (1990-1994), internet shift (1998-2003), 2-year avg. growth rate

Controls: size and market

Results:

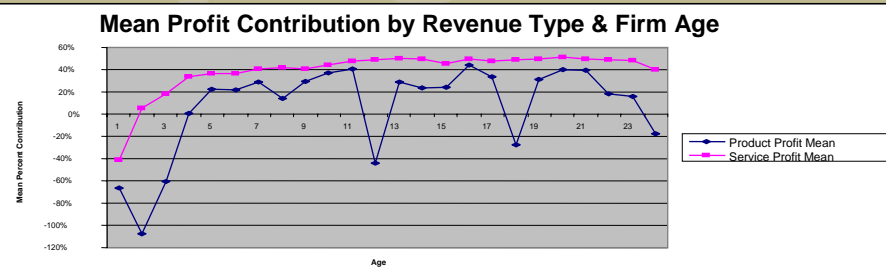
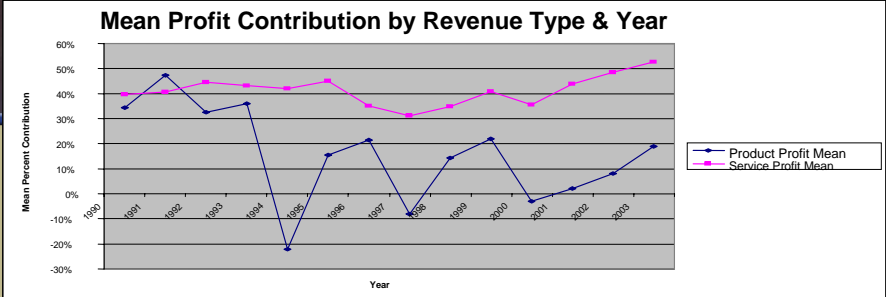
	<u>Coefficient</u>
Age	.015*
	(.001)
C/S	-.02
	(.013)
Internet	.10*
	(.011)
Ln sales	-.016*
	(.006)
2 year Avg. Prod Growth	-.006*
	(.001)
Market	selected significant
R2	.22
N	857

35

Revenue Mix and Performance?

- **Service-maintenance revenues** generate higher and more stable profits than product revenues **for all firms**
- **Hybrid firms** generally have higher and more stable profits than products or services firms

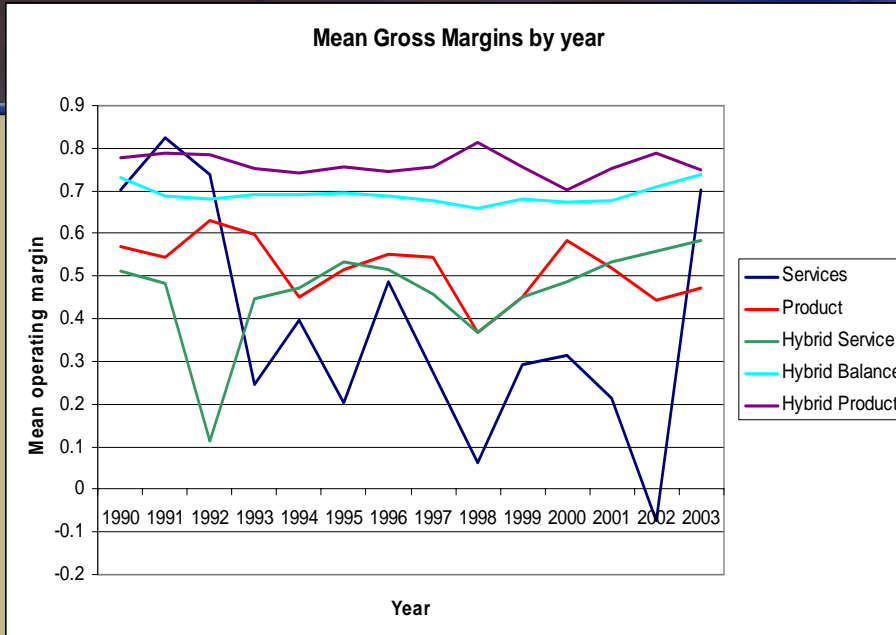
36



Product profitability = (product sales - (product cost + R&D)) / product sales

Service profitability = (service & maintenance revenue - service & maintenance cost) / service & maintenance revenue

37



Maintenance Contribution?

- Sample: 598 data points of firms per year that broke out maintenance from other service revenues (*i.e. probably a biased sample favoring firms with large maintenance %*)
- Mean of 61% maintenance as % of total service revenues
- Adj. mean of 55% if eliminate 75 data points of firms per year reporting 100% maintenance

Ran random effects regression using all observations with maint. broken out

Dependent variable: service margins

Explanatory Variable: maintenance as % of services

Control Variables: age, size, market, year

Maintenance comes out significant with a coefficient of .53.

Interpretation: A 10% increase in maintenance as a % of service revenues = 5.3% increase in service margins

39