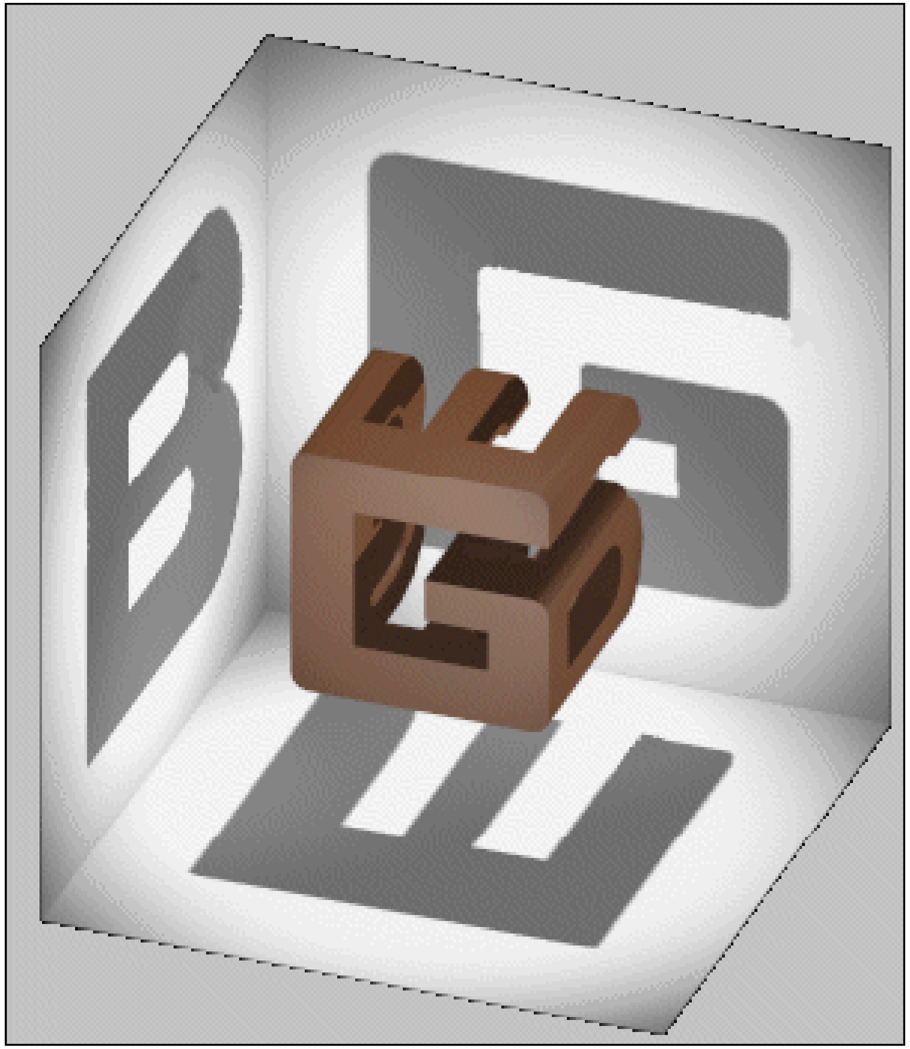
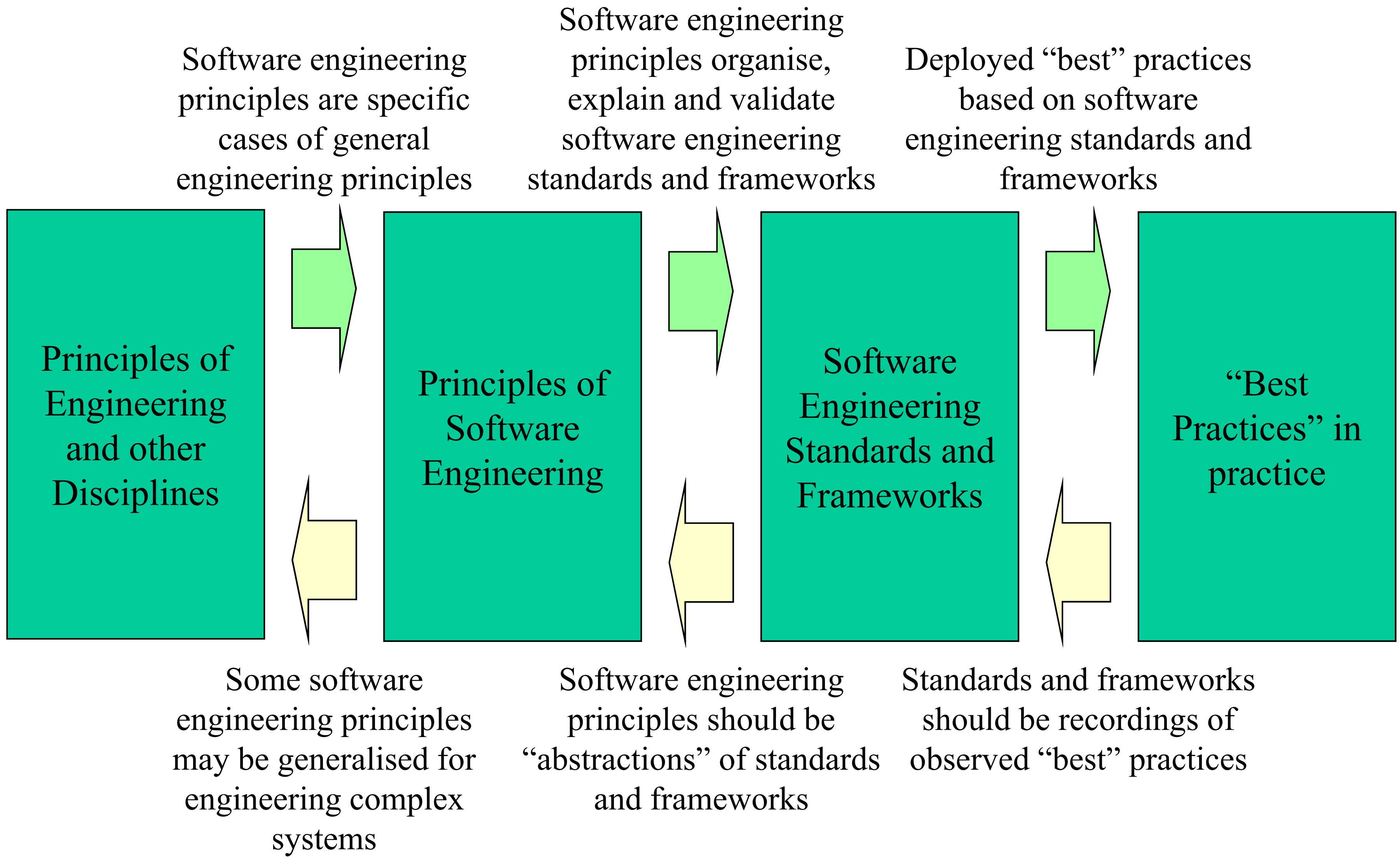
What’s all this
“Creative Commons” about?

Duncan Hall
SPDO Software and Computing
2009 Oct 6

- Some background – who am I?
- What is this Creative Commons <u>not</u> about?
- What are (some of) the challenges facing SKA?
- How will this workshop change the world?

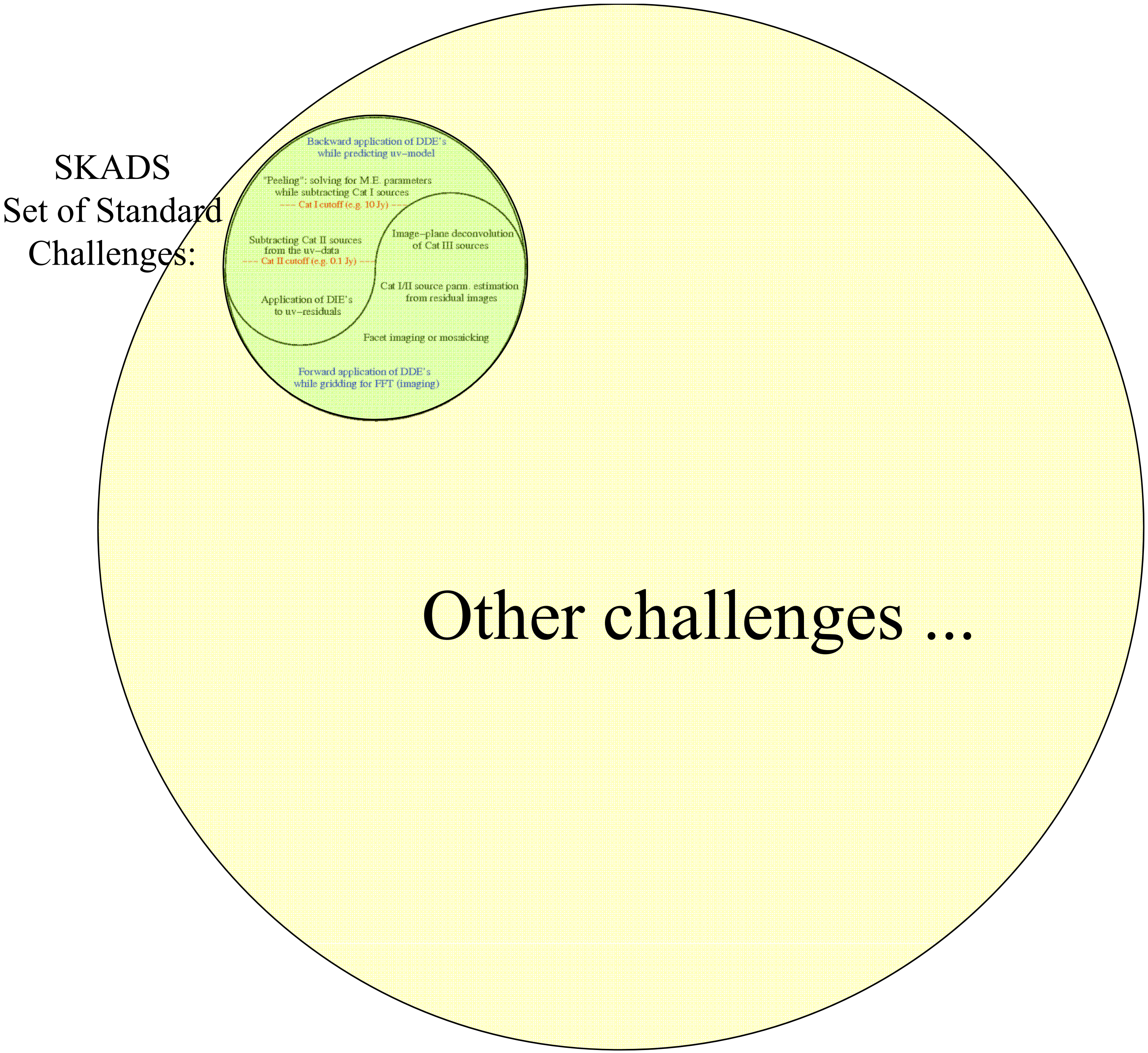
Multiple viewpoints:





Source: Jabir and Moore 1998, “A search for fundamental principles of software engineering”

- Some background – who am I?
- What is this Creative Commons <u>not</u> about?
- What are (some of) the challenges facing SKA?
- How will this workshop change the world?


SKADS
Set of Standard
Challenges:
Backward application of DDE's
while predicting uv–model
"Peeling": solving for M.E. parameters
while subtracting Cat I sources
--- Cat I cutoff (e.g. 10 Jy) ---
Subtracting Cat II sources
from the uv–data
--- Cat II cutoff (e.g. 0.1 Jy) ---
Image–plane deconvolution
of Cat III sources
Cat I/II source parm. estimation
from residual images
Application of DIE's
to uv–residuals
Facet imaging or mosaicking
Forward application of DDE's
while gridding for FFT (imaging)
Other challenges ...

CALIM08: Software Development Survey

- Was there a formal software process?
- Were architectural definition documents used?
- How did the change control board function?
- What was the review process?
- What were the team dynamics?
- How best to communicate across the team?
- Time and cost against estimated budget?
- What could have been done better?
- Suggestions for SKA software development?

Sound familiar?

- Over-commitment
  - Frequent difficulty in making commitments that staff can meet with an orderly engineering process
- Often resulting in a series of crises
  - During crises projects typically abandon planned procedures and revert to coding and testing
- In spite of ad hoc or chaotic processes, can develop products that work
  - However typically cost and time budgets are not met
- Success depends on individual competencies and/or “death march” heroics
  - Can’t be repeated unless the same individuals work on the next project
- Capability is a characteristic of individuals, not the organisation

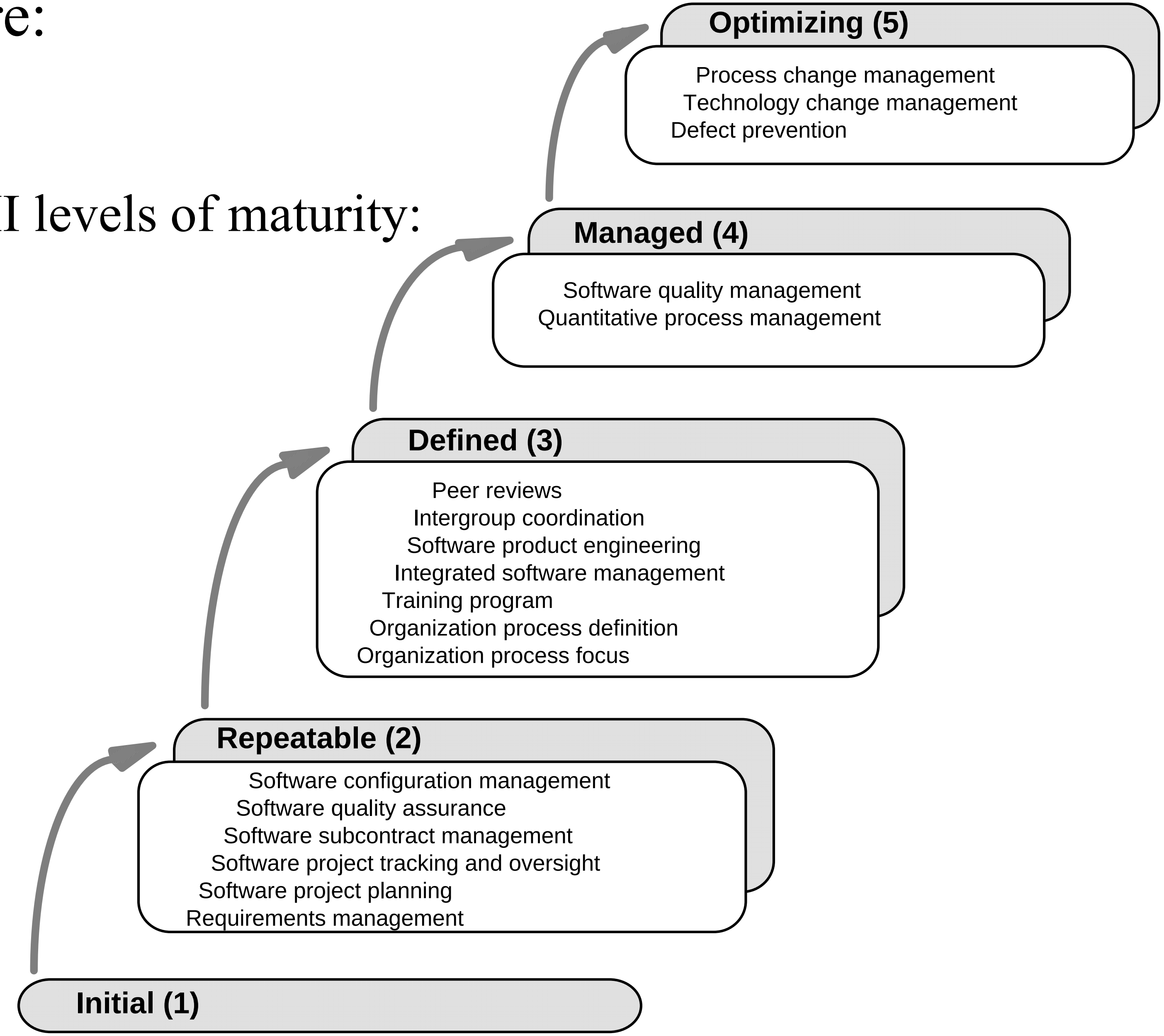
Mark Paulk et alia: The Capability Maturity Model for Software; in “Software Engineering” M. Dorfman and R. H. Thayer, 1997

What are the SEI CMMI process “Maturity Levels”?

Level	Process Characteristics
1	Process is informal and ad hoc
2	Project management and project oversight practices are institutionalised
3	Organisational processes, including technical and project management, are clearly defined and repeatable
4	Processes are stabilised and aligned to goals, and product and process are quantitatively controlled
5	Process improvement is consistently and rigorously practised at organisation and project levels

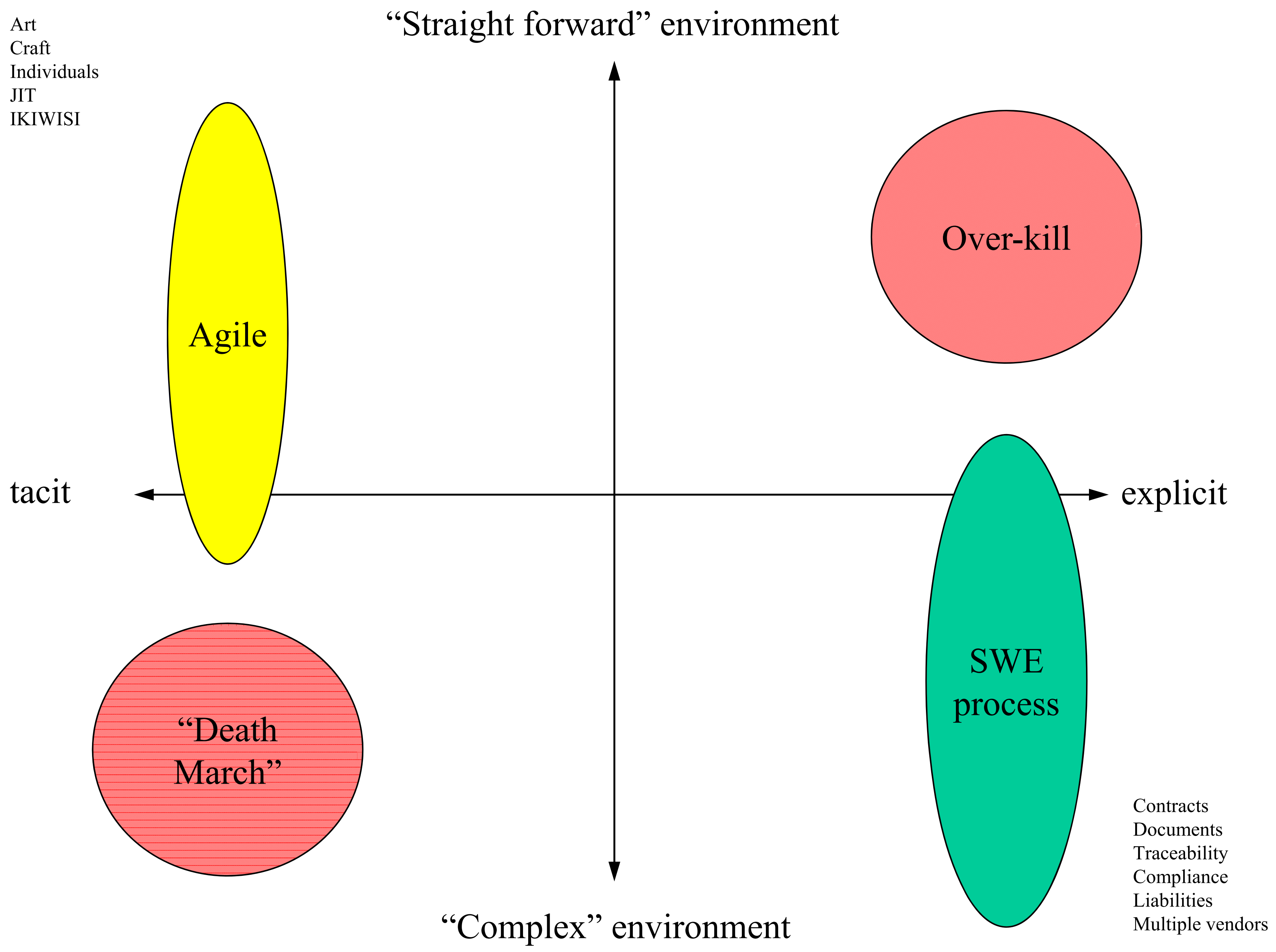
A "Leap of Faith": Reliable processes deliver reliable software:

CMMI levels of maturity:



Why work to build process reliability?

- All those practising as software engineers should desire to evolve out of the chaotic activities and heroic efforts of a Level 1 organisation
  - *Because no one likes a 'painful' work environment*
- Good software can be developed by a Level 1 organisation, but often at the expense of the developers
  - *People get tired of being the hero*
- At the repeatable level, Level 2, software engineering processes are under basic management control and there is a management discipline
  - *Even the most die-hard techie needs time away from work*



Art
Craft
Individuals
JIT
IKIWISI
"Straight forward" environment
Agile
Over-kill
tacit
explicit
SWE
process
"Death
March"
"Complex" environment
Contracts
Documents
Traceability
Compliance
Liabilities
Multiple vendors

Where are we (the SPDO) at?

Current development

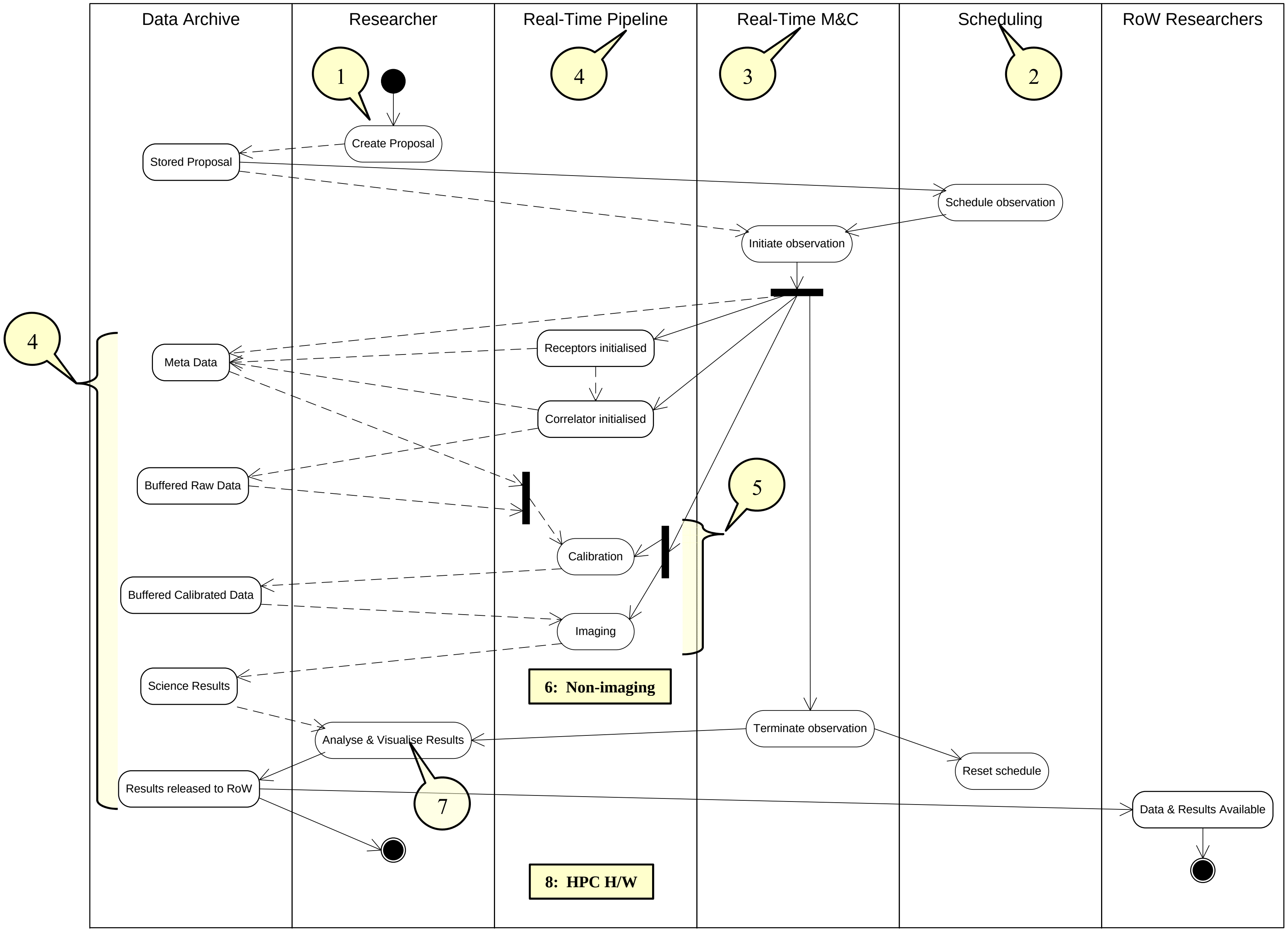
- PrepSKA 2008-2011
- The Preparatory Phase for the SKA is being funded by the European Commission's 7th Framework Program
- €5.5M EC funding for 3 years + €17M contributed funding from partners (still growing)
- €150M SKA-related R&D around the world
- Coordinated by the Science and Technology Facilities Council (UK)

WP2: Design + Cost

Coordinated by the SKA Program Development Office in Manchester

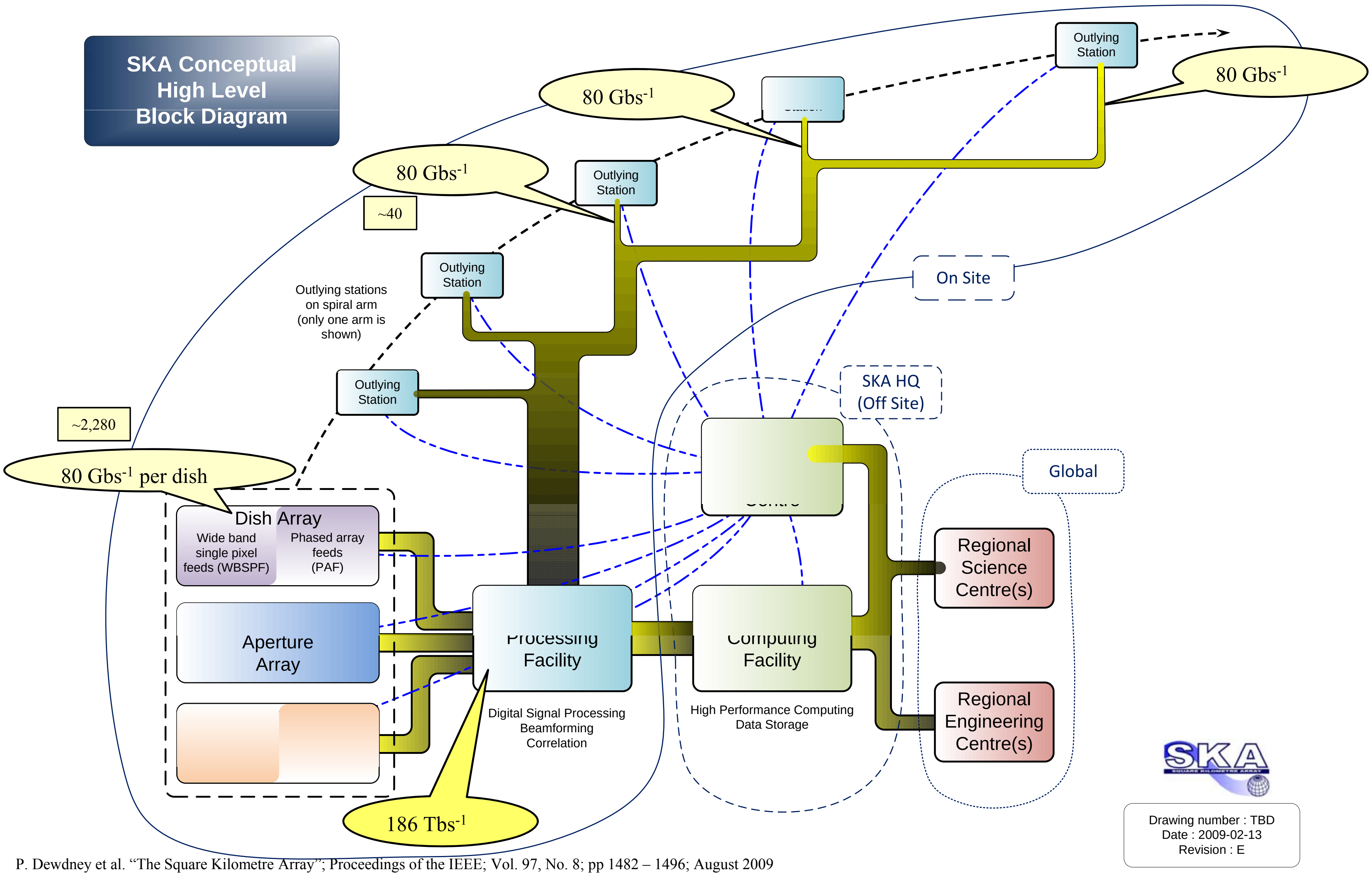
- System Definition
- Dishes, feeds, receivers
- Aperture arrays
- Signal transport
- Signal processing
- Software
- High performance computers
- Data storage
- Power requirements

A structure for SKA software and computing:



Estimating the sizes
of the
computing challenges

Φ2 real-time data from dishes



P. Dewdney et al. “The Square Kilometre Array”; Proceedings of the IEEE; Vol. 97, No. 8; pp 1482 – 1496; August 2009

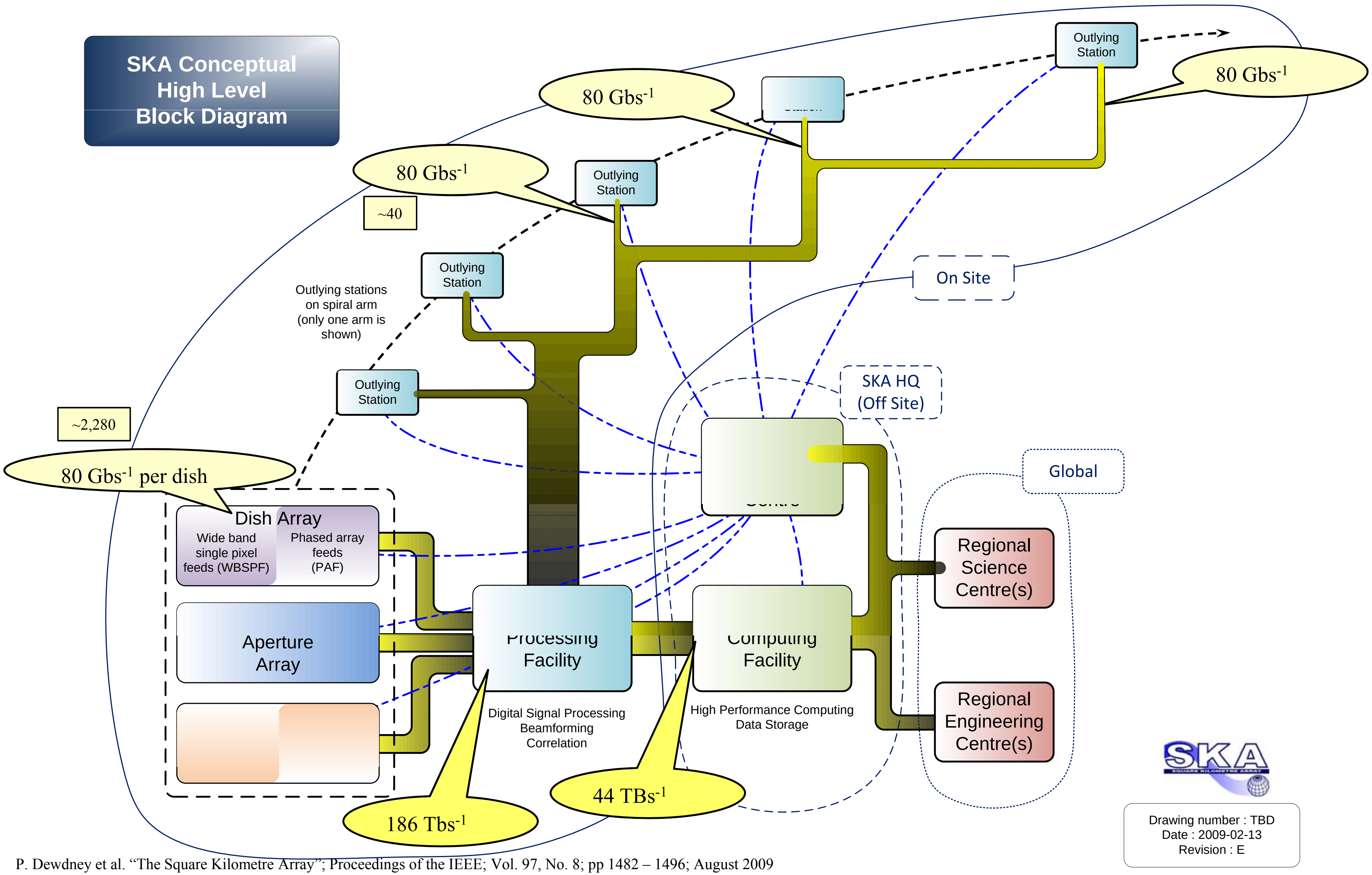
J. Cordes “The Square Kilometre Array – Astro2010 RFI #2 Ground Response” 27 July 2009; Table 1, pp 9 - 10

SKA Baseline system correlator output:

Digital Outputs			
Core-to-Mid antennas			
Sample streams*		4	Sampled 2 GHz sub-bands
bits per sample		4	
Outer Antennas (stations)			
Sample streams*		4	One summed beam, formed digitally per station
bits per sample		4	
Signal Transport System			Optical fiber to central data processor
Radius < 200 km		80 Gbit s^{-1}	Data rate per antenna*
			Optimized layout, buried fiber, highly multiplexed.
Radius > 200 km		80 Gbit s^{-1}	Data rate per station*
			Leased and locally buried fiber connections.
Central Data Processing System			
Correlator			
Input data streams		13,920	2280 ants x 6 streams + 40 stns x 6 streams
Frequency channels		10^5	Distributed as necessary over streams
Complex Correlations		1.1×10^{12}	$(2320^2/2)$baselines x 4 pol'n prod's x 10^5 chans
Dump Time		200 ms	Average dump time
Beamformer			
Bandwidth		2 GHz	Equivalent to one 2 GHz sub-band.
Core-Inner beamformer		10 beamformers	1500 inputs, dual polarization, steerable on sky
Time-domain Processor			
Pulsar / Transient Detector		20	One per output of beamformer system.

J. Cordes "The Square Kilometre Array – Astro2010 RFI #2 Ground Response" 27 July 2009; Table 1, pp 9 - 10

SKA Baseline system correlator output:



P. Dewdney et al. "The Square Kilometre Array"; Proceedings of the IEEE; Vol. 97, No. 8; pp 1482 – 1496; August 2009
J. Cordes "The Square Kilometre Array – Astro2010 RFI #2 Ground Response" 27 July 2009; Table 1, pp 9 - 10

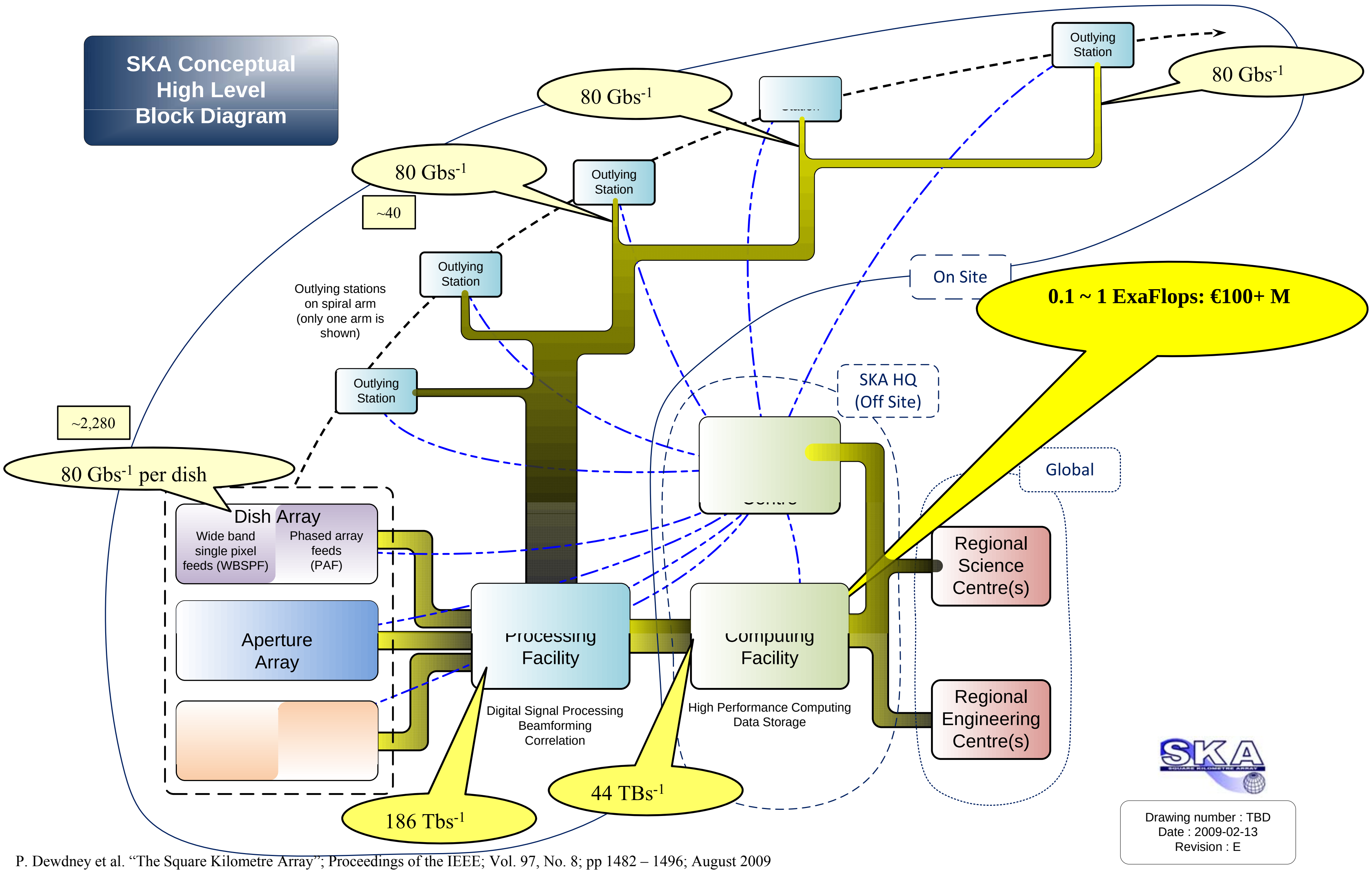
Current algorithm performance:

Peak data rate	25 MB/s
Data for Peak 8-hr observation	700GB
flops per float	100 - 10000
Peak compute rate	5Tflop
Average/Peak computing load	0.1
Average compute rate	0.5Tflop
Turnaround for 8-hr peak observation	40 minutes
Average/Peak data volume	0.1
Data for Average 8-hr observation	70GB
Data for Average 1-yr	80TB

Table I: Typical and peak data and computing rates for the EVLA

T. Cornwell "EVLA Memo 24: Computing for EVLA Calibration and Imaging", 2001 January12

Required computation performance:



P. Dewdney et al. "The Square Kilometre Array"; Proceedings of the IEEE; Vol. 97, No. 8; pp 1482 – 1496; August 2009
J. Cordes "The Square Kilometre Array – Astro2010 RFI #2 Ground Response" 27 July 2009; Table 1, pp 9 - 10

How much data will we need to store?

Assume disk buffer for 8 hours:

Peak data rate	25 MB/s
Data for Peak 8-hr observation	700GB
flops per float	100 - 10000
Peak compute rate	5Tflop
Average/Peak computing load	0.1
Average compute rate	0.5Tflop
Turnaround for 8-hr peak observation	40 minutes
Average/Peak data volume	0.1
Data for Average 8-hr observation	70GB
Data for Average 1-yr	80TB

Table I: Typical and peak data and computing rates for the EVLA

T. Cornwell "EVLA Memo 24: Computing for EVLA Calibration and Imaging", 2001 January12

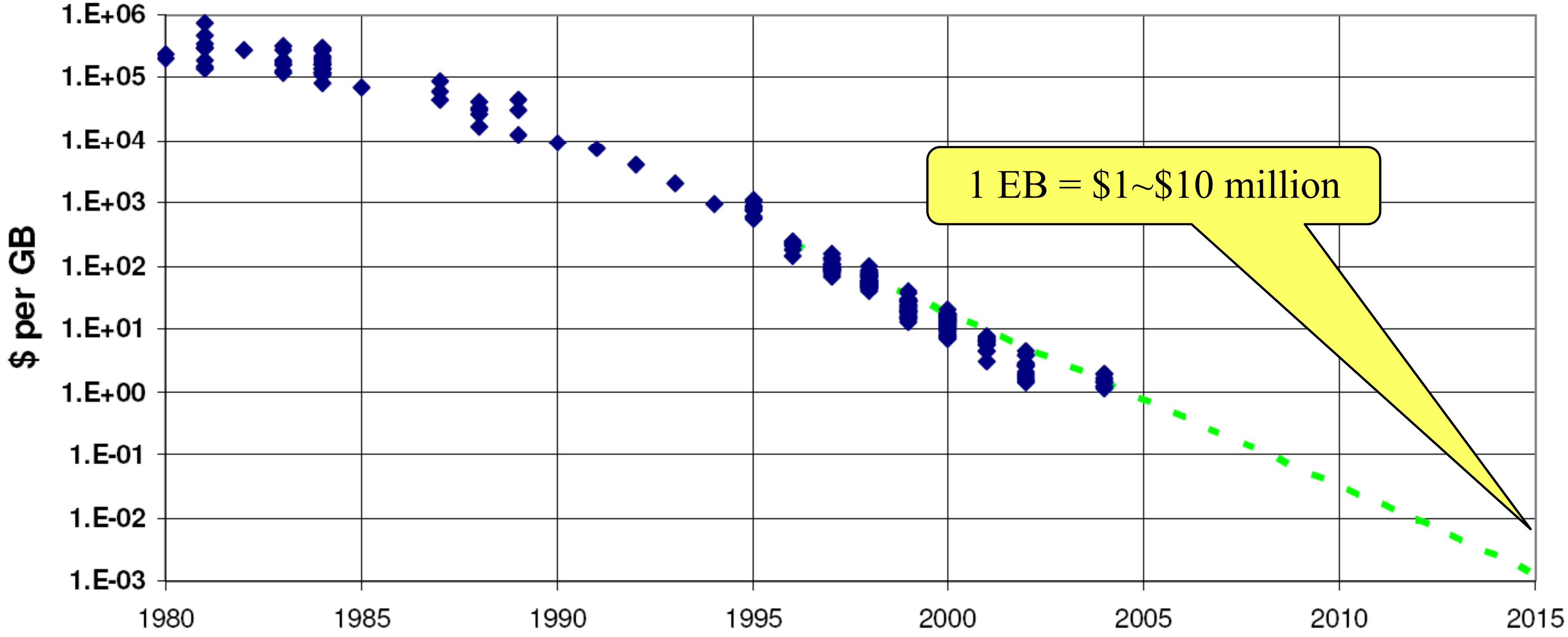
Disk buffer required:

$$44 \times 10^{12} \times 8 \times 60 \times 60$$

$$\approx 1 \times 10^{18}$$

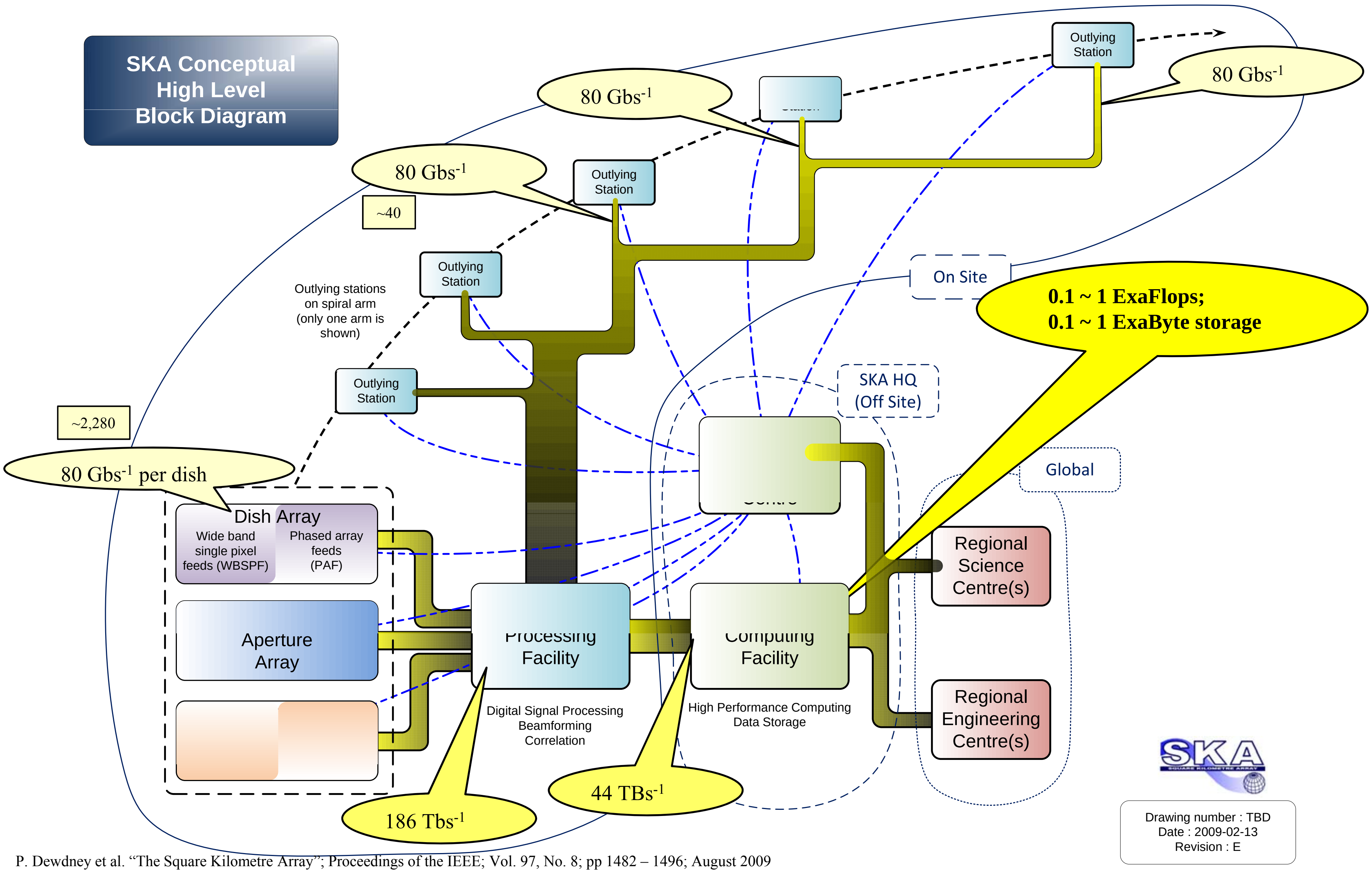
i.e. 1 ExaByte

Disk storage: annual 50% cost reduction



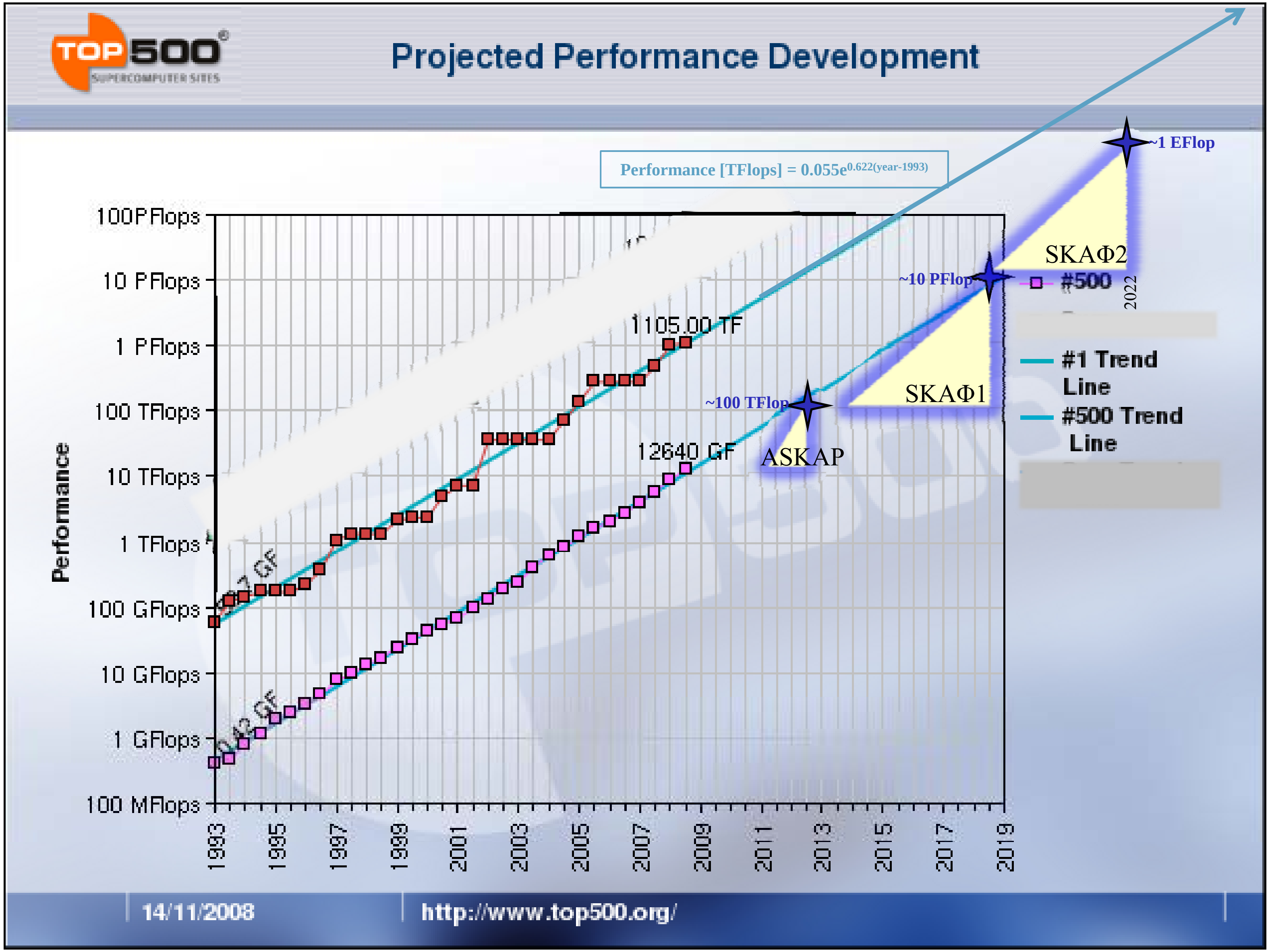
P. Kogge et alia "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems"; TR-2008-13, DARPA ExaScale Computing Study, 2008 Sep 28, page 125
Note: neither RAID, controllers, nor interconnect cables are included in these estimates

Computing and buffer requirements:



P. Dewdney et al. "The Square Kilometre Array"; Proceedings of the IEEE; Vol. 97, No. 8; pp 1482 – 1496; August 2009
J. Cordes "The Square Kilometre Array – Astro2010 RFI #2 Ground Response" 27 July 2009; Table 1, pp 9 - 10

Pushing the Flops envelope:



Cornwell and van Diepen "Scaling Mount Exaflop: from the pathfinders to the Square Kilometre Array" http://www.atnf.csiro.au/people/Tim.Cornwell/MountExaflop.pdf

Estimating the cost of power consumed in data centres

HP Labs cost model:

$$Cost_{total} = Cost_{space} + Cost_{IT\ hardware\ power} + Cost_{cooling} + Cost_{operation}$$

"Cost Model for Planning, Development and Operation of a Data Center"; C. D. Patel, A. J. Shah; Hewlett Packard Laboratories
Technical Report HPL-2005-107(R.1); 9 June 2005

Direct costs of power consumed:

$$Cost_{total} = Cost_{space} + Cost_{IT\ hardware\ power} + Cost_{cooling} + Cost_{operation}$$

$$Cost_{IT\ hardware\ power} = U_{\$,grid}\, P_{IT\ hardware} + K_1 U_{\$,grid}\, P_{IT\ hardware}$$

Where:

$K_1 = J_1 U_{\$,A\&M\ power} / (U_{\$,grid})$ [$K_1 \equiv$ Utility burdening factor, sometimes taken as ≈ 2]

J_1 = Installed maximum capacity [Watts] / (Utilised capacity [Watts])

$= P_{rated} / P_{IT\ hardware}$ [$J_1 \equiv$ Utilisation factor; typically $\approx 1.33 - 1.6$ for growth]

$$Cost_{IT\ hardware\ power} = U_{\$,grid}\, P_{IT\ hardware} + U_{\$,A\&M\ power}\, P_{rated}$$

Typically:

$$U_{\$,A\&M\ power} \approx U_{\$,grid}$$

$$Cost_{power} = U_{\$,grid}\, P_{IT\ hardware} (1 + J_1)$$

"Cost Model for Planning, Development and Operation of a Data Center"; C. D. Patel, A. J. Shah; Hewlett Packard Laboratories Technical Report HPL-2005-107(R.1); 9 June 2005

Direct costs of power for cooling:

$Cost_{total} = Cost_{space} + Cost_{IT\ hardware\ \ power} + Cost_{cooling} + Cost_{operation}$

$Cost_{cooling} = U_{\$,grid}\ P_{cooling} + K_2 U_{\$,grid}\ P_{cooling}$

<u>Where:</u>

$K_2 = J_1 U_{\$,A\&M\ cooling} / (U_{\$,grid})$ [$K_2 \equiv$ Cooling burdening factor]

$L_1 = P_{cooling} / (P_{IT\ hardware})$ [$L_1 \equiv$ Load factor; ≈ 0.8 due to thermodynamics]

$Cost_{cooling} = U_{\$,grid}\ L_1\ P_{IT\ hardware} + U_{\$,A\&M\ cooling}\ J_1\ L_1\ P_{IT\ hardware}$

<u>Typically:</u>

$U_{\$,A\&M\ cooling} \approx 0.5\ (U_{\$,A\&M\ power})$

$Cost_{cooling} = U_{\$,grid}\ L_1\ P_{IT\ hardware}\ (1 + 0.5\ J_1)$

"Cost Model for Planning, Development and Operation of a Data Center"; C. D. Patel, A. J. Shah; Hewlett Packard Laboratories
Technical Report HPL-2005-107(R.1); 9 June 2005

“Not quite total” power cost model:

$$Cost_{\text{“NQT”}} = Cost_{space} + Cost_{IT\ hardware\ \ power} + Cost_{cooling} + Cost_{operation}$$

$$Cost_{\text{“NQT”}} = 0 +$$

$$U_{\$,grid}\, P_{IT\ hardware}\, (1 + J_1) +$$

$$U_{\$,grid}\, L_1\, P_{IT\ hardware}\, (1 + 0.5\, J_1)$$

$$+ 0$$

$$= U_{\$,grid}\, P_{IT\ hardware}\, (1.8 + 1.4\, J_1)$$

$$Cost_{\text{“NQT”}} \approx (\ \text{3.7 to 4.0 multiplied by}\ )\ U_{\$,grid}\, P_{IT\ hardware}$$

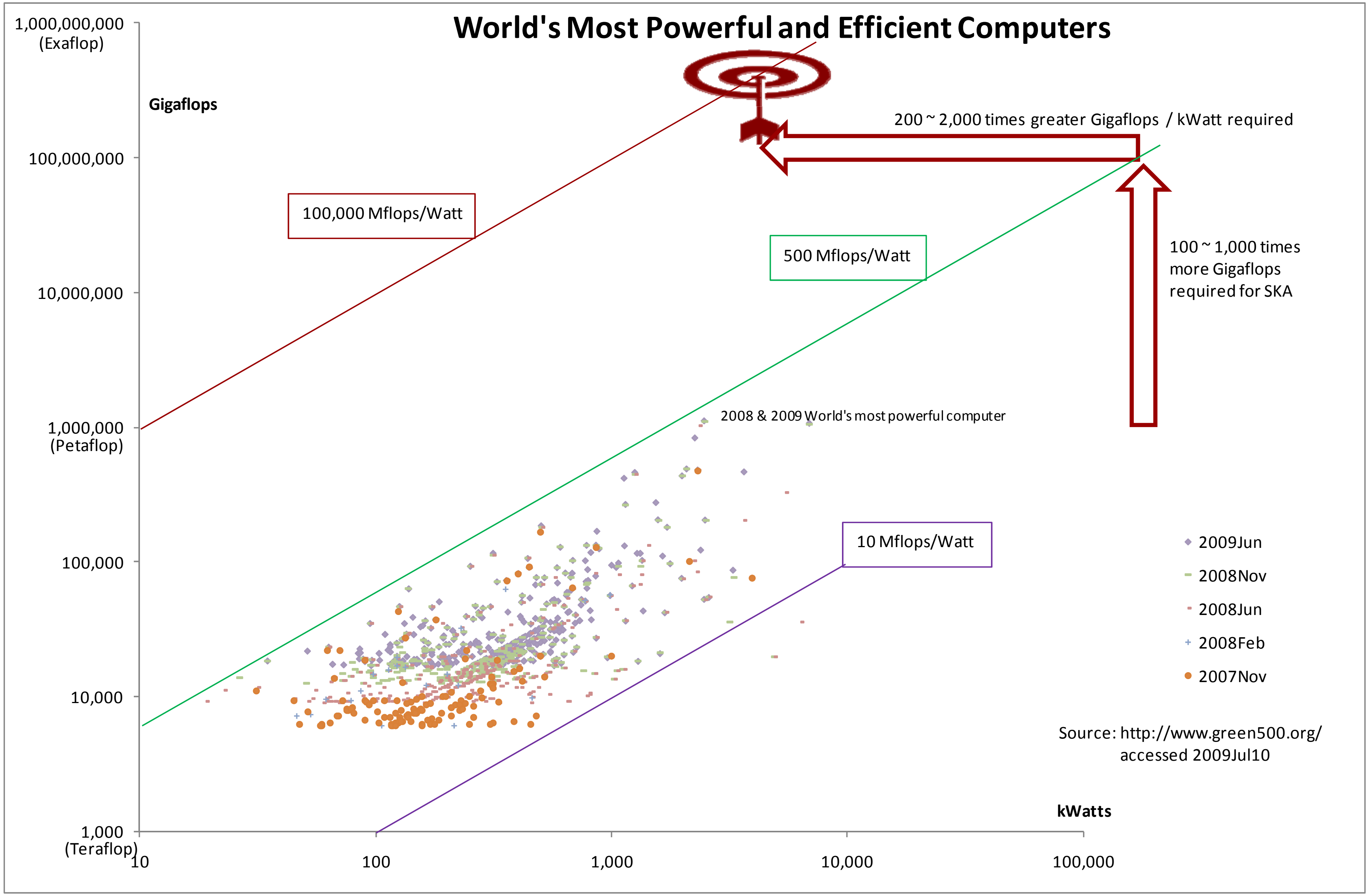
Model and reality of power costs:

Model: Using assumptions similar to prices typically encountered in New Zealand, the HP Labs model calculates that the monthly cost (including software, licences, operations personnel and hardware depreciation – but not including real estate costs) of the servers, power and air conditioning for a continuous server consumption of 10 kW is of the order of NZ$28,000; or about $340,000 [i.e. about €150,000] per year.

Reality: For New Zealand, the annual cost just to deliver power to data centre buildings and building services infrastructure is estimated to be $150,000 per 10 kW consumed. [i.e. €70,000]. This New Zealand power infrastructure cost includes engines, switchboards, Uninterruptible Power Supplies (UPS), air conditioning, seismic structures, lighting etc."

So annualised total cost of ownership for power consumed by data processing equipment in well designed data centres in central business district settings are of the order of €7 to €15 per Watt consumed

Power efficiency challenges:



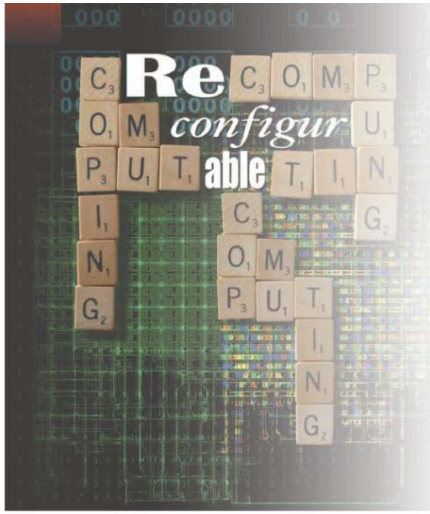
http://www.green500.org accessed July 2009

Required: HPRC architectures – <u>and</u> software

NOVEL ARCHITECTURES

Computing Models for FPGA-Based Accelerators

Field-programmable gate arrays are widely considered as accelerators for compute-intensive applications. A critical phase of FPGA application development is finding and mapping to the appropriate computing model. FPGA computing enables models with highly flexible fine-grained parallelism and associative operations such as broadcast and collective response. Several case studies demonstrate the effectiveness of using these computing models in developing FPGA applications for molecular modeling.



High-Performance Reconfigurable Computing

Duncan Buell, University of South Carolina
Tarek El-Ghazawi, George Washington University
Kris Gaj, George Mason University
Volodymyr Kindratenko, University of Illinois at Urbana-Champaign

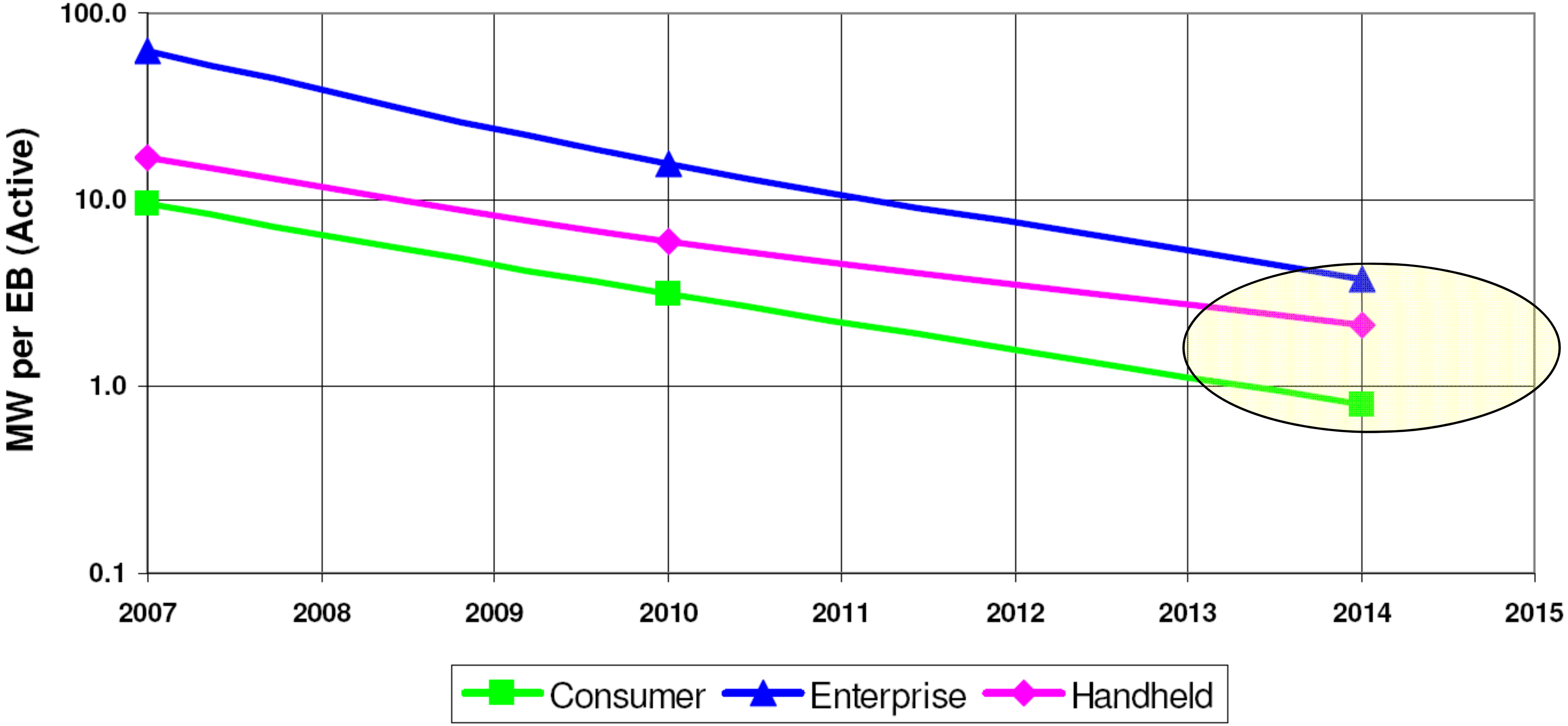
High-performance reconfigurable computers have the potential to exploit coarse-grained functional parallelism as well as fine-grained instruction-level parallelism through direct hardware execution on FPGAs.

High-performance reconfigurable computers (HPRCs)[1,2] based on conventional processors and field-programmable gate arrays (FPGAs)[3] have been gaining the attention of the high-performance computing community in the past few years.[4] These synergistic systems have the potential to exploit coarse-grained functional parallelism as well as fine-grained instruction-level parallelism through direct hardware execution on FPGAs.

HPRCs, also known as reconfigurable supercom-

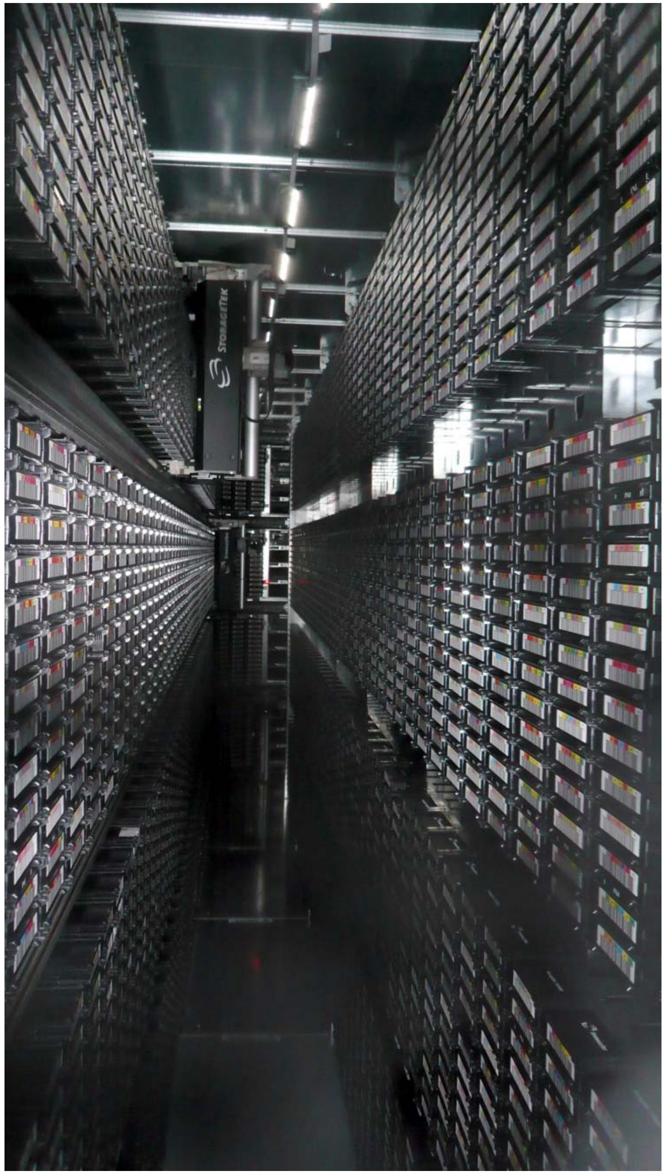
Martin C. Herbordt, et alia, Computing in Science & Engineering, Nov-Dec 2008
Duncan Buell et alia, IEEE Computer, March 2007

Power for EB-size disk looks reasonable



P. Kogge et alia “ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems”; TR-2008-13, DARPA ExaScale Computing Study, 2008 Sep 28, page 124
Note: power numbers here are for the drives only; any electronics associated with drive controllers [e.g. ECC, RAID] needs to be counted separately

Archive: 10 PetaByte tape robot at CERN



500-GB tapes switched to 1-TB models – an upgrade that took a year of continuous load/read/load/write/discard operations, running in the interstices between the data centre’s higher-priority tasks

http://www.flickr.com/photos/doctorow/2711081044/sizes/o/in/set-72157606675048531/
C. Doctorow “Welcome to the Petacentre”; Nature, Vol. 455, $ September 2008, pp. 17-21

Estimating the sizes of the software challenges

An ill-conditioned non-linear problem:

$$Effort\ (€) = \prod_{\substack{\forall P\ \ \forall S \\ P,S>1}} \left\{ \left[\text{Problem space} \right] \otimes \left[\text{Solution space} \right] \right\}$$

where

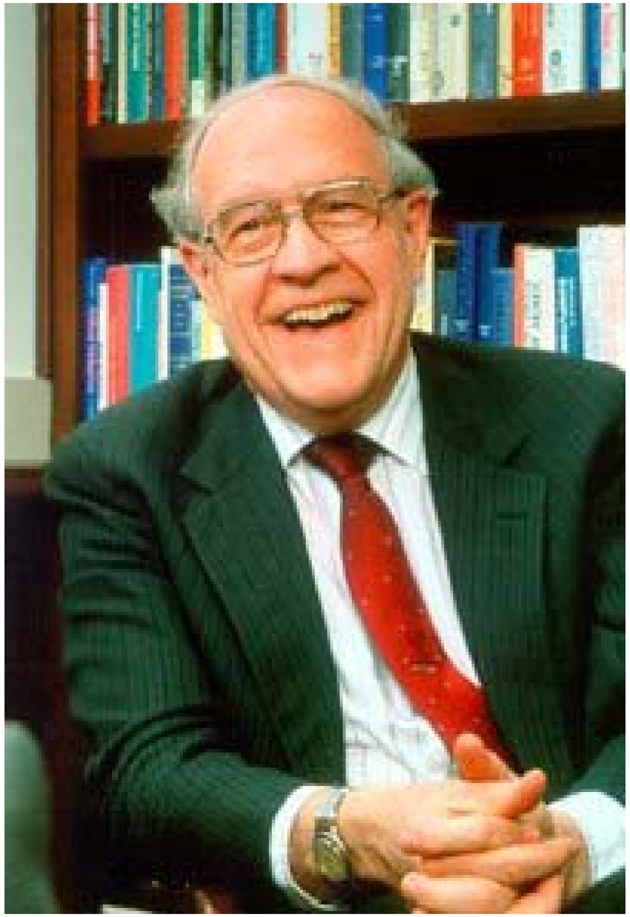
$$\left[\text{Problem space} \right] = \underbrace{\begin{bmatrix} Size\ (+) \\ Complexity\ (+) \\ Time\ (-) \\ Interfaces\ (+) \\ Reliability\ (+) \\ \vdots \end{bmatrix}}_{\text{Requirements}} \times \left[e^{\underbrace{\frac{d}{dt}\left[\text{Problem space} \right]}_{\text{Change requests}}} \right]$$

$$\left[\text{Solution space} \right] = \begin{bmatrix} People\ skills\ (-) \\ Colocation\ (-) \\ Processes\ (-?) \\ Prior\ experience\ (-) \\ Reuse\ opportunities\ (-) \\ Toolsets\ (-?) \\ \vdots \end{bmatrix}$$

1,000+ research studies, experience reports and books ...

... demonstrate that we still really don’t know for sure how to accurately estimate software development:

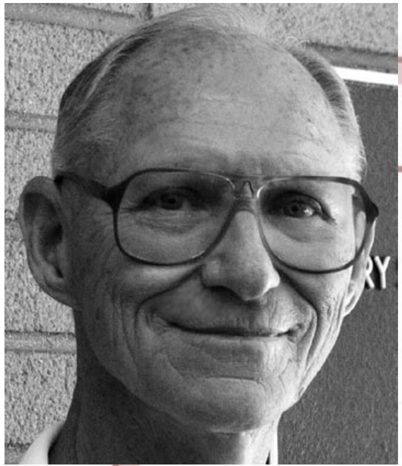
- “ ... we still don’t know what we are doing, unless it is very similar to something we have done before
- The challenge is to make software engineering as predictable a discipline as civil or electrical engineering
- I still do not expect any radical breakthrough, any silver bullet, to solve this problem
- But the accretion of many contributions has already made much progress, and I believe continued careful research, ever validated by real practice, will bring us to that goal”



Magne Jørgensen and Barry Boehm “Software Development Effort Estimation” IEEE Software, March/April 2009
Fred Brooks Jnr.: “Three Great Challenges for Half-Century-Old Computer Science”; Journal of the ACM, Vol. 50, No. 1, January 2003, pp. 25–26

Parametric model example:
COnstructive COst MOdel
COCOMO II

COCOMO II: Effort Equations



COCOMO II.2000 Effort Equations

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_i + PM_{Auto}$$

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

$$PM_{Auto} = \frac{\text{Adapted SLOC} \times \left(AT/100\right)}{ATPROD}$$

Symbol	Description
A	Effort coefficient that can be calibrated
AT	Percentage of the Adapted SLOC that is re-engineered by automatic translation
ATPROD	Automatic translation productivity
B	Scaling base-exponent for Effort that can be calibrated
E	Scaling exponent for Effort
EM	Effort Multipliers: seven (7) for the Early Design and seventeen (17) for the Post-Architecture modes
PM	Person Months effort from developing new and adapted code
PM_{Auto}	Person Months effort from automatic translation activities
SF	Five (5) Scale Factors
SLOC	Source Lines of Code

PM : Person Months of Effort

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_{i}$$

EM_i : Effort Multipliers

Key Process Areas (KPA)	Almost Always[1]	Frequently[2]	About Half[3]	Occasionally[4]	Rarely if Ever[5]	Does Not Apply[6]	Don't Know[7]
Requirements Management • System requirements allocated to software are controlled to establish a baseline for software engineering and management use. • Software plans, products, and activities are kept consistent with the system requirements allocated to software.	☐	☐	☐	☐	☐	☐	☐
Software Project Planning • Software estimates are documented for use in planning and tracking the software project. • Software project activities and commitments are planned and documented. • Affected groups and individuals agree to their commitments related to the software project.	☐	☐	☐	☐	☐	☐	☐
Software Project Tracking and Oversight • Actual results and performances are tracked against the software plans • Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans. • Changes to software commitments are agreed to by the affected groups and individuals.	☐	☐	☐	☐	☐	☐	☐
Software Subcontract Management • The prime contractor selects qualified software subcontractors. • The prome contractor and the subcontractor agree to their commitments to each other. • The prome contractor and the subcontractor maintain ongoing communications. • The prime contractor tracks the subcontractor's actual results and performance against its commitments.	☐	☐	☐	☐	☐	☐	☐
Software Quality Assurance (SQA) • SQA activities are planned. • Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.	☐	☐	☐	☐	☐	☐	☐

(continued)

E : the scaling exponent

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

E : the scaling exponent

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

SF_j : the five Scale Factors

Table 2.10 Scale Factor Values SF_j for COCOMO II Models

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
	---------------------------- The estimated Equivalent Process Maturity Level (EPML) or ----------------------------					
	PMATSW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

RESL: Architecture / Risk Resolution

Table 2.13 RESL Rating Levels

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR or LCA.	None	Little	Some	Generally	Mostly	Fully
Schedule, budget, and internal milestones through PDR or LCA compatible with Risk Management Plan.	None	Little	Some	Generally	Mostly	Fully
Percent of development schedule devoted to establishing architecture, given general product objectives.	5	10	17	25	33	40
Percent of required top software architects available to project.	20	40	60	80	100	120
Tool support available for resolving risk items, developing and verifying architectural specs.	None	Little	Some	Good	Strong	Full
Level of uncertainty in key architecture drivers: mission, user interface, COTS, hardware, technology, performance.	Extreme	Significant	Considerable	Some	Little	Very Little
Number and criticality of risk items.	> 10 Critical	5–10 Critical	2–4 Critical	1 Critical	> 5 Non-Critical	< 5 Non-Critical

Further COCOMO II formulations: Size

COCOMO II.2000 Sizing Equations

$$\text{Size} = \left(1 + \frac{\text{REVL}}{100}\right) \times (\text{New KSLOC} + \text{Equivalent KSLOC})$$

$$\text{Equivalent KSLOC} = \text{Adapted KSLOC} \times \text{AAM} \times \left(1 - {}^{\text{AT}}/_{100}\right)$$

$$\text{AAM} = \begin{cases} \dfrac{\text{AA} + \text{AAF} \times (1 + [0.02 \times \text{SU} \times \text{UNFM}])}{100}, & \text{for AAF} \le 50 \\ \dfrac{\text{AA} + \text{AAF} \times (\text{SU} \times \text{UNFM}])}{100}, & \text{for AAF} > 50 \end{cases}$$

$$\text{AAF} = (0.4 \times \text{DM}) + (0.3 \times \text{CM}) + (0.3 \times \text{IM})$$

Symbol	Description
AA	Percentage of Assessment and Assimilation
AAF	Adaptation Adjustment Factor
AAM	Adaptation Adjustment Modifier
AT	Percentage of the Adapted KSLOC that is re-engineered by automatic translation
CM	Percent Code Modified
DM	Percent Design Modified
IM	Percent of Integration Integration Required for the Adapted Software
KSLOC	Thousands of Source Lines of Code
REVL	Percentage of Requirements Evolution and Volatility
SU	Percentage of Software Understanding
UNFM	Programmer Unfamiliarity with Software

Further COCOMO II formulations: Schedule

COCOMO II.2000 Schedule Equations

$$TDEV = [C \times (PM_{NS})^F] \times \frac{SCED\%}{100}$$

$$F = D + 0.2 \times [E - B]$$

Symbol	Description
B	The scaling base-exponent for the effort equation
C	Schedule coefficient that can be calibrated
D	Scaling base-exponent for Schedule that can be calibrated
E	The scaling exponent for the effort equation
F	Scaling exponent for Schedule
PM_{NS}	Person Months estimated without the SCED cost driver (Nominal Schedule) and without PM_{Auto}
SCED%	Required Percentage of Schedule Compression relative to Nominal Schedule
TDEV	Time to Develop in calendar months

Further COCOMO II parameters

COCOMO II.2000 Scale Factors and Efforts Multipliers

Drivers	Symbol	XL	VL	L	N	H	VH	XH	Productivity Range[1, 2]
Scale Factors									
PREC	SF1		6.20	4.96	3.72	2.48	1.24	0.00	1.33
FLEX	SF2		5.07	4.05	3.04	2.03	1.01	0.00	1.26
RESL	SF3		7.07	5.65	4.24	2.83	1.41	0.00	1.39
TEAM	SF4		5.48	4.38	3.29	2.19	1.10	0.00	1.29
PMAT	SF5		7.80	6.24	4.68	3.12	1.56	0.00	1.43
Post-Architecture Effort Multipliers									
RELY	EM1		0.82	0.92	1.00	1.10	1.26		1.54
DATA	EM2			0.90	1.00	1.14	1.28		1.42
CPLX	EM3		0.73	0.87	1.00	1.17	1.34	1.74	2.38
RUSE	EM4			0.95	1.00	1.07	1.15	1.24	1.31
DOCU	EM5		0.81	0.91	1.00	1.11	1.23		1.52
TIME	EM6				1.00	1.11	1.29	1.63	1.63
STOR	EM7				1.00	1.05	1.17	1.46	1.46
PVOL	EM8			0.87	1.00	1.15	1.30		1.49
ACAP†	EM9		1.42	1.19	1.00	0.85	0.71		2.00
PCAP†	EM10		1.34	1.15	1.00	0.88	0.76		1.76
PCON	EM11		1.29	1.12	1.00	0.90	0.81		1.51
APEX	EM12		1.22	1.10	1.00	0.88	0.81		1.51
PLEX	EM13		1.19	1.09	1.00	0.91	0.85		1.40
LTEX	EM14		1.20	1.09	1.00	0.91	0.84		1.43
TOOL	EM15		1.17	1.09	1.00	0.90	0.78		1.50
SITE	EM16		1.22	1.09	1.00	0.93	0.86	0.80	1.53
SCED	EM17		1.43	1.14	1.00	1.00	1.00		1.43
Early Design Effort Multipliers									
RCPX	EM1	0.49	0.60	0.83	1.00	1.33	1.91	2.72	5.55
RUSE	EM2			0.95	1.00	1.07	1.15	1.24	1.31
PDIF	EM3			1.00	1.00	1.00			1.00
PERS	EM4	2.12	1.62	1.26	1.00	0.83	0.63	0.50	4.24
PREX	EM5	1.59	1.33	1.12	1.00	0.87	0.74	0.62	2.56
FCIL	EM6	1.43	1.30	1.10	1.00	0.87	0.73	0.62	2.31
SCED	EM7		1.43	1.14	1.00	1.00	1.00		1.43

For Effort Calculations:
Multiplicative constant A = 2.94;
Exponential constant B = 0.91

For Schedule Calculations:
Multiplicative constant C = 3.67;
Exponential constant D = 0.28

[1]For Scale Factors:

$$PR_{SF_n} = \frac{(100)^{0.91 + (0.01 \times SF_{n_{MAX}})}}{(100)^{0.91}}$$

[2]For Effort Multipliers:

$$PR_{EM_n} = \frac{EM_{n_{MAX}}}{EM_{n_{MIN}}}$$

†PR for Personnel/team capability shown on the front cover is determined as a combination of ACAP and PCAP:

$$PR_{Personnel/team\ capacity} = \frac{ACAP_{MAX} \times PCAP_{MAX}}{ACAP_{MIN} \times PCAP_{MIN}}$$

COCOMO II.2000 Post-Architecture Cost Driver Descriptions

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
RELY	slight inconvenience	low, easily recoverable losses	Moderate, easily recoverable losses	high financial loss	risk to human life	
DATA		(testing DB bytes / Pgm SLOC) < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P > 1000	
CPLX	See Table 2.19					
RUSE		none	across project	across program	across product line	across multiple product lines
DOCU	many lifecycle needs uncovered	some lifecycle needs uncovered	correct amount for lifecycle needs	excessive for lifecycle needs	very excessive for lifecycle needs	
TIME			≤ 50% use of available execution time	70% use	85% use	95% use
STOR			≤ 50% use of available storage	70% use	85% use	95% use
PVOL		major change every 12 mo.; minor change every 1 mo.	major: 6 mo.; minor: 2 wk.	major: 2 mo.; minor: 1 wk.	major: 2 wk.; minor: 2 days	
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	
APEX	≤ 2 months	6 months	1 year	3 years	6 years	
PLEX	≤ 2 months	6 months	1 year	3 years	6 years	
LTEX	≤ 2 months	6 months	1 year	3 years	6 years	
TOOL	edit, code, debug	simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
SITE: Collocation	international	multi-city and multi-company	multi-city or multi-company	same city or metro area	same building or complex	fully collocated
SITE: Communication	some phone, mail	individual phone, FAX	narrow-band email	wide-band electronic communication	wide-band elect. comm, occasional video conf.	interactive multimedia
SCED	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	

COCOMO II.2000 Post-Architecture Scale Factor Descriptions

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	Somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
FLEX†	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL‡	little (20%)	some (40%)	Often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
	or the estimated Process Maturity Level					

† degree of required adherence to stated system requirements, budgets, and schedules.
‡ percentage of module interfaces specified, subjectively averaged with the percentage of known significant risks mitigated.

Formal models – or Expert judgement?

Formal models or Expert judgement? 1of3

- **In spite of massive effort and promotion, available empirical evidence shows that formal estimation models aren't in much use ...**
- **... projects officially applying a formal estimation model actually use the model as a disguise for expert estimation**
- **All meaningful estimation models require judgment to produce the input to the models**
- **... the relation between effort and size in software development contexts isn't stable**
- **In situations involving high cost and schedule uncertainty, it's a good idea to draw upon as many sources of insight as possible**

Magne Jørgensen and Barry Boehm "Software Development Effort Estimation" IEEE Software, March/April 2009

Formal models or Expert judgement? 2of3

- **... software development situations frequently contain highly specific, highly important information ...**
- **... expert judgment can have great advantages in situations with highly specific information that's not mechanically integrated, or integrated at all, in a model**

- **[BB]: "I used to think that closed-loop feedback and recalibration would enable organizations and models to become increasingly perfect estimators.**
- **But I don't any more**
- **The software field continues to reinvent and re-baseline itself too rapidly to enable this to happen"**

Magne Jørgensen and Barry Boehm "Software Development Effort Estimation" IEEE Software, March/April 2009

Formal models or Expert judgement? 3of3

- **A major advantage of a parametric model is that it doesn't modify its estimates when customers, managers, or marketers apply pressure**
- **Using a calibrated parametric model enables negotiation ... rather than a contest of wills between self-described experts**
- **... the usual practice is to discard [cost models] as having served their purpose and to avoid future embarrassment when the estimates are overrun**
- **So, use incremental development and *timeboxing* – also known as *cost and schedule as an independent variable***
- **Simple models typically perform just as well as more advanced models ...**

Magne Jørgensen and Barry Boehm "Software Development Effort Estimation" IEEE Software, March/April 2009

The importance of requirements

“You cannot estimate jobs for which you have not scoped the work”

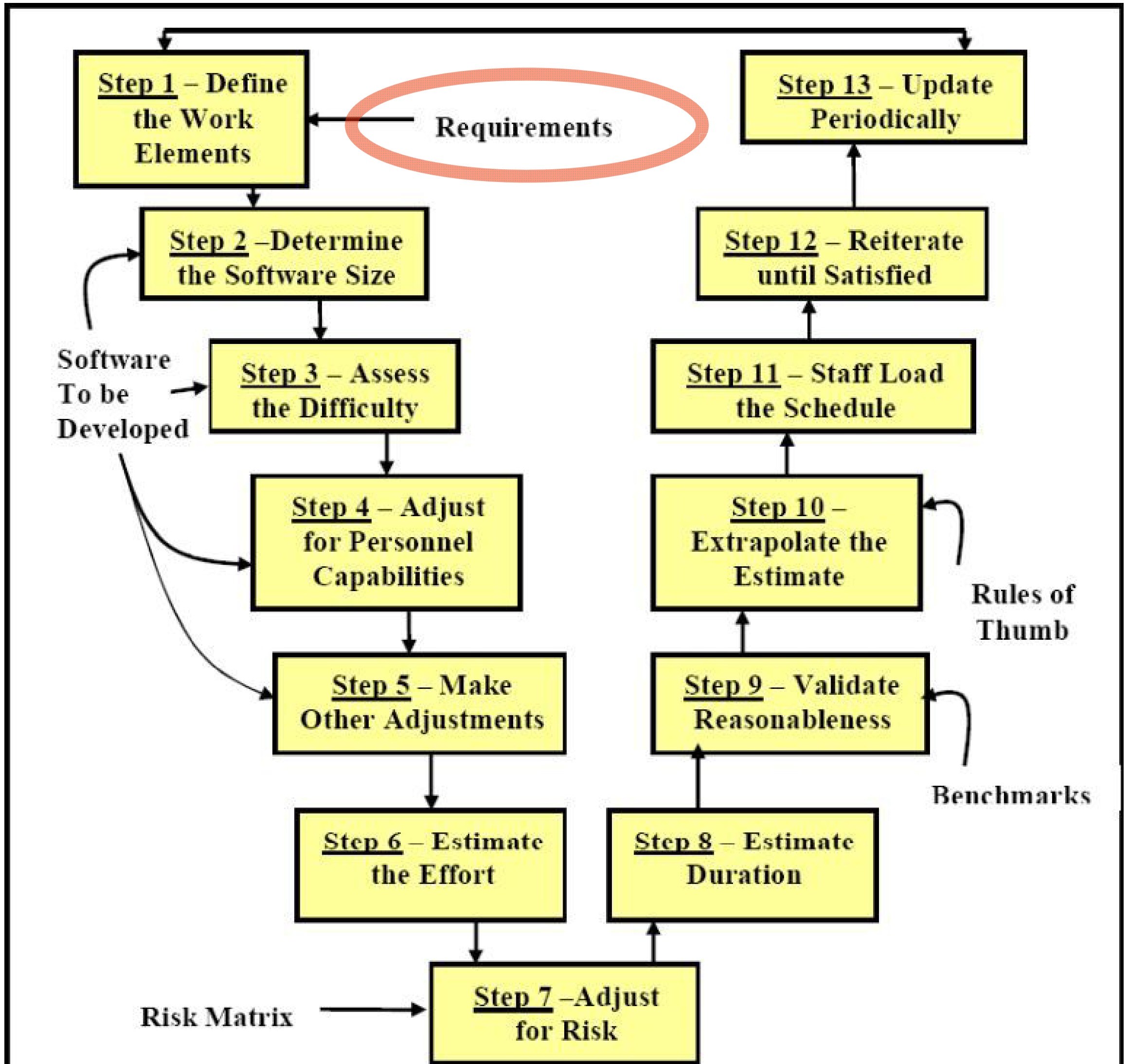


Figure 1. Thirteen-Step Estimating Process

Donald Reifer, “The Poor Person’s Guide to Estimating Software Development Costs”, IEEE

"You cannot estimate jobs for which you have not scoped the work"

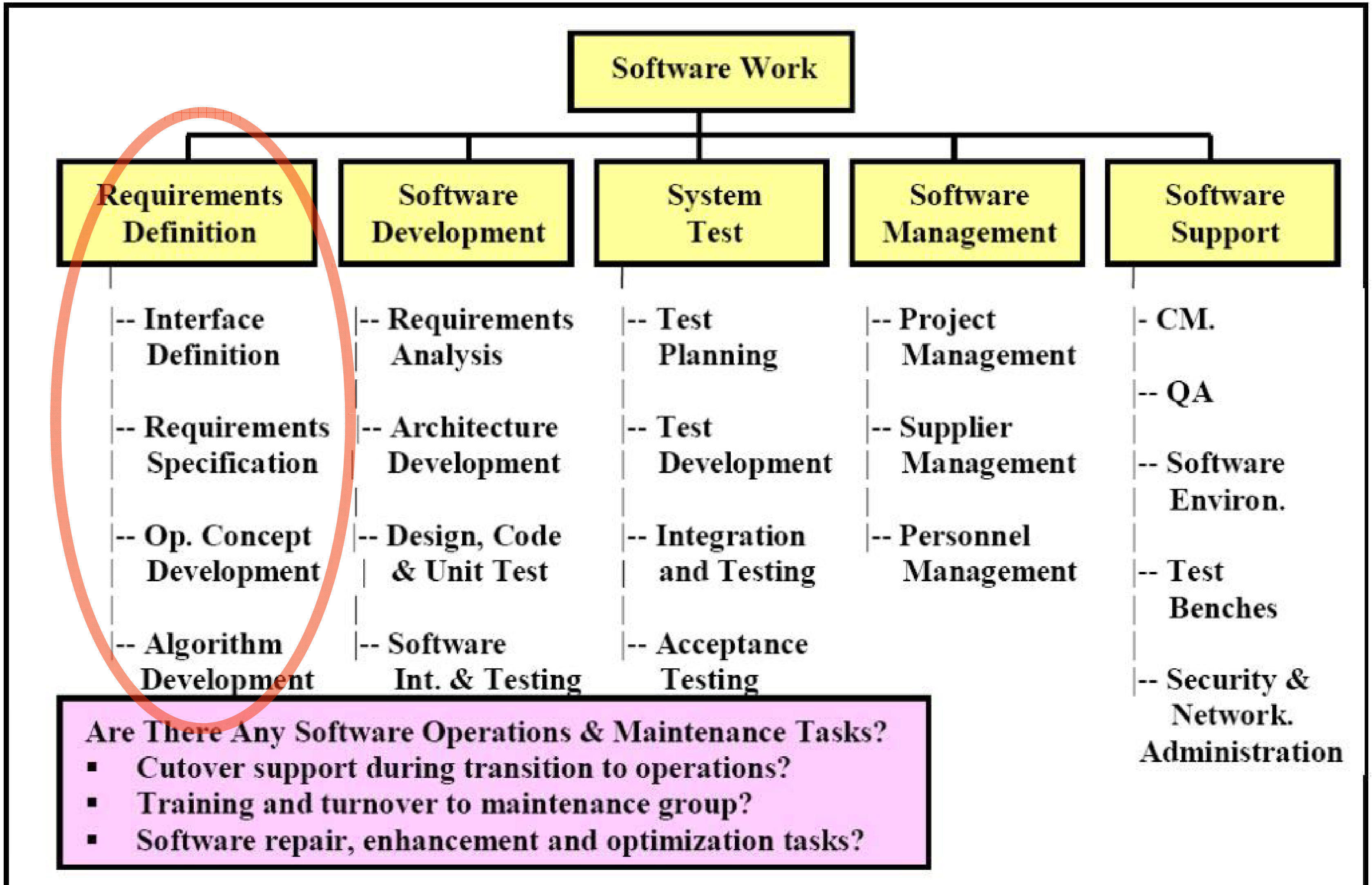


Figure 2. Typical Software Project Work Breakdown Structure (WBS)

Donald Reifer, "The Poor Person's Guide to Estimating Software Development Costs", IEEE

From the Aug 2008 IEEE Software editorial:

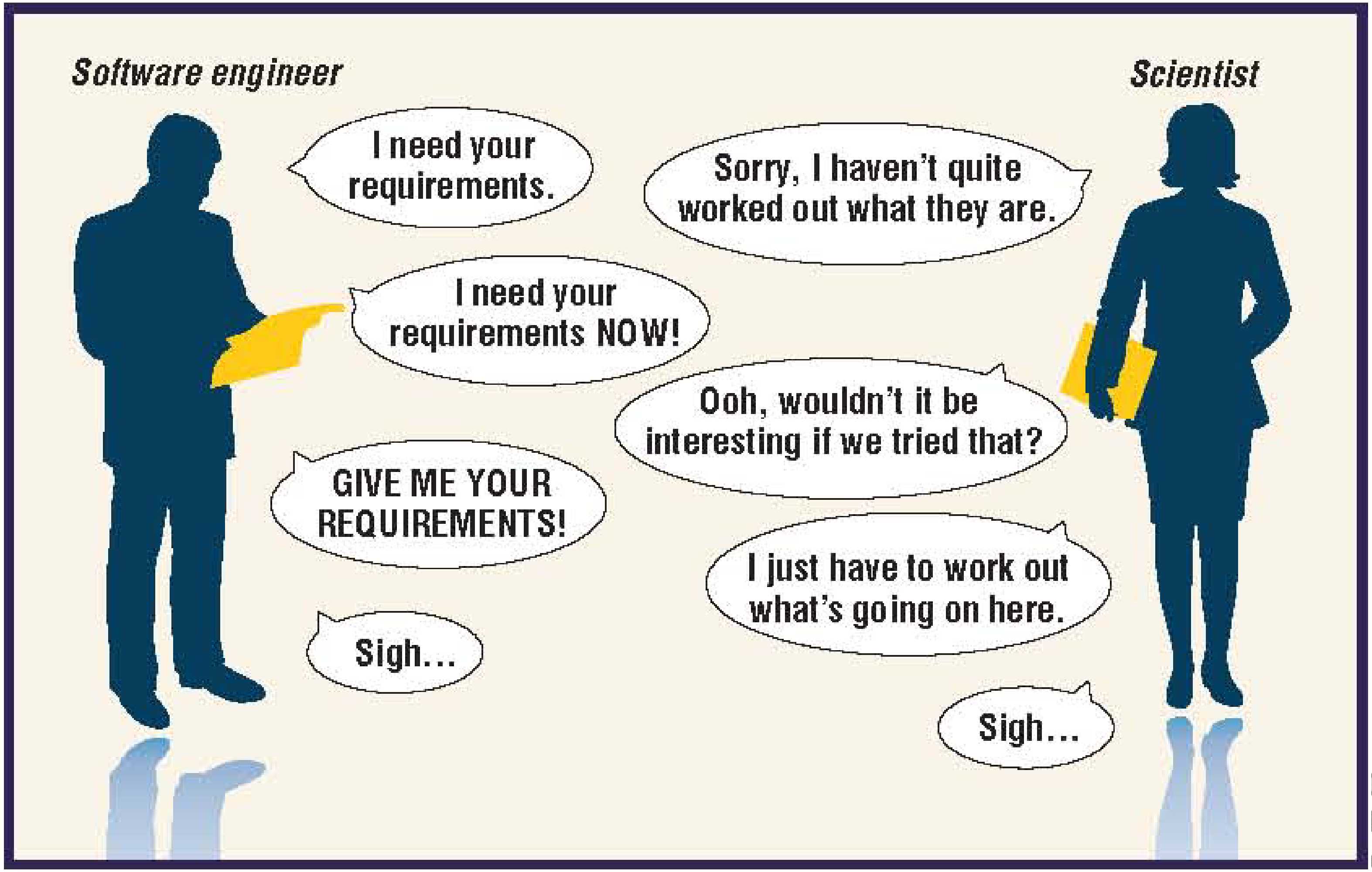


Figure 2. A clash between software engineers and scientists. The former expect requirements to be specified up front; the latter expect them mostly to emerge.

Getting requirements sorted is important ...

- **Glass's Law:**
  - **Requirements deficiencies are the prime source of project failures**

- **Boehm's First Law:**
  - **Errors are most frequent during the requirements and design activities and are more expensive the later they are removed**

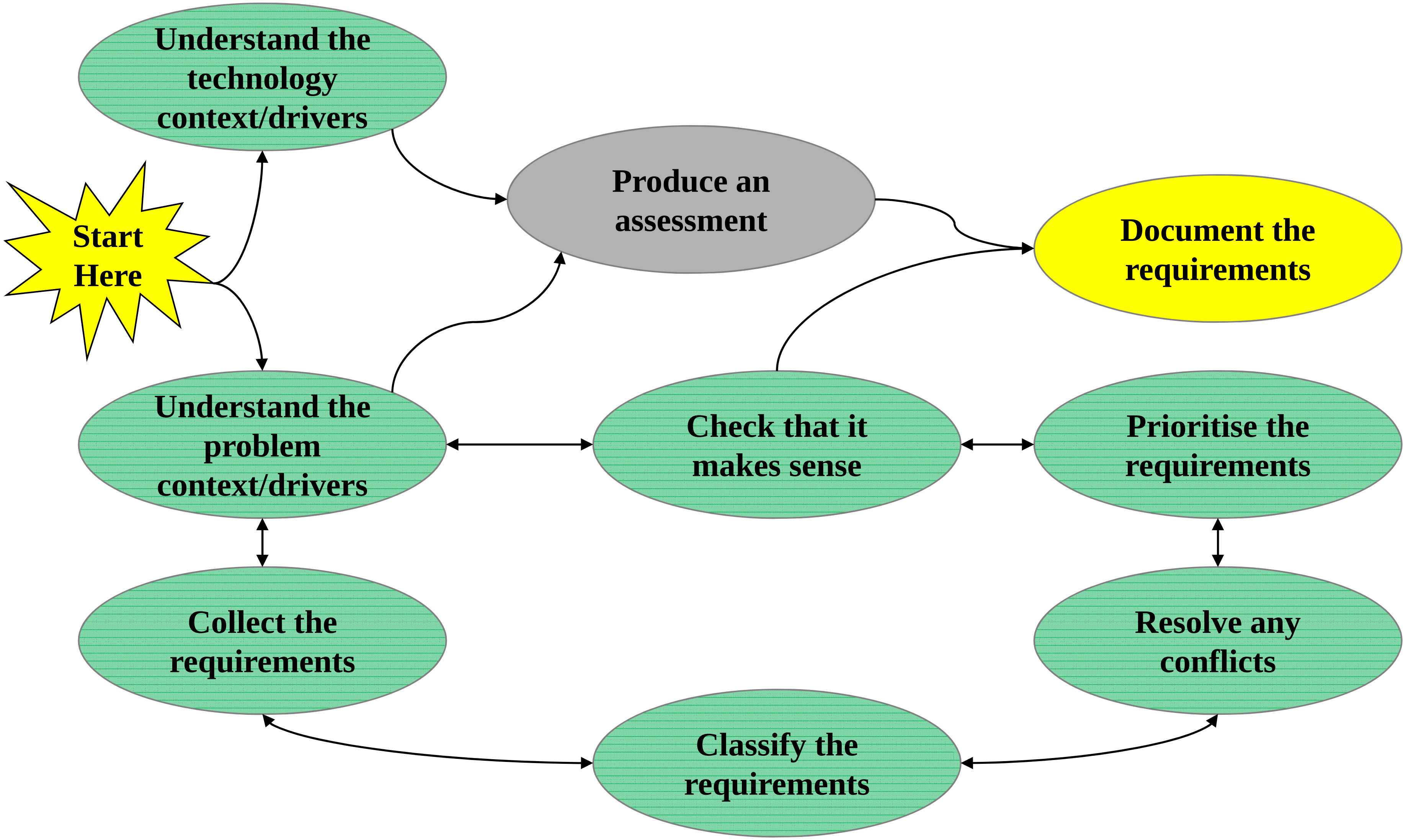
Sources:
Glass, R. L. (1998) *Software Runaways*
Boehm, B. W. et alia (1975) *Some experience* ... IEEE Trans. Software Engineering; 1/1, 125ff
Boehm, B. W. et alia (1984) *Prototyping* ... IEEE Trans. Software Engineering; 10/3, 290ff

... but hard to get right

- **“The hardest single part of building a software system is deciding precisely what to build.**
- **No other part of the conceptual work is as difficult as establishing the requirements.**
- **...**
- **No other part of the work so cripples the resulting system if done wrong.**
- **No other part is as difficult to rectify later.”**

Source: Fred Brooks (1987) *No Silver Bullet*

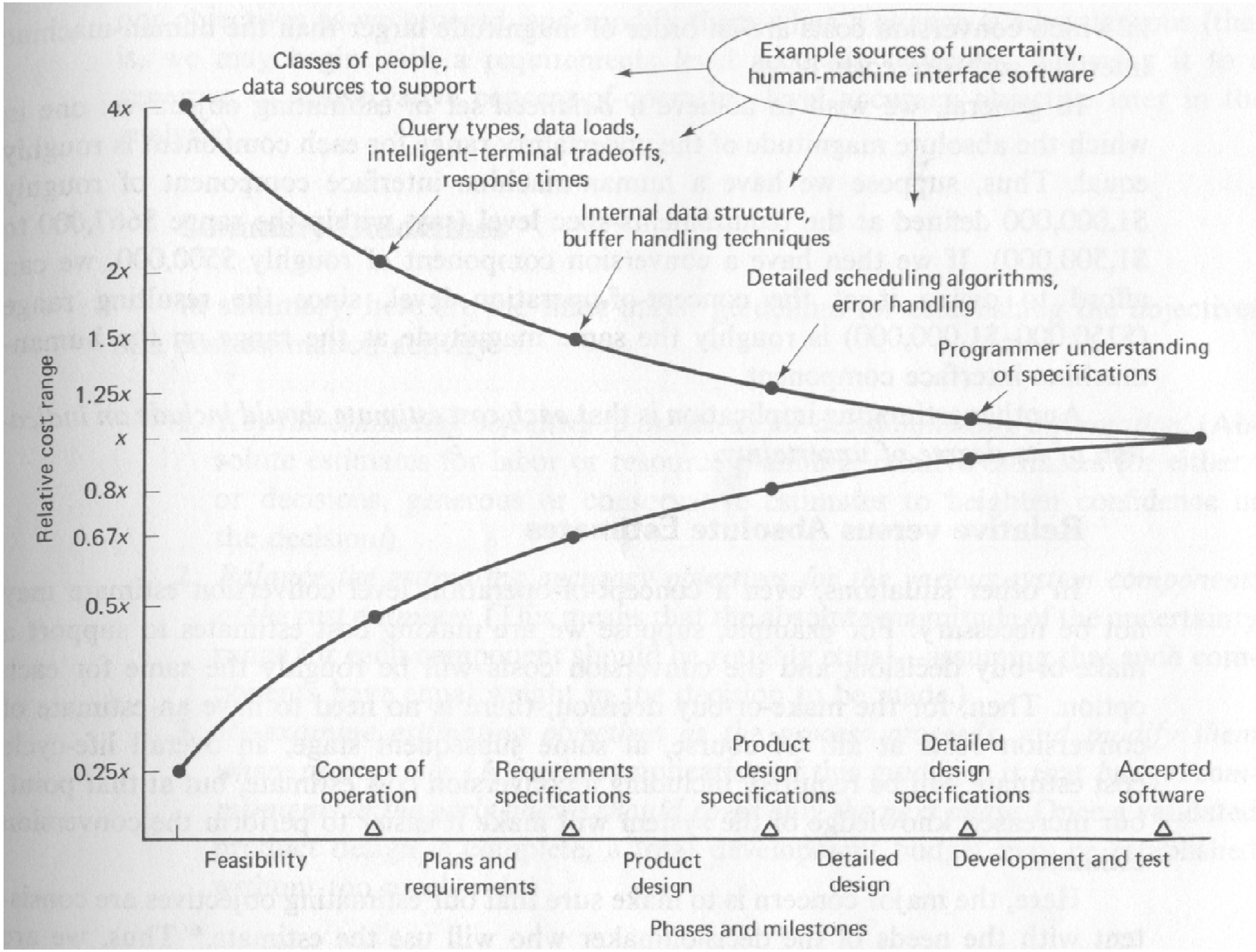
Eliciting requirements requires dialogue, analysis and iteration:



Estimation uncertainties

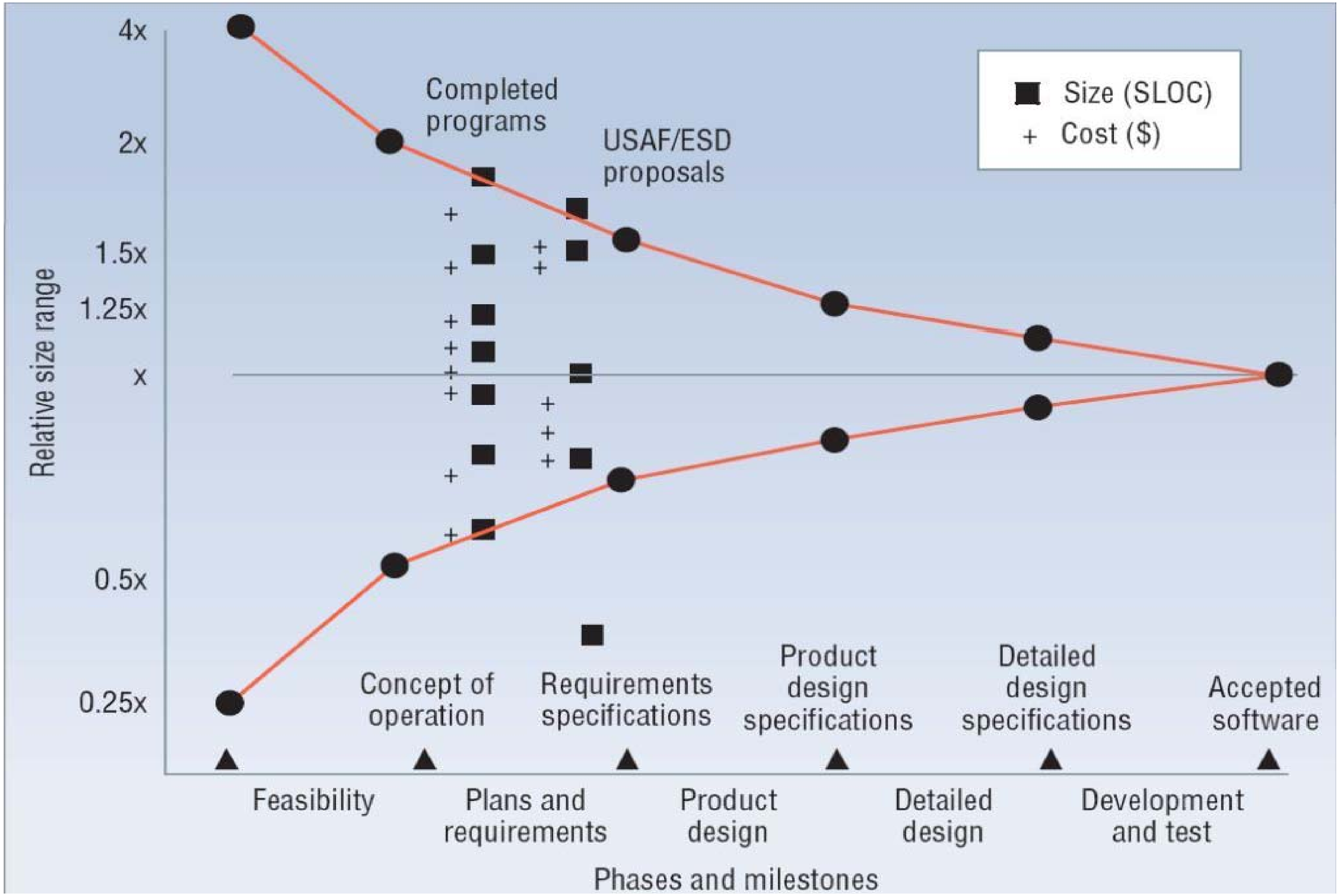
Estimation uncertainties

The Boehm-McConnell "cone of uncertainty"



1981

The Boehm-McConnell "cone of uncertainty"



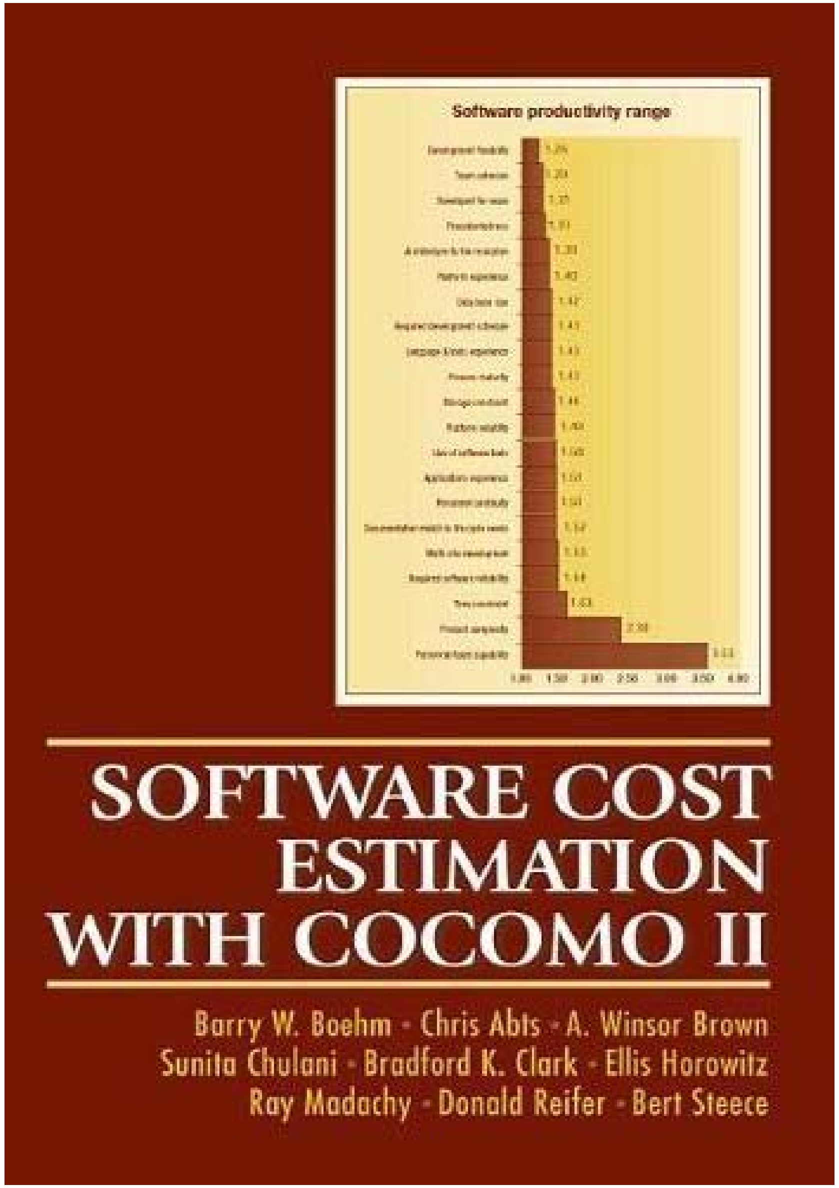
2008

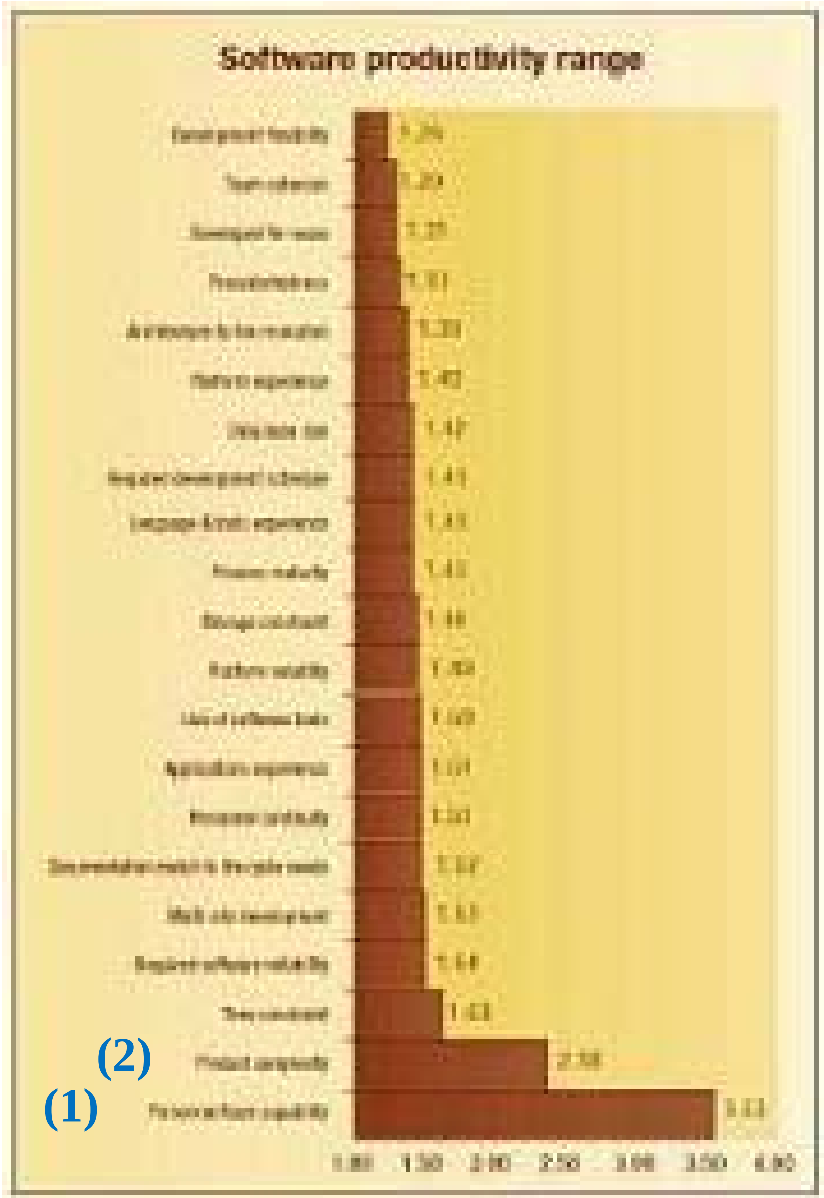
Can large software really be <u>that</u> hard?

Yes!

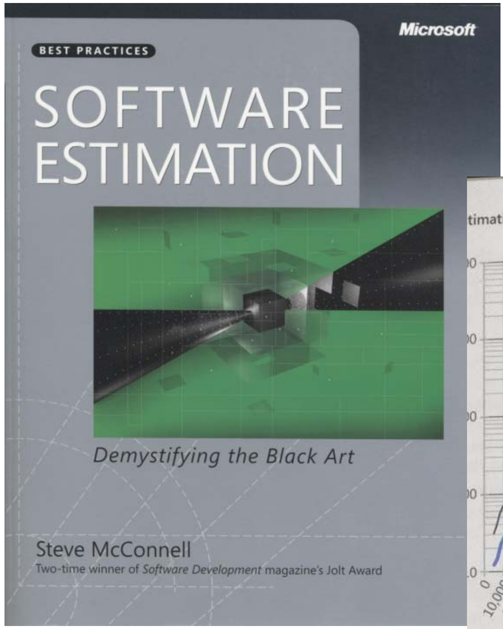
- First-order model for estimating effort
- Diseconomies of scale
- Confirmation from the literature

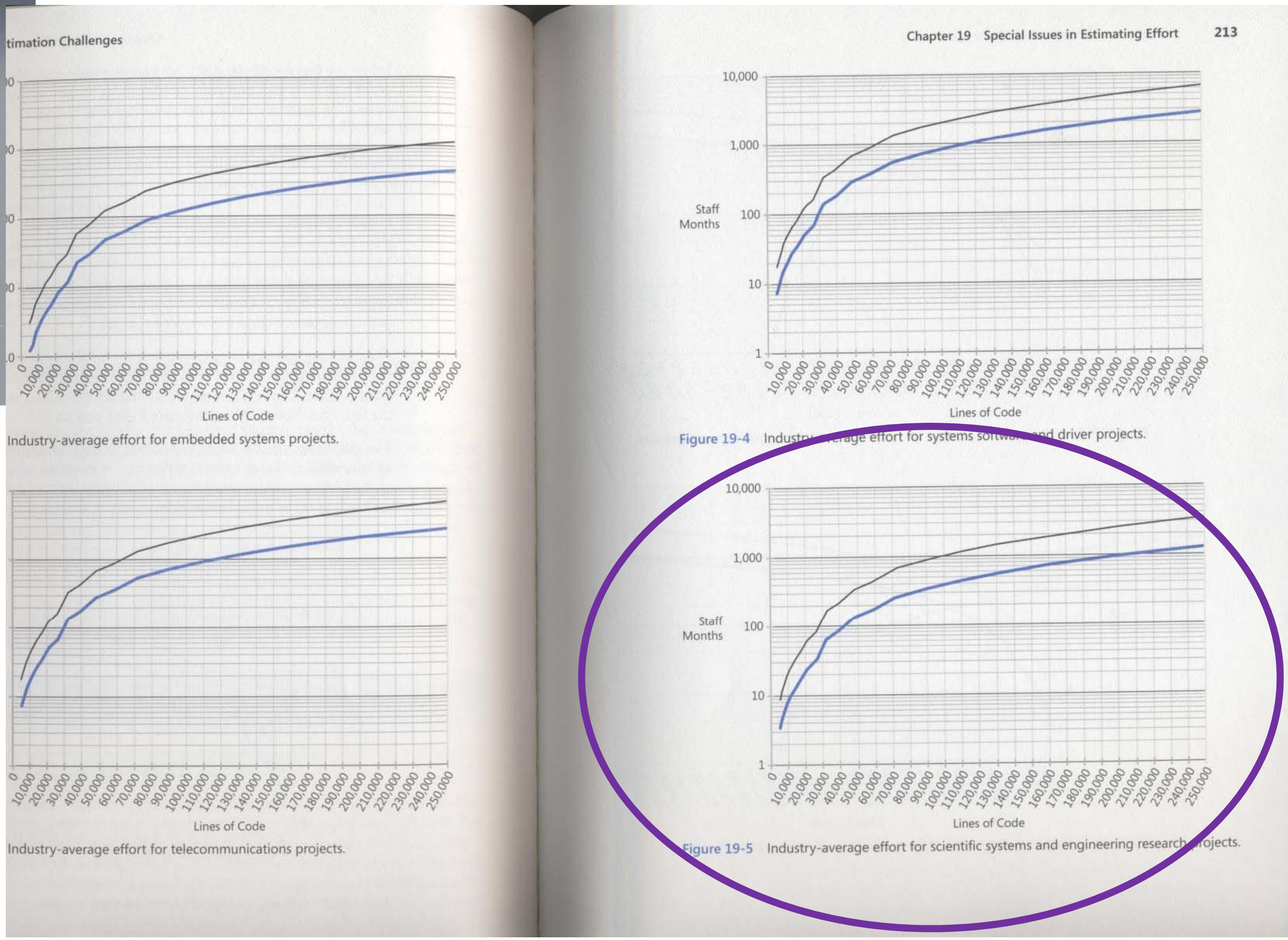
Two drivers dominate the COCOMO II cost formulation:
(1) Personnel / Team Capability and (2) Product Complexity



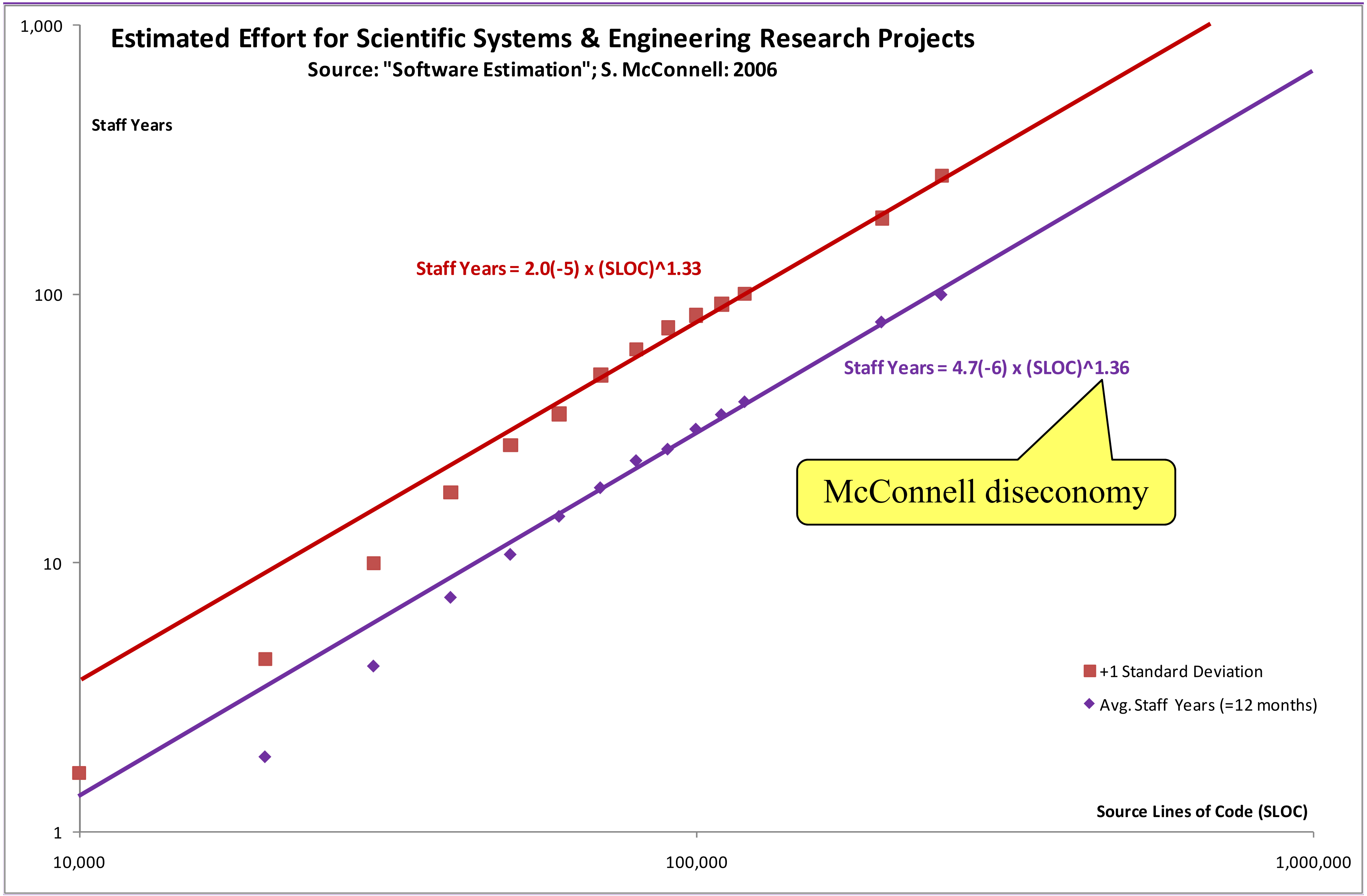


First-order model for estimating effort

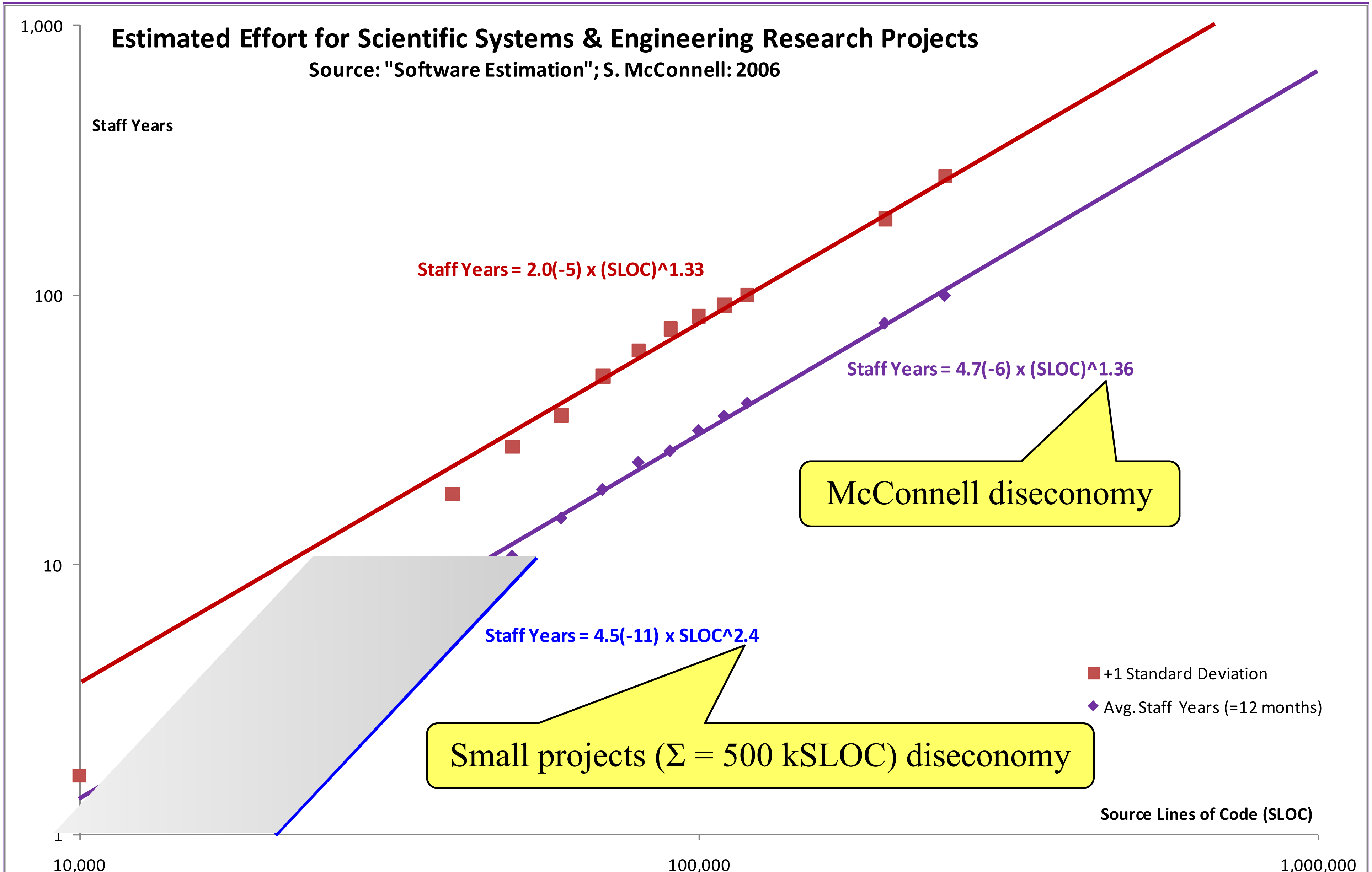




McConnell's data on log-log axes

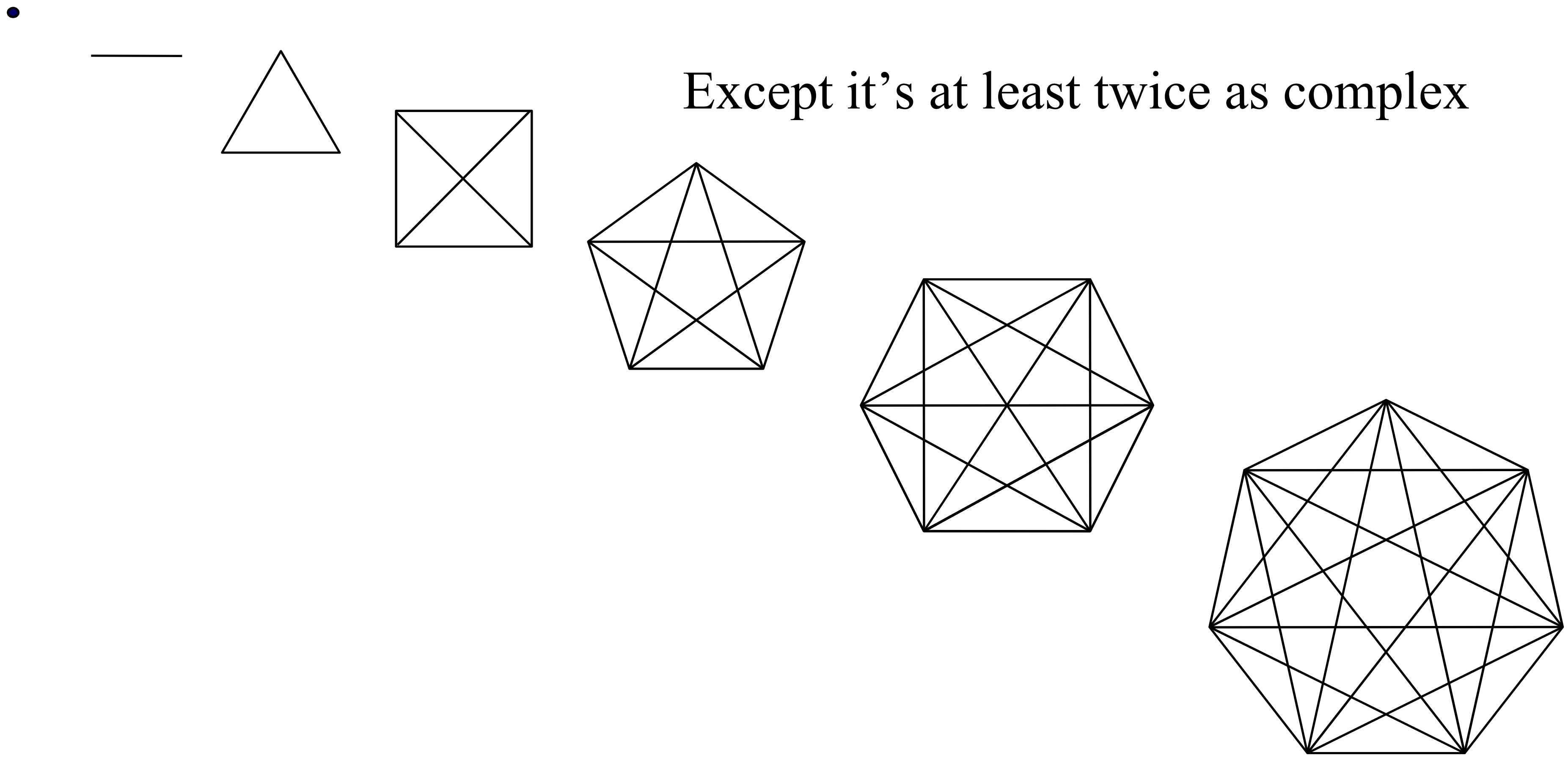


“Small” projects study c.f. McConnell data



http://www.skatelescope.org/pages/WP2_Meeting_10Nov08/Day2-3_Hall.pdf

Collaboration is just like correlation:



Except it's at least twice as complex

- [Without modularisation] Must establish, coordinate and regularly use and maintain ~n^2 links
- So worst-case diseconomy of scale likely to have slope >2 on log-log effort-size charts

How big are the “legacy” codes?

Table I. Sample community codes for radio astronomy imaging.

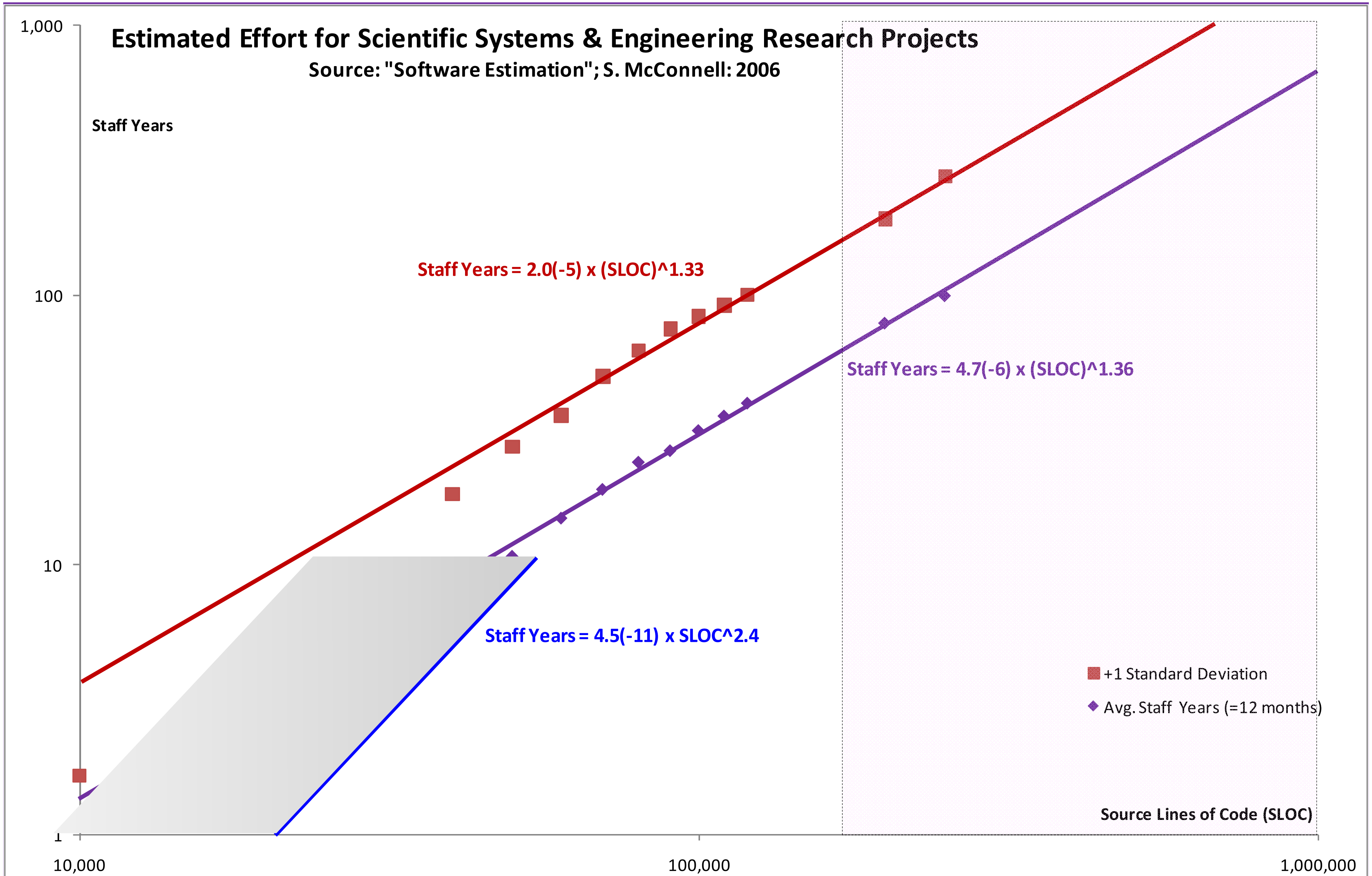
Package name	Development languages (ordered by prevalence)	Size (MSLOC[a])
Astronomical Image Processing System (AIPS)[b]	Fortran 77, C	0.6
Multi-channel Image Reconstruction, Image Analysis, and Display (MIRIAD)[c]	Fortran 77, C	0.2
Astronomical Information Processing System (AIPS++)[d]	C++, Glish [14], Fortran 77	1.0

[a]MSLOC = 10^6 SLOC, as measured by SLOCCount (written by David A. Wheeler).
[b]Modified version of base release 15OCT97.
[c]Release v4.
[d]Modified version of code base v1.8 #667.

MSLOC:	
Debian 4.0	283
Mac OS X 10.4	86
Vista	50
Linux kernel 2.6.29	11
OpenSolaris	10

Kemball, R. M. Crutcher, R. Hasan: “A component-based framework for radio-astronomical imaging software systems” Software Practice and Experience: 2008; 38: pp 493–507
http://en.wikipedia.org/wiki/Source_lines_of_code

Legacy: ~50 to ~700+ staff years effort?

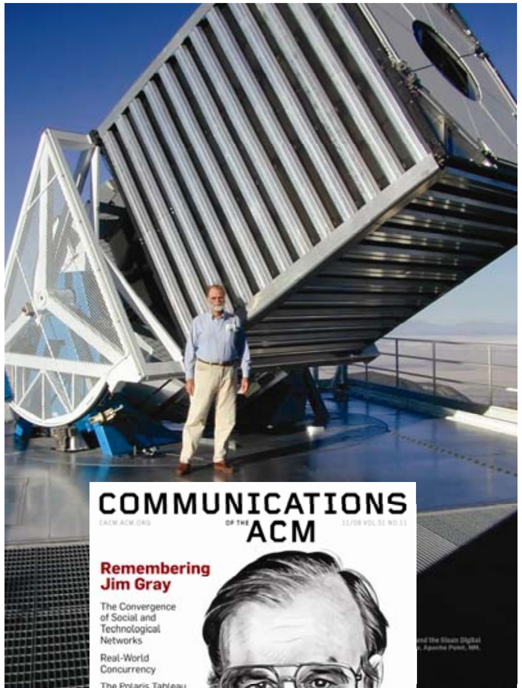


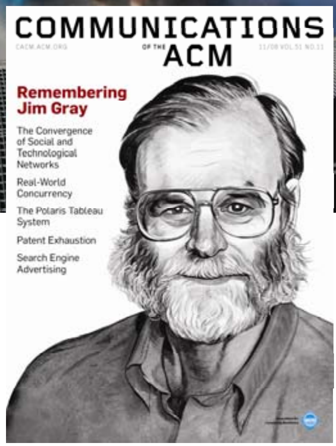
http://www.skatelescope.org/pages/WP2_Meeting_10Nov08/Day2-3_Hall.pdf;

Human frailty:

- We must work together to complete large projects in reasonable time, and have other people try to catch our mistakes
- Once we start working together, we face other problems
- The natural language we use to communicate is wonderfully expressive, but frequently ambiguous
- Our human memory is good, but not quite deep and precise enough to remember a project's myriad details
- We are unable to track what everyone is doing in a large group, and so risk duplicating or clobbering the work of others
- Large systems can often be realised in multiple ways, hence engineers must converge on a single architecture and design

SDSS: Gray and Szalay

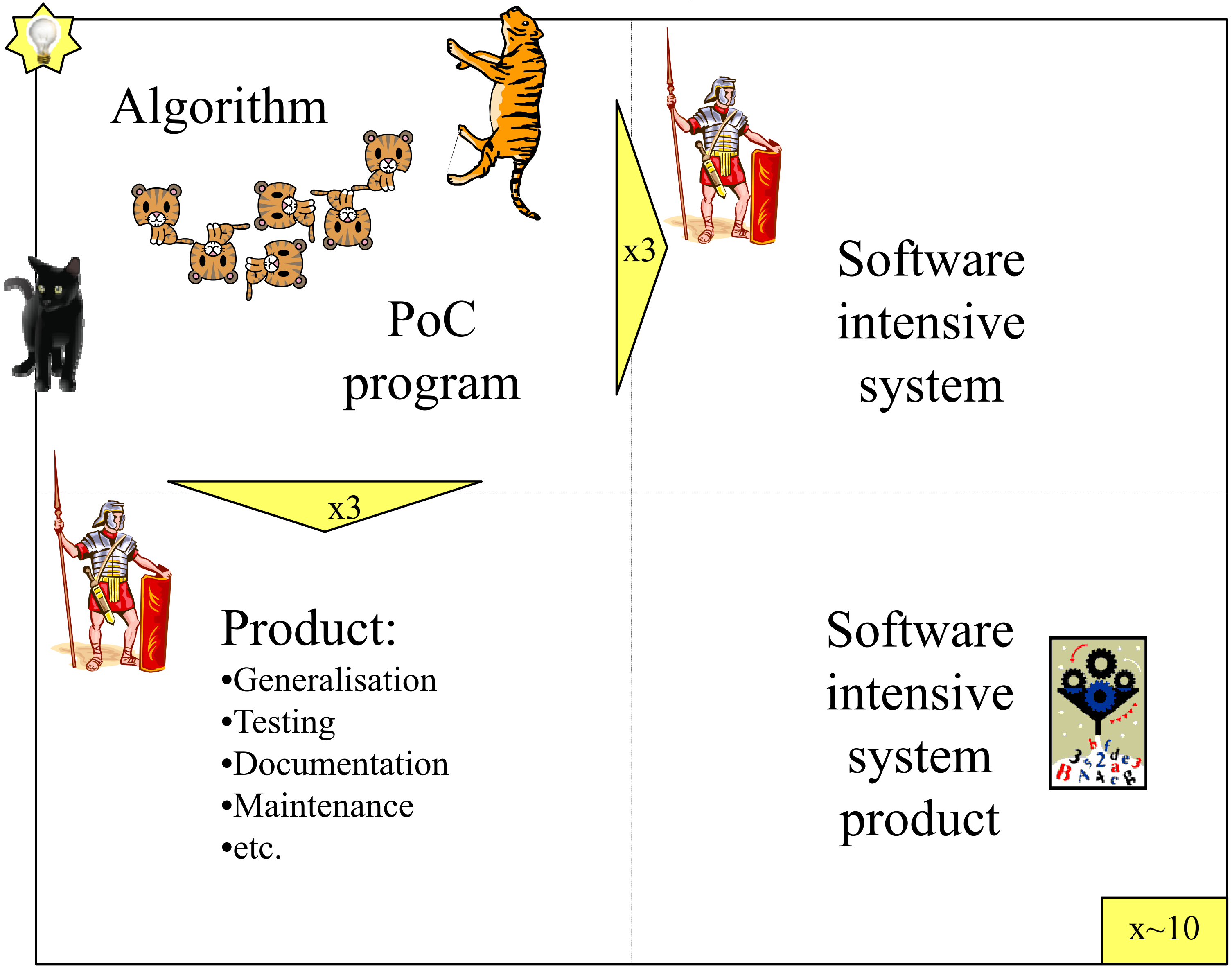




Where the Rubber Meets the Sky:
Bridging the Gap between Databases and Science
MSR-TR-2004-110: 2004 October

- One problem the large science experiments face is that **software is an out-of-control expense**
- They **budget 25% or so for software and end up paying a lot more**
- The **extra software costs are often hidden** in other parts of the project – the instrument control system software may be hidden in the instrument budget

A software intensive production system is much more than the initial algorithm:



Fred Brooks; "The Mythical Man-Month – Essays on Software Engineering Anniversary Edition": 1995

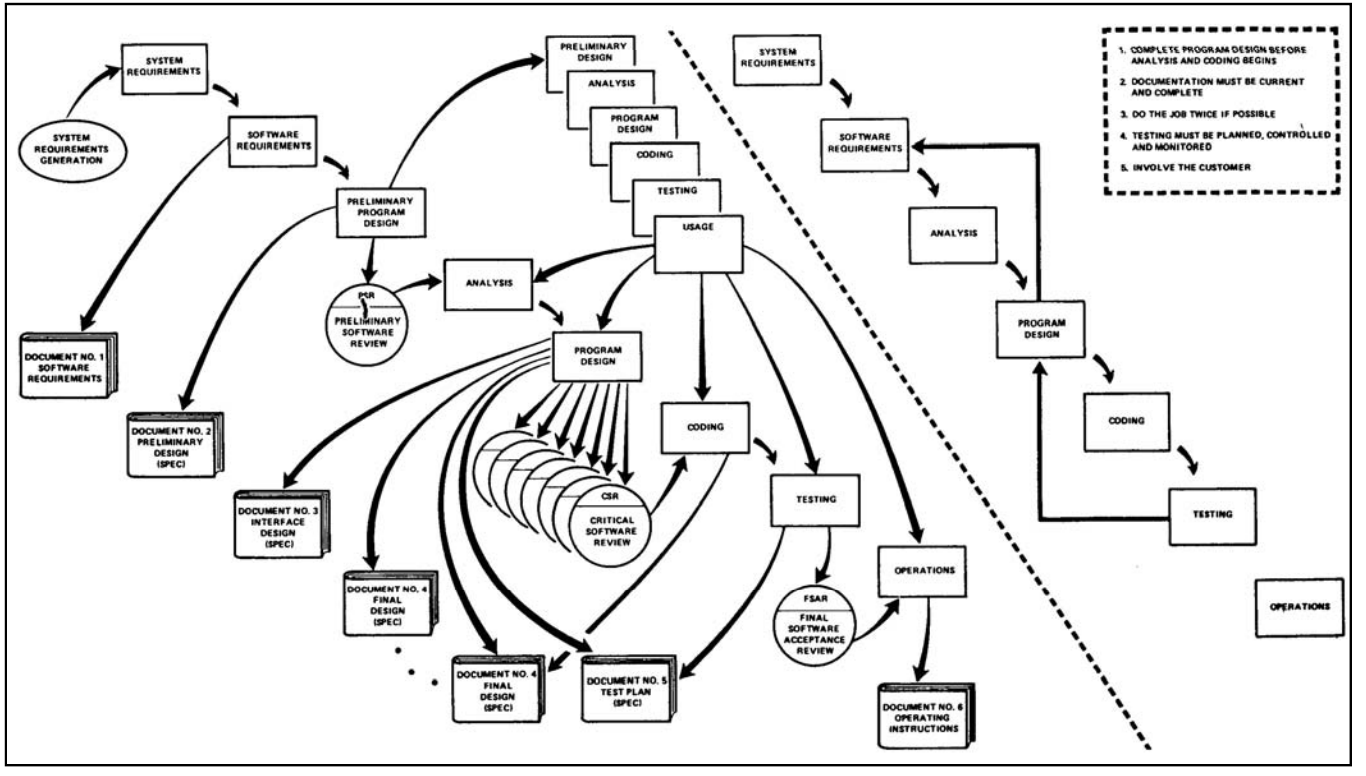
We can’t “wish” the hardness away

And why not? Three answers:

- It’s been around a long time
- There are no silver bullets
- Bad things happen if we rely solely on wishes and prayers ...
  - But of course, any assistance may be of help

Myth busting – number 1:

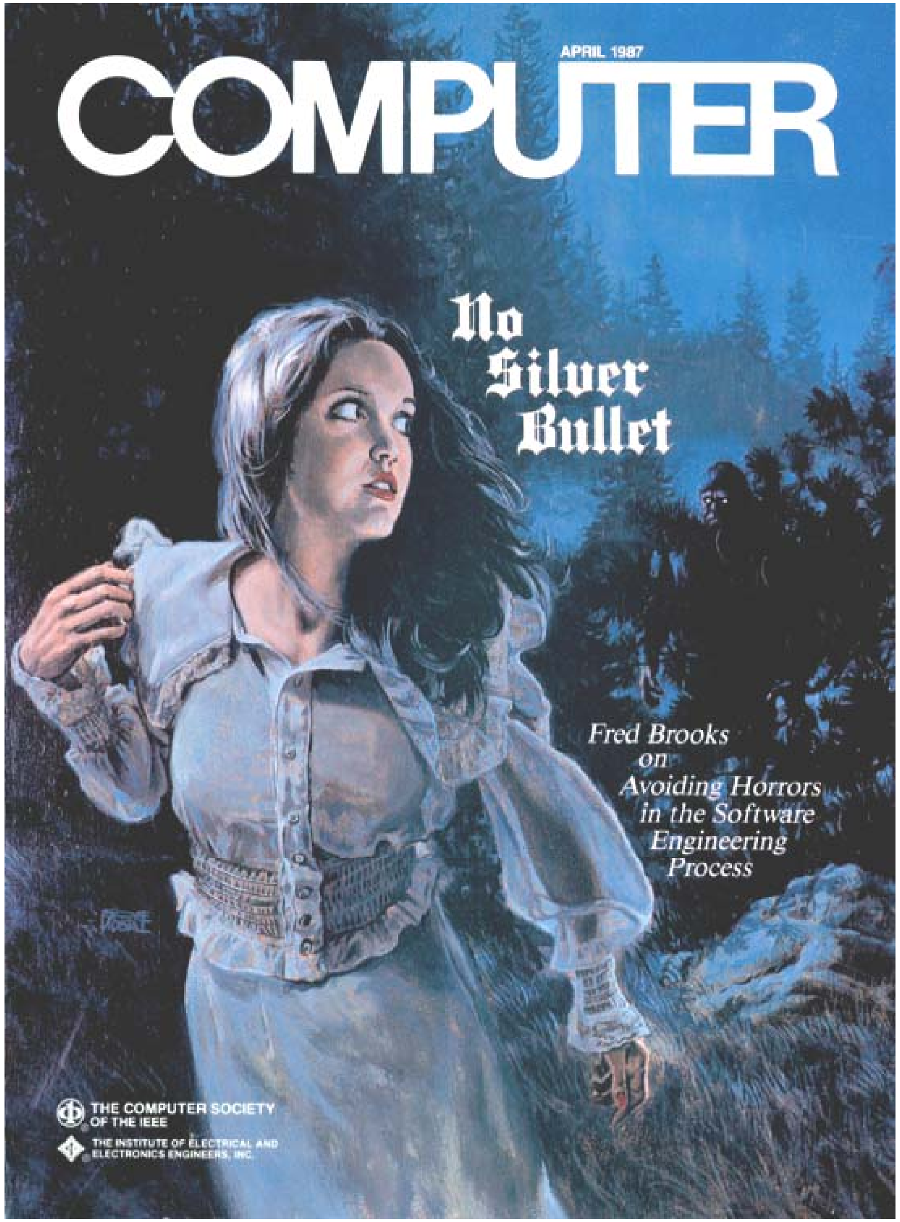
- **The myth**
  - The old guys used "waterfall" - also expressed as "traditional software engineering"
  - We are a lot better now
- **The reality?**
  - They were giants in the old days: Parnas, Jackson, Brooks ...
  - As documented in 1970 by Royce, aspects of the mis-named "waterfall" are very similar to today's "agile"



IEEE WESCON, Aug 1970, pp 1-9

Myth busting – number 2:

- **The myth**
  - Modern approach 'X' will slay the Werewolf of intractable software development
- **The reality?**
  - **<u>There is no silver bullet</u>:** Brooks' "essential" hardness is ever-present
  - Various brass and lead bullets do reasonable jobs to address "accidental" hardness – but each has its own risks and required overhead

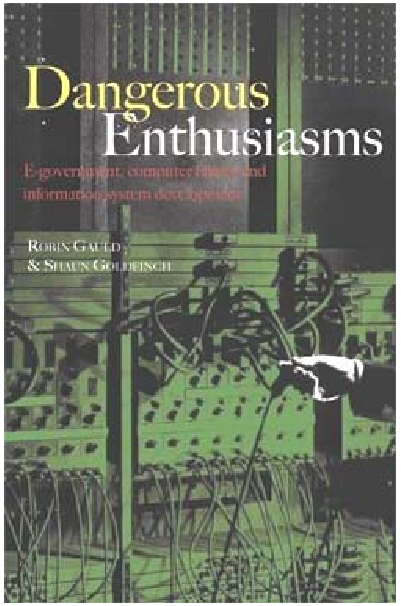


Myth busting – number 3:

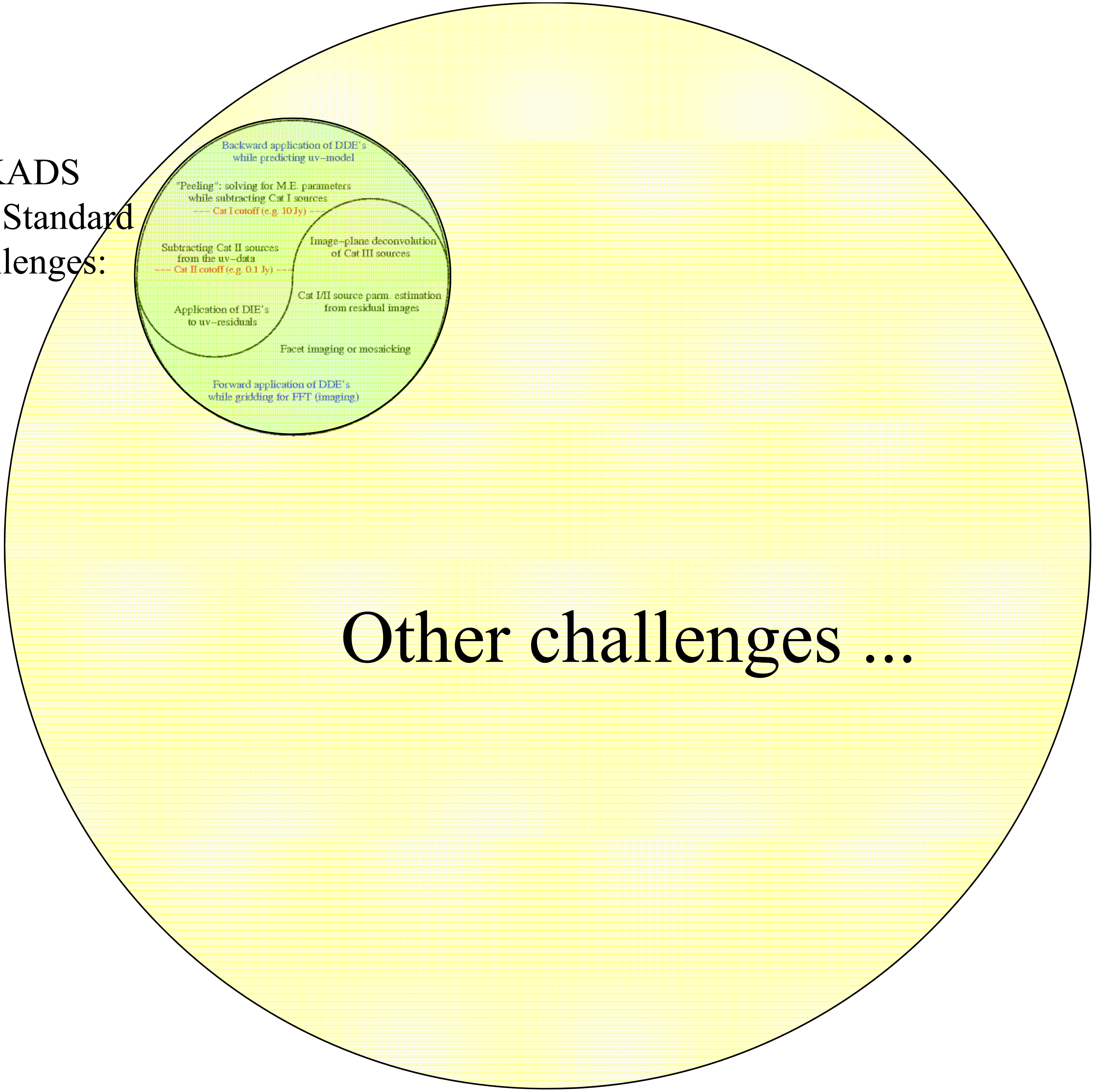
- **The myth**
  - Just get good coders – and let them have at it
- **The reality?**
  - Yes, it is “not impossible” that locking coders in a room with a gap under the door will eventually result in on-time in-budget delivery that meets all expectations
  - However:
    - The attendant risks are high
    - Software has become central to large science projects
    - Software projects in the public domain can be subject to embarrassing scrutiny

Public domain – four pathological enthusiasms:

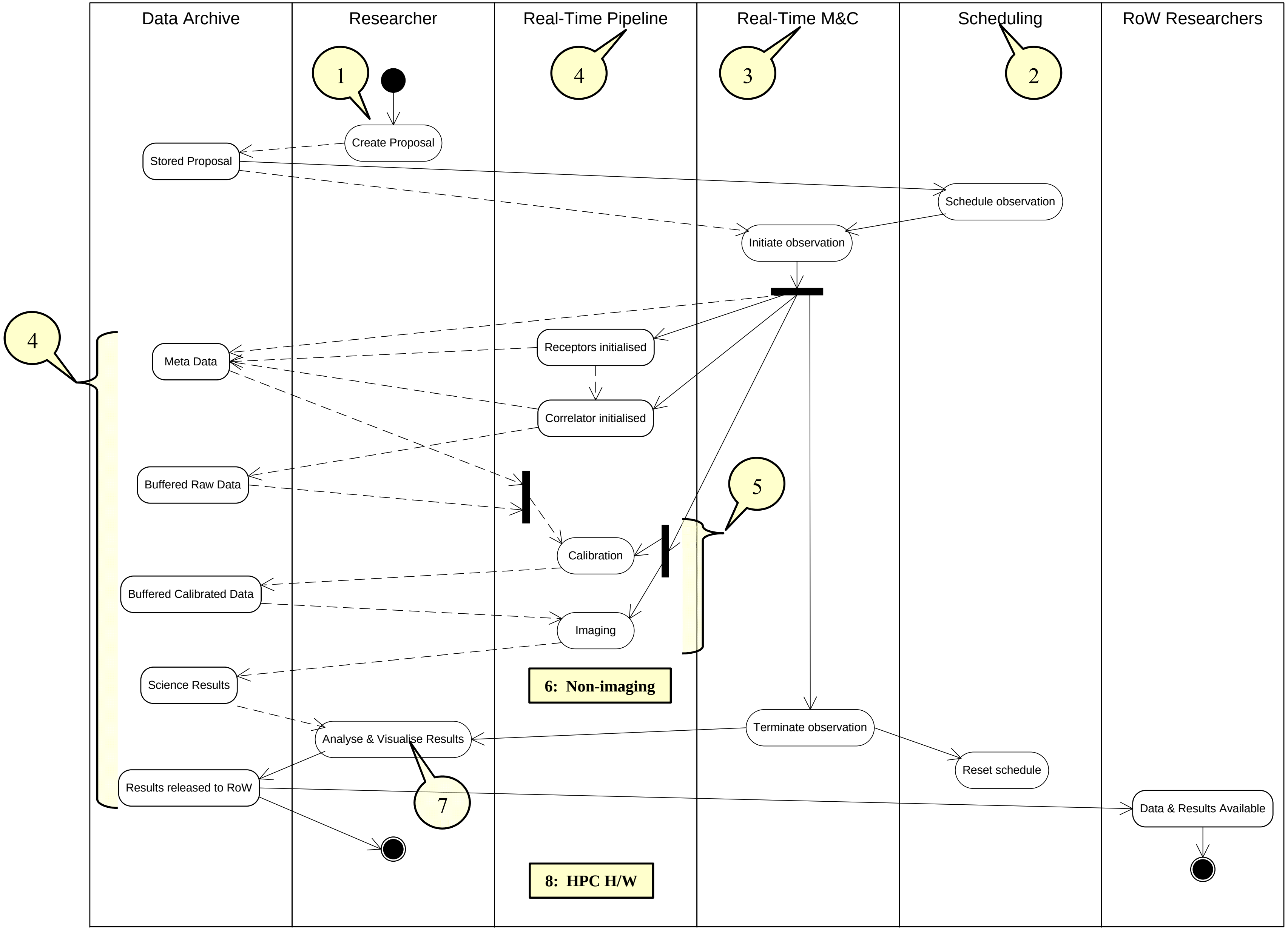
1. Idolisation – technological infatuation
2. Technophilia – the “myth of the technological fix”
3. Lomanism – enthusiasm induced by overzealous sales tactics, as epitomised by Willie Loman in Arthur Miller’s Death of a Salesman
4. Faddism – the tendency to link software development to the latest fad, such as “XP” or “XML” or management theory X, or Y, or Z



SKADS
Set of Standard
Challenges:



A structure for SKA software and computing:



Summary

Current SKA plans are “challenging”:

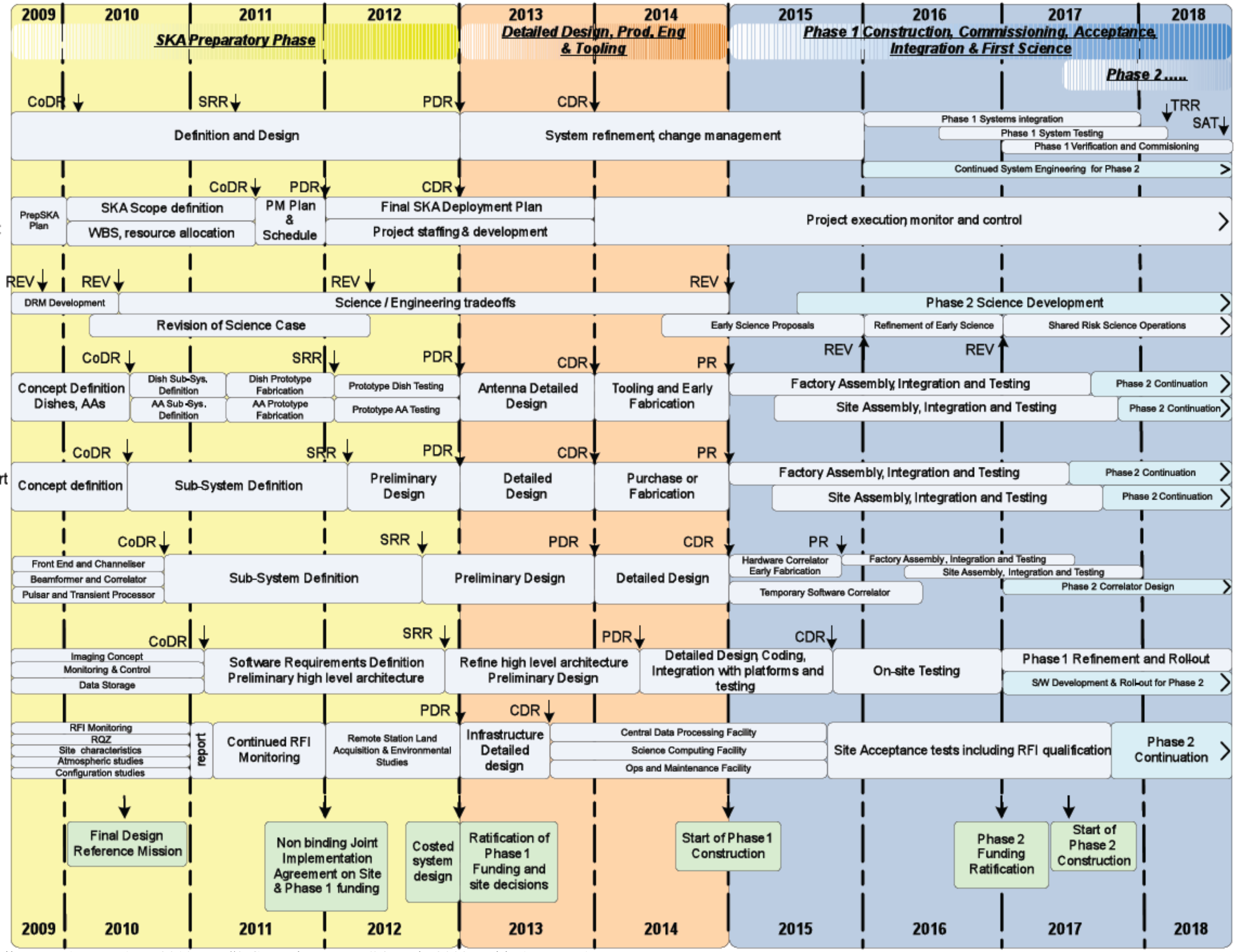
- Data rates will push computing requirements into the ExaScale (10^{18}) regime, with hardware costs order €100+ Million for each generation of box
- Computation is likely to consume Megawatts, costing €10s Millions annually
- The data pipeline is likely to require novel HPRC architectures and software
- Data store and archive management will require significant effort
- Software estimation benefits from formal models and expert judgement
- Software estimation is contingent on defining requirements
- “Designing to cost” requires prioritisation of requirements
- Significant uncertainties are inherent in estimating large scale software
- Reliable production software requires order of10x more effort than software developed for proofs of concept
- It is likely that the software will require large scale internationally collaborative development: order ~1,000+ staff years, even with reuse of extant codes and OTS

What is the future direction for SKA?

Phased construction

- Phase 1:
  - 2013 – 2018 construction
  - 10-20% of the collecting area
- Phase 2:
  - 2018 – 2022 construction
  - Full array at low and mid frequencies
- Phase 3:
  - 2023+ construction
  - High frequencies

Schedule to 2018



J. Cordes "The Square Kilometre Array – Astro2010 RFI #2 Ground Response" 27 July 2009; Table 1, page 51

- Some background – who am I?

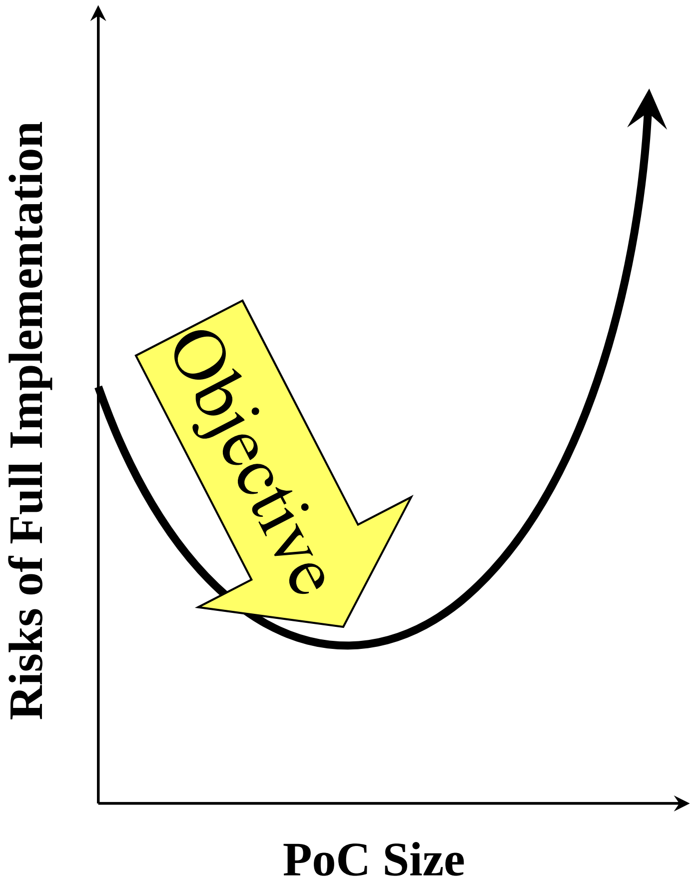
- What is this Creative Commons <u>not</u> about?

- What are (some of) the challenges facing SKA?

- How will this workshop change the world?

"Creative Commons" are good for developing "Proofs of Concept" for core algorithms

- Focus on areas of uncertainty:
  - We want to learn something
- Focus on what we want to learn about:
  - Scope, objectives, issues, assumptions, risks and deliverables
- Size the effort between "learning" and "doing" to minimise risks to the entire community



A couple of measurable objectives:
Do better than the giants of the past

Peak data rate	25 MB/s
Data for Peak 8-hr observation	700GB
flops per float	100 - 10000
Peak compute rate	5Tflop
Average/Peak computing load	0.1
Average compute rate	0.5Tflop
Turnaround for 8-hr peak observation	40 minutes
Average/Peak data volume	0.1
Data for Average 8-hr observation	70GB
Data for Average 1-yr	80TB

Table I: Typical and peak data and computing rates for the EVLA

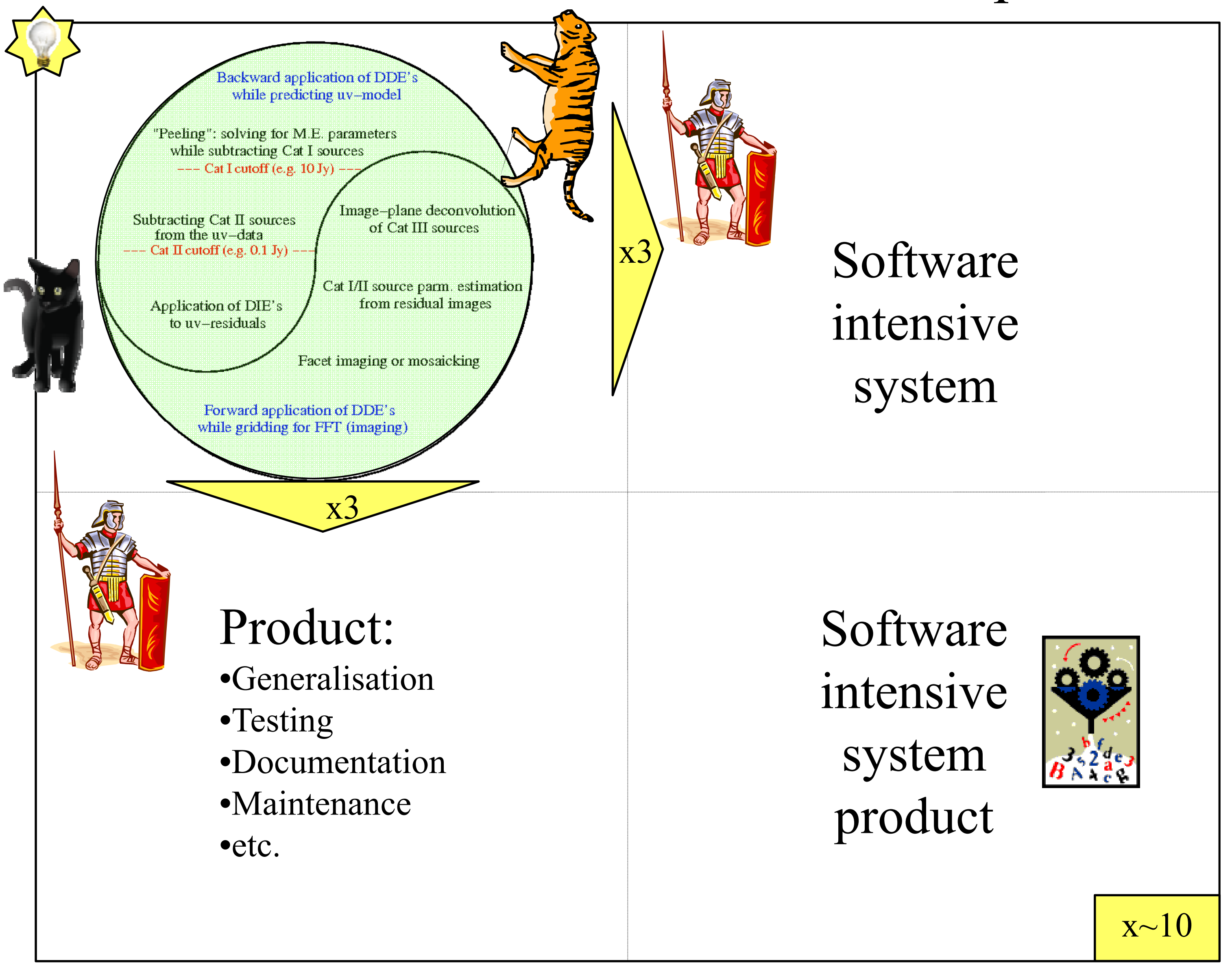
T. Cornwell "EVLA Memo 24: Computing for EVLA Calibration and Imaging", 2001 January12

More objectives: achieve 10^7 dynamic range, with:

- Automatic flagging
- Automatic termination at 10^7 dynamic range
- “Reasonable” use of core memory and disk storage
- On real data
- With real noise
- In real time
- To meet end users’ requirements

Another – not so measurable – objective:
co-operate, compete and share using a common framework – e.g. Tree Definition Language, TDL

SKA's core algorithms most likely will be based on these kinds of Creative Commons' developments ...



Fred Brooks; "The Mythical Man-Month – Essays on Software Engineering Anniversary Edition": 1995

Summary

http://en.wikipedia.org/wiki/File:Pogo_-_Earth_Day_1971_poster.jpg