ICEAA

ISBSG

IT Confidence 2016

4°International Conference on
IT Data collection, Analysis and Benchmarking
Los Angeles, CA (USA) – September 7, 2016

Software Data Collection

A Historical Perspective

Dr. Randall Jensen

Software Acquisition Consultant

Software Data Collection

Goals of the presentation

- ✓**G1.** Expand the definition of software data
- ✓**G2.** Extend the scope of relative software data
  - ✓Technology
  - ✓Management and communications
  - ✓Process
- ✓**G3.** Project needs of next generation

Data Collection Prehistory

- The concern of data collection began as a concern over the role of management in industrial development.
- The next major concern was the role of people in industrial productivity – assembling and wiring.
- Wiring and assembling evolved into coding as computers evolved from programming with wires into producing software in the late 1940s.
- What happened to the concern over people in productivity?

Management Research Highlights

- General management concepts, 1911

  Management philosophy: Plan, Organize, Command, Coordinate, Control
- Hawthorne experiment (1924-1932)
- People impacts, Mayo, 1933
- Lockheed Skunk Works, 1940s
- Theory X/ Theory Y, McGregor, 1960
- CMM, Humphrey, 1989

Lockheed Skunk Works™

- Unofficial name given to Lockheed Advanced Development Projects Unit managed by Kelly Johnson – dates back to the period around WWII.
- Dispenses with both physical and nonphysical walls
- Known for producing the P-80 fighter aircraft prototype (XP-80) in only 143 days.
- Johnson: "*We are defined, not by the technologies we create, but by the process in which we create them"*

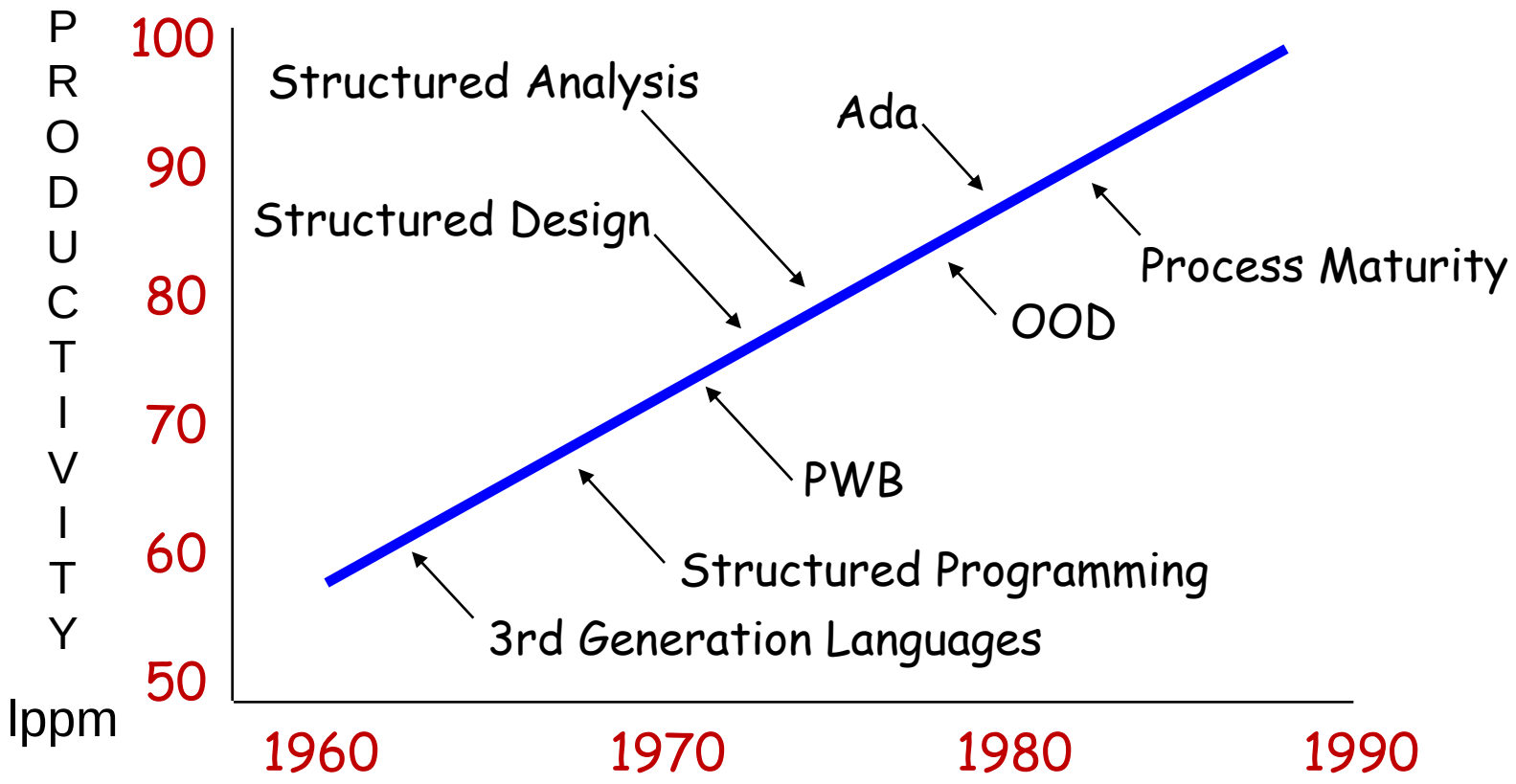
Early Software Data Collection

- Serious software data collection started in about 1970
  - Development cost
  - Delivery schedule
  - Lines of code
- Ray Wolverton's *IEEE Transactions on Computers* paper "The Cost of Developing Large Scale Software," June 1974
- Software life cycle definitions introduced

Process Evolution

- Code and Repair
- Waterfall 2167A (1960s)
- Proto typing
- Incremental
- Spiral Development (1970s)
- Rapid Application Development (RAD)
- Capability Maturity Model (CMM) (1990s)
- CMM Integrated (CMMI)
- Agile

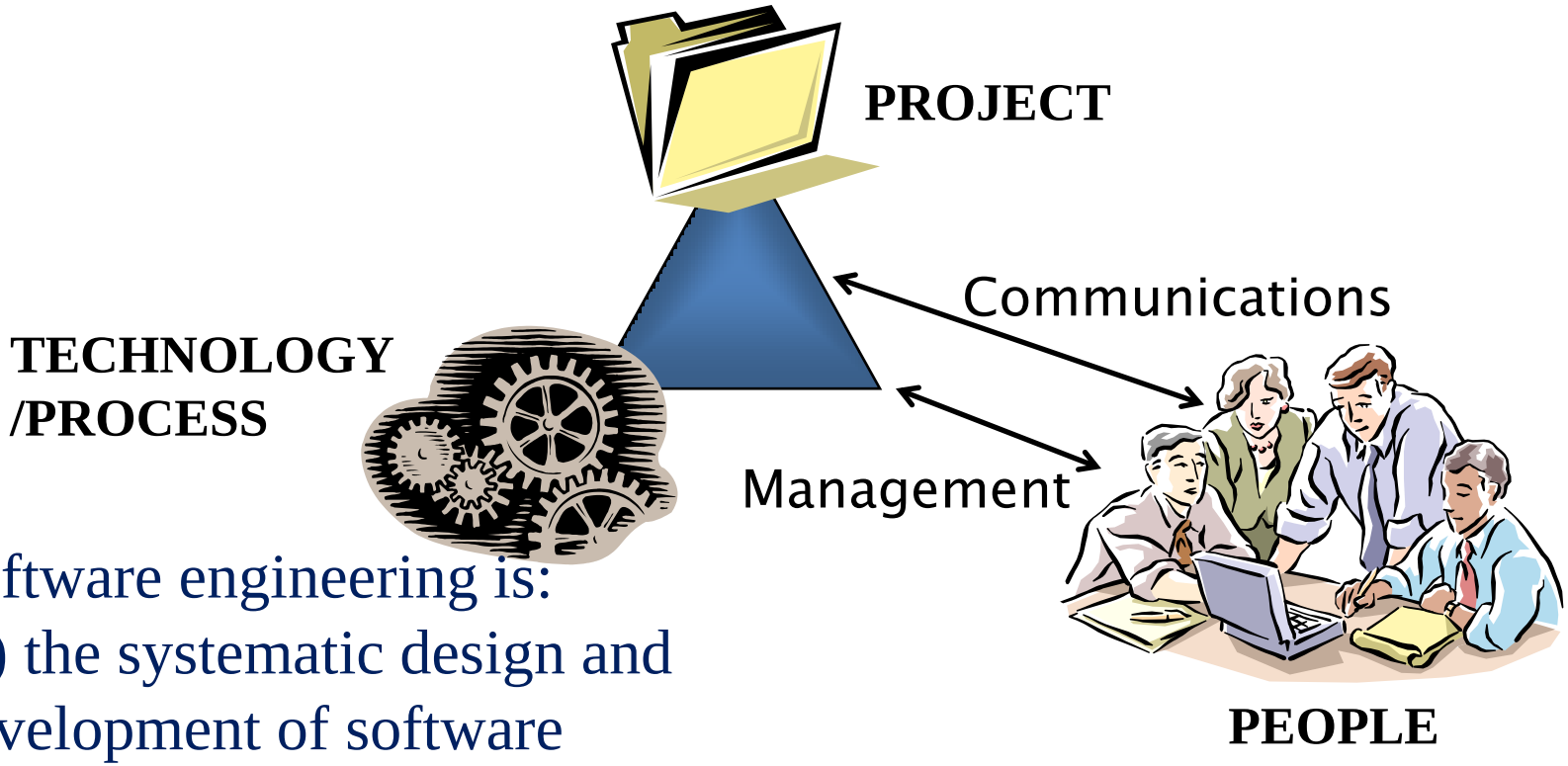
Software Productivity Gains



1975 Pair Programming Pilot Study

- Real time system executive
- 30,000 Fortran source lines
- 6 software components
- 5 two-person teams plus manager
- Average productivity prior to study = 77 LPPM
- Pilot study productivity = 175 LPPM
- Error rate <0.1% of norm

3 Dimensions of Management



Software engineering is:
(1) the systematic design and development of software products (2) the management of the software process.

Effectiveness Formula

$$E = C[M(CS)]$$

where

***E* = Effectiveness (0 – 1)**

C = Communication skills (0 – 1)

M = Management concept awareness (0 – 1)

CS = Computer science technical ability (0 – 1)

Jensen and Tonies, *Software Engineering*, PH, 1979

2015 Average Effectiveness = 0.25

Estimating model development

- Software development project data collection dominated the 1970 to 1990 period
  - Size (mostly SLOC), delivery schedule, and delivered product cost (person months)
- Estimating models introduced in 1980 and later
- Software project data now include model parameter information
  - Developer, project and environment constraints
  - Capability ratings always greater than 75th percentile
- The estimates today are as good as the estimates made in the early 1980s

Component-level estimating models

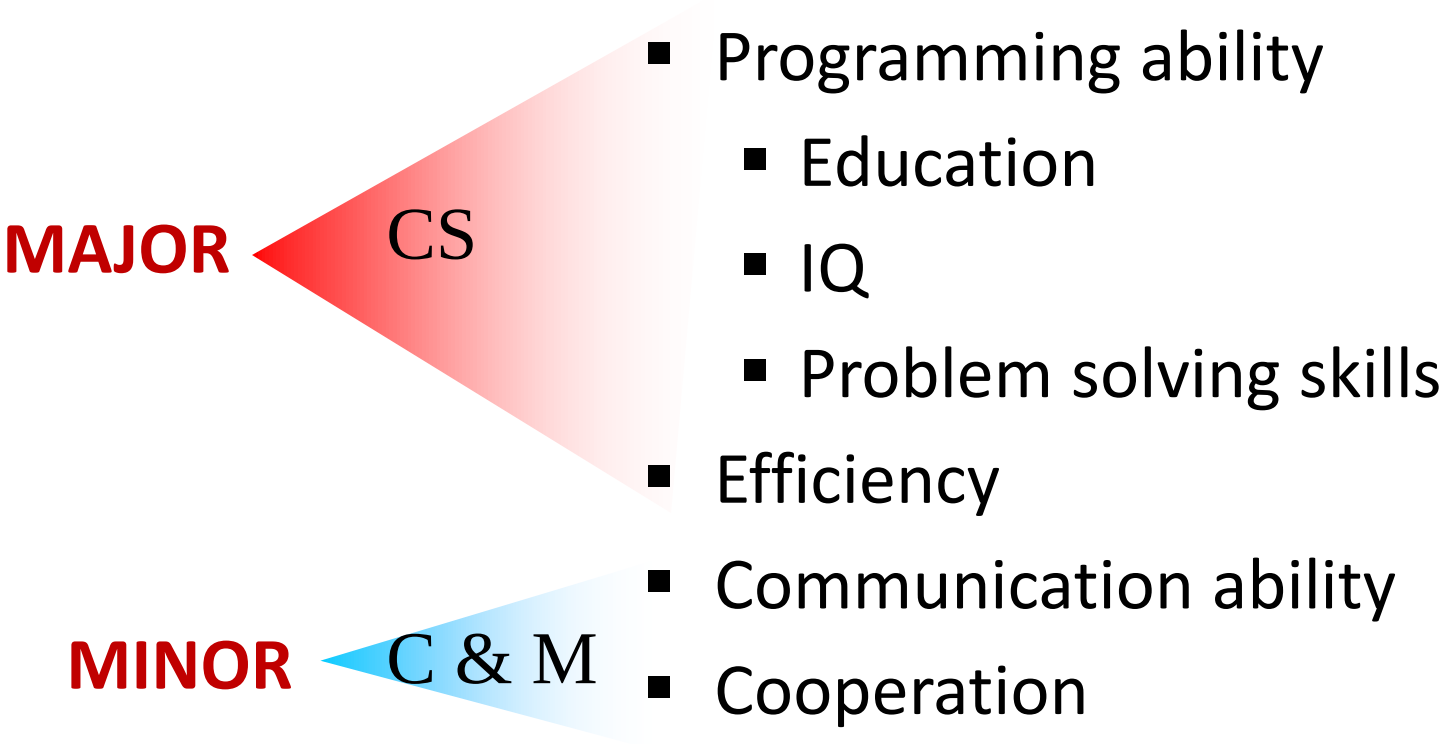
Model	Release	Released by
Price-S	1980	Price Systems
Seer	1979	SEI
SEER-SEM	1990	Galorath Inc.
Sage	1995	SEI
COCOMO	1981	Wolverton / Boehm
COCOMO II	2000	USC
REVIC	1985	R. Kile
SLIM	1976	L. Putnam

ACAP - Analyst Capability

COCOMO Definition

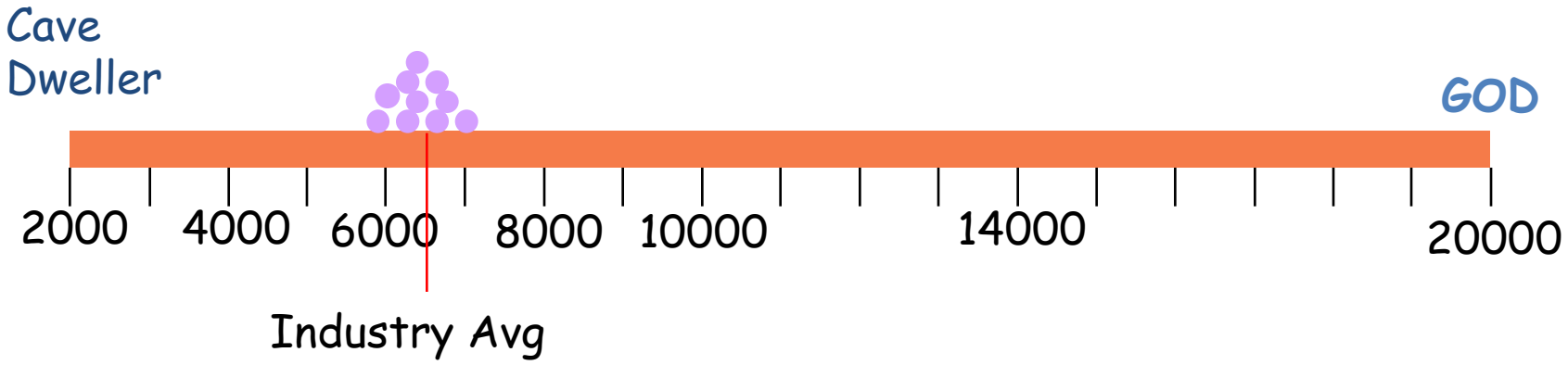
Value	
0.71	90th percentile of industry population
0.86	75th percentile of industry population
1.00	50th percentile of industry population
1.19	35th percentile of industry population
1.46	15th percentile of industry population

COCOMO Capability Rating



Traditional data does not work for agile development environments

The Seer Ctb Calibration 1980



- 1980 Seer Basic Technology Constant distribution
- Distribution unchanged over time (1980-2010) for classic projects
- Explains why estimating tools are unchanged over time
- Management style was the differentiator

Fundamental effort equation

$$E = CS_e^{\alpha}$$

Where: E = Development effort

C = Environment calibration constant

S_e = Lines of Source Code (includes reuse effects)

α = Entropy constant

Accounting for entropy

- Estimating model adjustment that allows for productivity change with size

$$Effort = productivity \times size^{Penalty}$$

- Typical penalty values

Tool	Penalty
COCOMO (embedded)	1.2
COCOMO II (embedded)*	~1.15
Price-S	~1.2
REVIC (embedded)	1.2
Sage	1.2
SEER-SEM	1.2
SLIM	1.263

** Penalty is adjustable to fit project. Table uses typical for embedded systems*

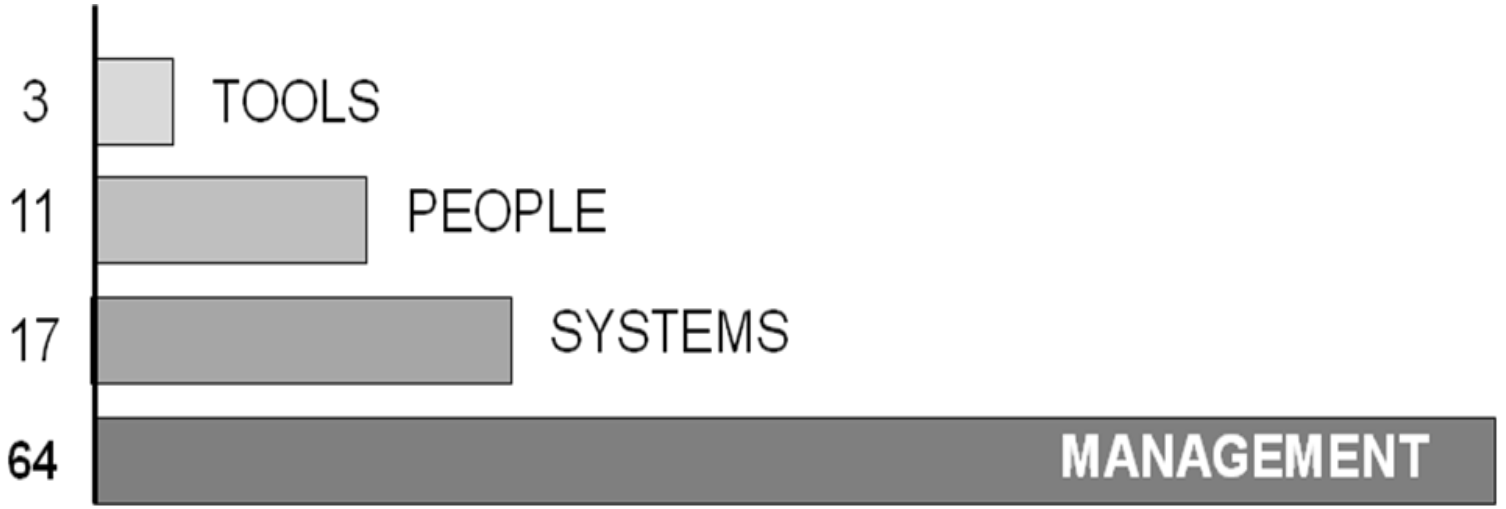
Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions over processes and tools.*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation.*
- *Responding to change over following a plan.*

That is, while we value the items on the right, we value the items on the left more.

Relative Impact of Project Elements



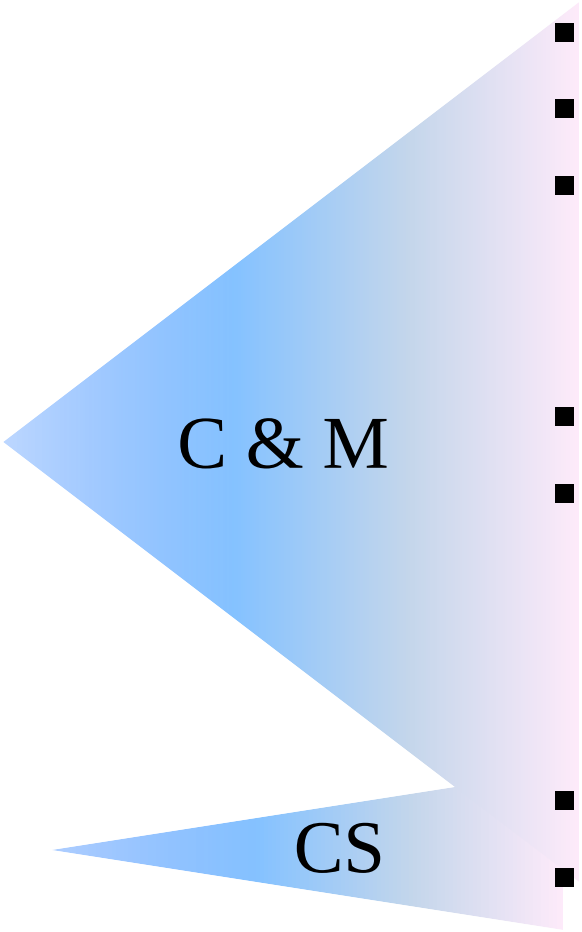
Source: G. Weinberg, Quality Software Management, Vol. 3

ACAP - Analyst Capability

Sage Definition

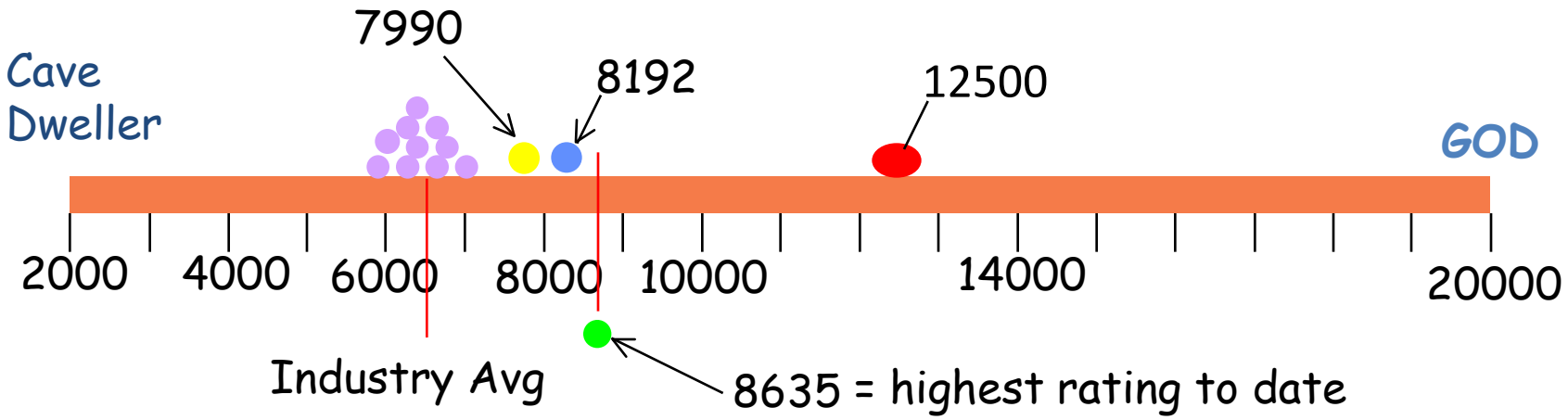
Value	
0.71	Highly motivated AND Experienced team organization
0.86	Highly motivated OR Experienced team organization
1.00	Traditional software development organization
1.19	Poorly motivated OR non-associative organization
1.46	Poorly motivated AND non-associative organization

Effective Capability Rating



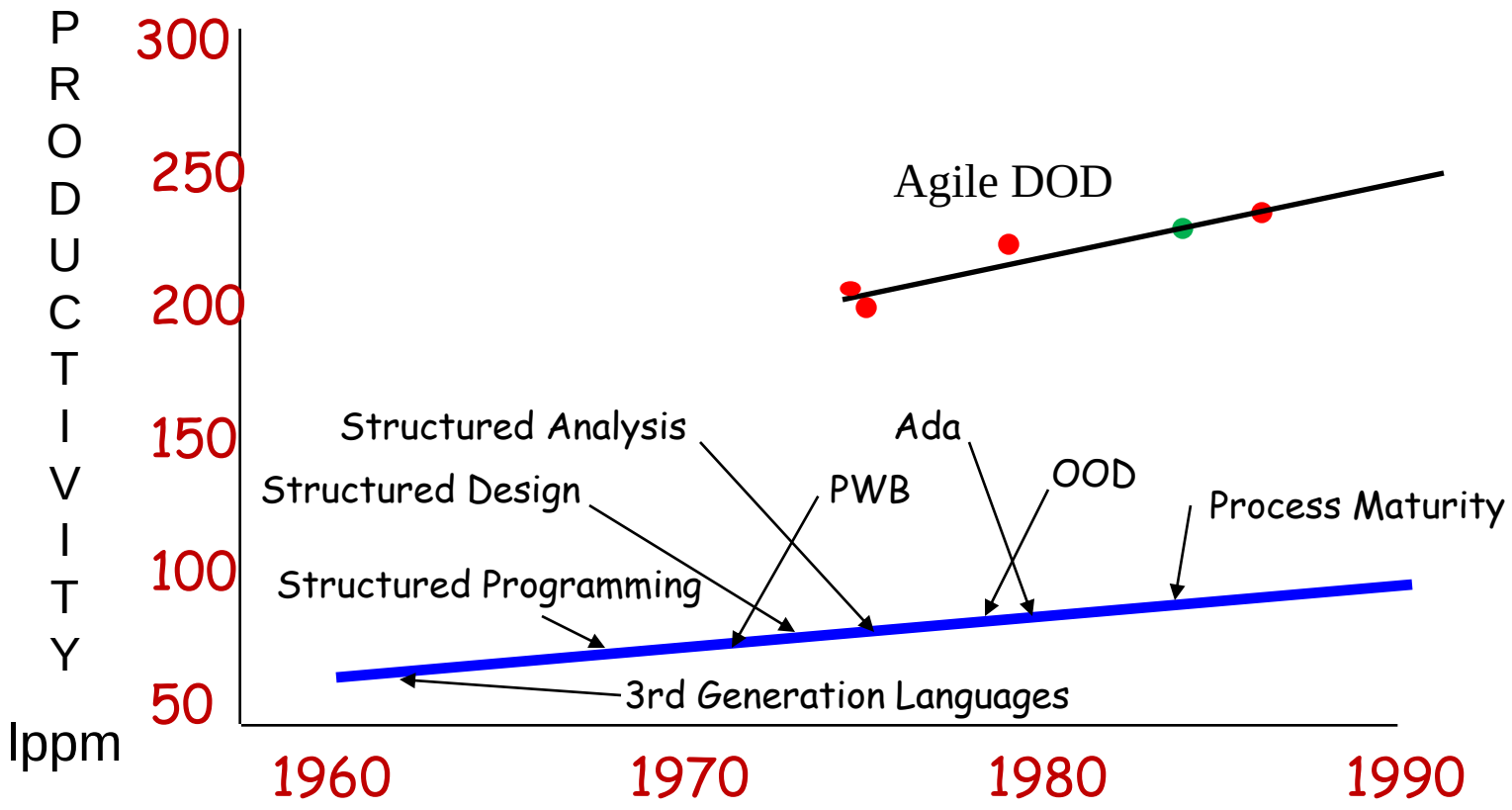
- Motivation
- Management style
- Use of team methods
  - Communication ability
  - Cooperation
- Hawthorne Effect
- Working environment
  - Noise level
  - Individual working (thinking) space
  - Proximity of team members
- Problem solving skills
- Programming ability

The Ctb Calibration Surprise



- Same application, same organization, same facilities, same people
- Two projects far outside grouping
- Cost models could not explain outriders
- Management style was the differentiator

Software Productivity Gains



History Contributions

- Low productivity improvement over the last 3 decades has been primarily driven by technology
- Traditional development models work well with traditional development environments.
- Productivity/Effectiveness is a function of 3 attributes: communications, management, and technology
- Communications and management are **KEY** productivity drivers
- Effectiveness Formula applies to almost all development environments

Software Capability Estimating Model

Capability Rating

Development Component:	*Task Name*
Complexity, D	12
Effective Size	
SLOC, New	80,000
SLOC, Modified	2,000
SLOC, Reused	33,000
Size, Total	115,000
Size, Effective	82,650
Ctb Calculation	
ACAP	1
AEXP	1.29
MODP	1.1
PCAP	1
RESP	1
TOOL	1
TURN	0.87
Capability Score	0.19
Percentile Rating	29%
Ctb	4,953
Templates	
Management	3
Product	13

Cte Calculation	
Experience	
DEXP	1.00
LEXP	1.00
PEXP	1.00
TEXP	1.00
Support	
PVOL	1.00
RLOC	1.00
Environment	
DVOL	1.07
RDED	1.00
Management	
MCLS	1.00
MORG	1.00
MULT	1.00
Product	
DISP	1.05
HOST	1.00
MEMC	1.00
QUAL	1.03
RTIM	1.00
RVOL	1.07
SECR	1.00
SPEC	1.12
TEST	1.05
TIMC	1.00
TVOL	1.00
Cte	3,402

Basic Estimate		
Development Effort	586.5	PM
Productivity	140.9	SLOC/PM
Requirements Level	8.0	Percent
Requirements Effort	46.9	PM
Integration Level	22.0	Percent
Integration Effort	129.0	PM
Total Effort	762.5	PM
Development Schedule	26.2	MO
Requirements Schedule	7.4	MO
Peak development staff	34.5	Pers
Peak programming staff	18.5	Pers
Max staffing rate	26.2	Pers/Yr

Maintenance	
Upgrades (ACT)	10 %/Yr
Upgrade Effort	87.2 PM/Yr
Knowledge Retention	88.2 PM/Yr
Support level	88.2 Pers/ Yr

Effectiveness Value Rating

Industry Average

Effectiness Value	Percentile of Industry	Ctb Value
0.10	5.0	
0.17	22.4	4570
0.20	35.2	
0.23	50.0	5707
0.28	57.9	6735
0.30	61.0	7000
0.31	62.5	7350
0.40	75.2	8630
0.45	81.0	9587
0.50	86.0	
1.00	99.9	

The Next Generation?

- The next generation of software development is already here
  - First evidence of an organization shift is traceable to the mid 1940s (Lockheed Skunk Works®)
- Symptoms of next generation shift
  - Effectiveness Formula (1979)
  - Agile Manifesto (2001)
  - Herman Miller Co. introduces new office evolution including movable walls and open workplace forms (2013)