



The Opportunities and Challenges for Computational Science and Engineering

Douglass Post

DoD HPCMP Chief Scientist and CREATE Project Manager

post@hpcmo.hpc.mil

**Inauguration of FZ-KFA Virtual Institute for High
Productivity Supercomputing**

July 4, Jülich, Germany



-
- **Promise**
 - **HCPMP**
 - **Code characterization**
 - **Challenges**
 - **Solutions**
 - **CREATE - 3 new projects testing the solutions**
 - **Tools part of solution**
 - **Conclusions and recommendations**



- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**

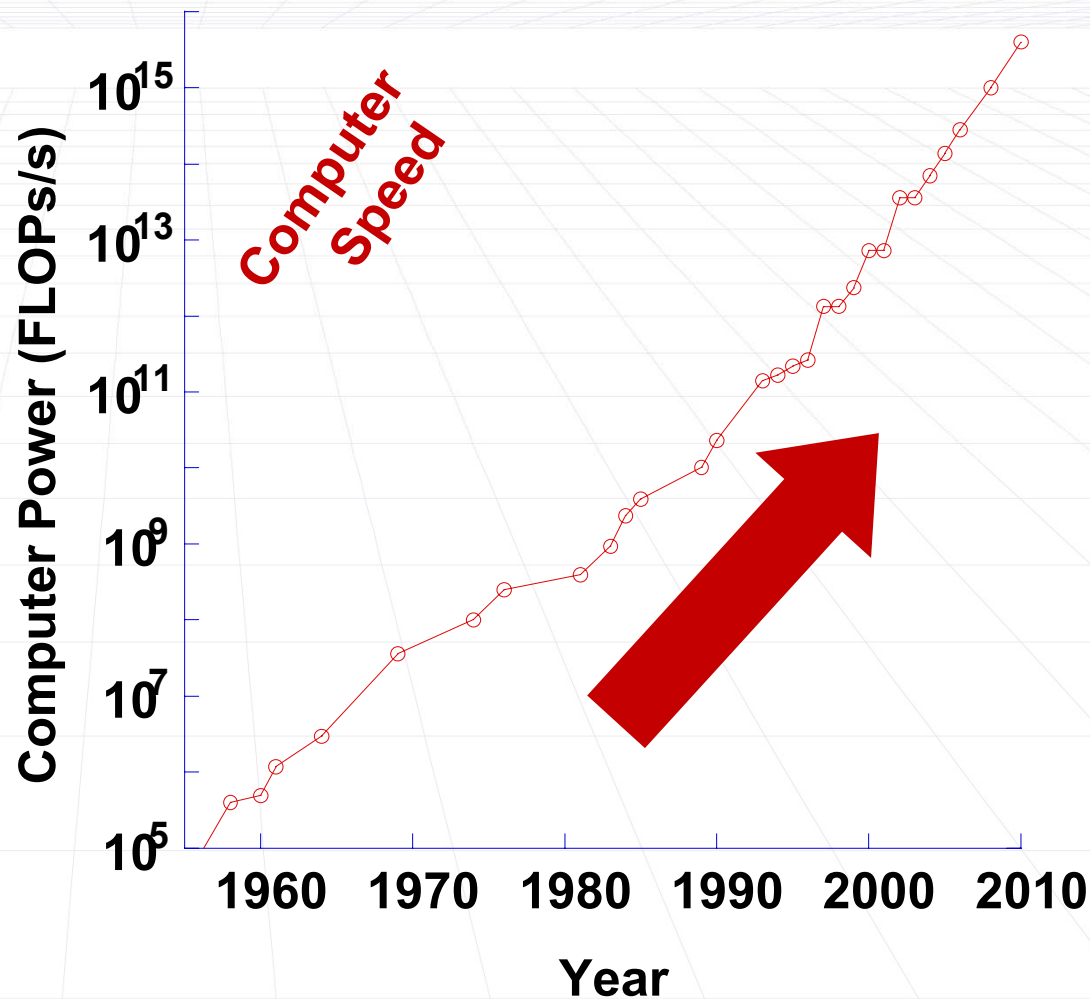


Exponential Growth In Computer Power and Codes Is Enabling Computational Science and Engineering to Be a “Disruptive” Technology.



Enable paradigm shift

- Potential to change the way problems are addressed and solved
- Make reliable predictions about the future
- Superior engineering & manufacturing
- Enable research to make new discoveries
- A vastly more powerful solving methodology for society!



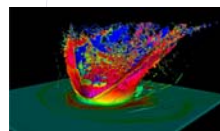
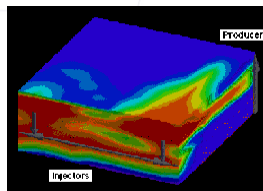
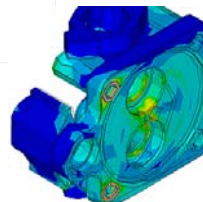
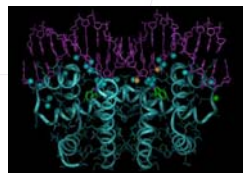
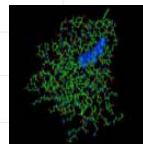
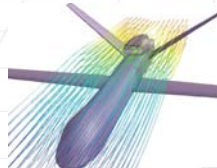
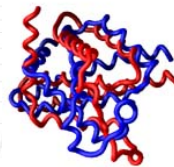
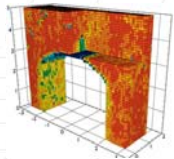
Computer power comes at the expense of complexity!



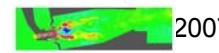
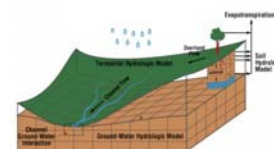
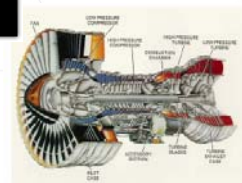
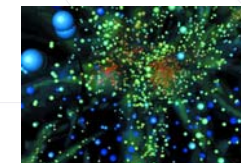
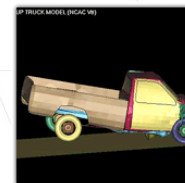
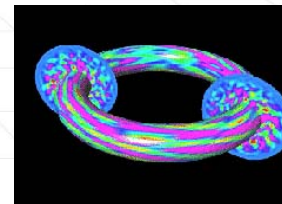
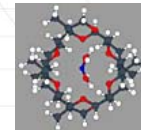
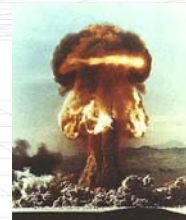
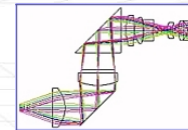
Computational Science And Engineering Is Becoming an Important Tool In Science And Engineering



Accelerator Design
Aircraft Design
Archaeology
Armor Design
Astrophysics
Atomic And Molecular Physics
Automobile Design
Bioengineering And Biophysics
Bioinformatics
Chemistry
Civil Engineering
Climate Prediction
Computational Biology
Computational Fluid Dynamics
Cosmology
Cryptography
Data Mining
Drug discovery
Earthquakes
Economics
Engineering Design And Analysis
Finance
Fluid Mechanics
Forces Modeling And Simulation
Fracture Analysis
General Relativity Theory
Genetics
Geophysics

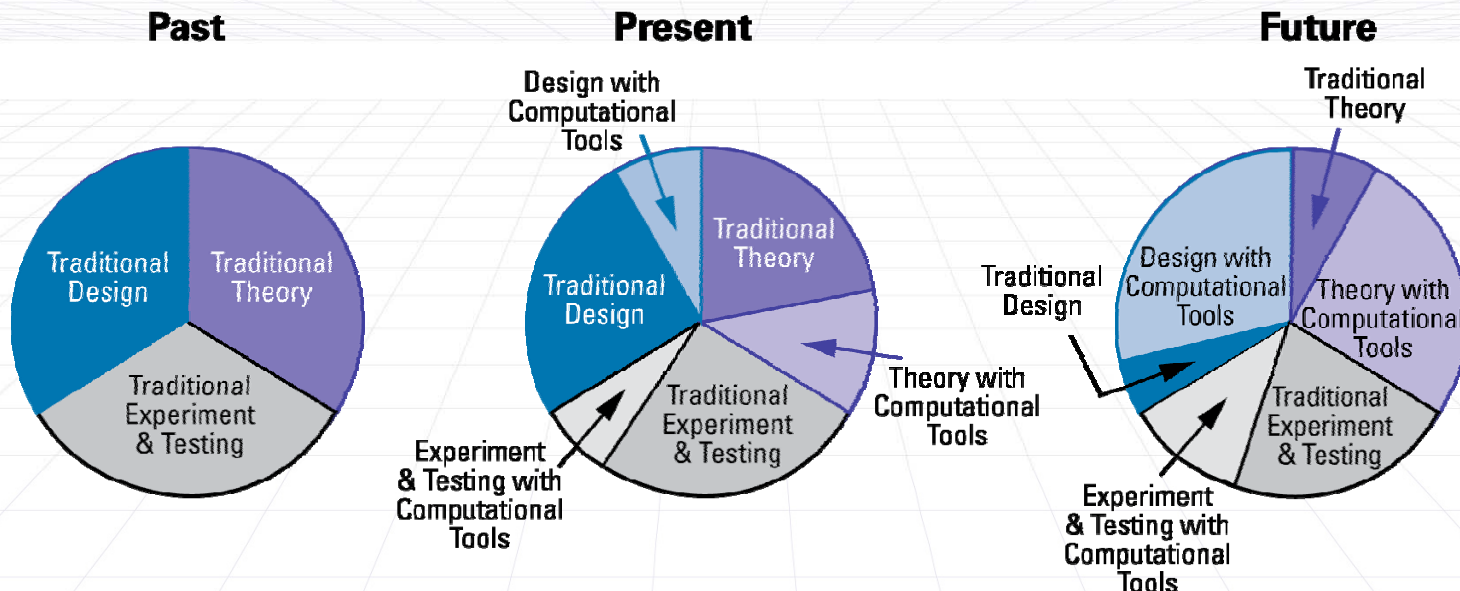


Groundwater And Contaminant Flow
High Energy Physics Research
Hydrology
Image Processing
Inertial Confinement Fusion
Integrated Circuit Chip Design
Magnetic Fusion Energy
Manufacturing
Materials Science
Medicine
Microtomography
Nanotechnology And Nanoscience
Nuclear Reactor Design And Safety
Nuclear Weapons
Ocean Systems
Petroleum Field Analysis And Prediction
Optics and Optical Design
Political Science
Protein Folding
Radar signature and antenna analysis
Radiation Damage
Satellite Image Processing
Scientific Databases
Search Engines
Shock Hydrodynamics
Signal Processing
Space Weather
Volcanoes
Weather Prediction
Wild Fire Analysis





Computational Tools Are Becoming Widely Used In Science And Engineering



	Past	Present	Future
Theory	Pencils, paper; slide rules	New: symbolic math; computational solutions	New: Almost all computational
Experiments	Physical hardware; notebooks; chart recorders; polaroid film, ...	New: computerized data collection & analysis; little V&V of computations; simple simulations & experimental design	New: Extensive V&V of computations; simulations are a part of experimental methodology
Eng. Design	Pencils, paper; slide rules	New: CAD-CAM; computational design analysis	New: Computational design & optimization



Computational Science And Engineering (CSE) Is A Uniquely Powerful Tool For Studying The Interaction Of Many Different Natural Effects



Science-based: Laws of nature govern individual interactions

- 1. Scientific discovery**
- 2. Experimental analysis and design**
- 3. Prediction of operational conditions**
- 4. Engineering design and analysis**

Heuristic-based: laws governing individual interactions are heuristic and/or empirical

- 5. Data collection, analysis & mining - Statistics.....**
 - Social sciences, medicine, education, research**
- 6. Heuristic simulations and decision tools (economic forecasts, war and strategy simulations,..)**
- 7. Cellular automata**



Computational Science is hard. Computational Engineering is harder.



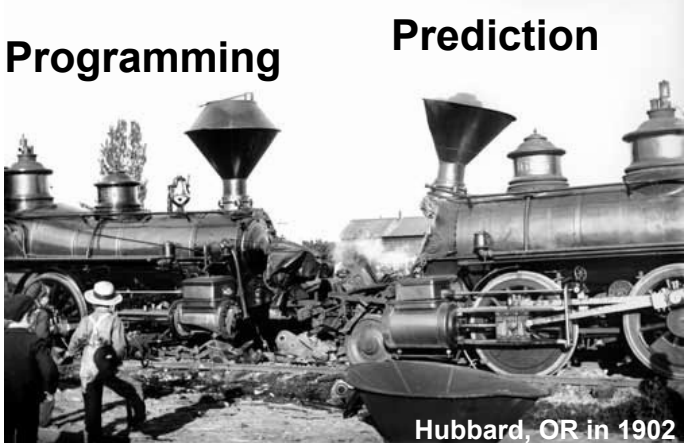
- Computational Science is challenging:
 - Develop a complex code for complex computers, apply it to study a scientific research problem, and publish the findings.
- Computational Engineering is even more challenging:
 - Develop sets of computational engineering tools that will be used by other engineering groups to design and analyze systems that will be built by others, tested by others and employed in operations by others.
- Additional challenges due to the use of non-collocated, multi-institutional teams.



Three Challenges

Performance, Programming and Prediction



- 1. Performance Challenge - Computers power increasing through growing complexity**
 - Massive parallelization, multi-core & heterogeneous (CELL, FPGA, GPU...) processors, complex memory hierarchies.....
 - 2. Programming Challenge -Programming for Complex Computers**
 - Rapid code development of codes with good performance
 - 3. Prediction Challenge —Developing **predictive** codes with complex scientific models**
 - Develop accurate predictive codes
 - Verification
 - Validation
 - Code Project Management
 - Train wreck coming between the last two
- 

Programming Prediction

Hubbard, OR in 1902
- Better software development and production tools are desperately needed for us to take full advantage of computers



How can we develop the computational capability we need?



We need to develop a complete problem solving system:

- 1. Computers**
- 2. Software—both applications and tools**
- 3. V&V**
- 4. Users**
- 5. Sponsors**



Next Generation Computers Offer Society Unparalleled Power to Address Important Problems



- Next generation computers will provide exciting opportunities to develop and deploy very powerful application codes, much more powerful than present tools:
 - Utilize accurate solution methods
 - Include all the effects we know to be important
 - Model a complete system
 - Complete parameter surveys in hours rather than days to weeks to months
- Greatest opportunities include large-scale codes that integrate many multi-scale effects to model a complete system
- **Developing such codes is the major bottleneck!**
 - Requires large (10 to 30 professionals), multi-disciplinary, multi-institutional teams 5 to 10 years
 - Codes must to scale to many, many thousands of processors
- How do we position ourselves to take advantage of the opportunity that the next generation of computers will offer?



- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**



HPC Centers



Defense Research & Engineering Network



Institutes/Portfolios



DHPIs





HPCMP is the second largest collection of supercomputers in the KNOWN world!



- HPCMP Modernizes DoD computing with \$50M annual purchases.

MSRC Systems				No. of Avail PEs	Peak GFLOPS of Actual PEs	Equiv. of 1,024-PE HABU
	Number of Actual PEs	HABU Rating per 1,024 PEs	Memory (GB)			
ERDC	SGI Origin 3900	1,024	3.08	1,008	1,434	3.08
	Cray XT3 (Upgrade)	8,320	11.54	8,192	43,264	93.76
	Cray Hood	8,848	10.39	8,608	40,701	89.76
NAVO	IBM Regatta P4	2,944	6.55	2,832	20,019	18.83
	IBM Cluster 1600 P5	2,976	12.31	2,816	20,237	35.78
	IBM Cluster 1600 P5	1,504	13.66	1,408	10,227	20.06
	IBM Regatta P4	1,408	2.10	1,328	7,322	2.89
	IBM Regatta P4	512	6.55	464	3,482	3.28
ARL	SGI Altix Cluster (D)	256	8.68	256	1,536	2.17
	IBM Opteron Cluster	2,372	4.73	2,304	10,437	10.96
	Linux Networkx Xeon Cluster	2,100	5.80	2,048	12,852	11.89
	Linux Networkx Woodcrest Cluster	4,286	16.07	4,160	51,432	67.26
	Linux Networkx Dempsey Cluster	3,360	10.86	3,336	21,504	35.63
	Linux Networkx Cluster	256	5.21	256	1,567	1.30
ASC	IBM Regatta P4 (D)	32	2.55	32	166	0.08
	SGI Origin 3900	2,048	3.08	2,032	2,867	6.16
	SGI Origin 3900 (D)	128	1.90	128	179	0.24
	HP Opteron Cluster	2,048	6.71	2,048	10,650	13.42
	SGI Altix Cluster	2,048	6.84	2,000	12,288	13.68
	SGI Altix 4700 (Density)	256	12.02	250	1,638	3.00
	SGI Altix 4700 (8192 2GB Density, 1024 4GB Memory)	9,216	12.02	9,000	58,982	108.14
MSRC Totals				54,506	332,784	541.4

12/2007

7/6/2007



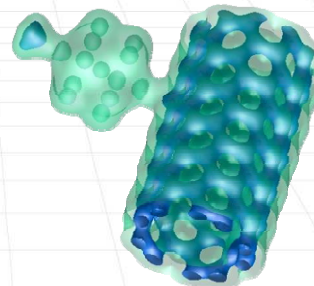
- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Multi-scale issues**
- **Tools part of solution**
- **Conclusions and recommendations**

What Kind Of Codes Are We Talking About?

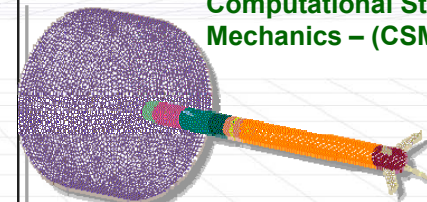
We surveyed our Large, Diverse DoD HPC Community to characterize our codes

- 587 projects and 2,262 users at approximately 144 sites
- Requirements categorized in 10 Computational Technology Areas (CTA)
- DoD HPCMP has about 20 computers with ~240 TFlops/s peak (circa 2006)

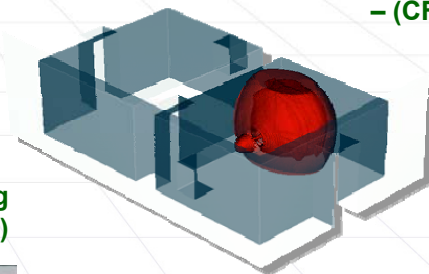
Electronics, Networking, and Systems/C4I – (ENS)



Computational Structural Mechanics – (CSM)

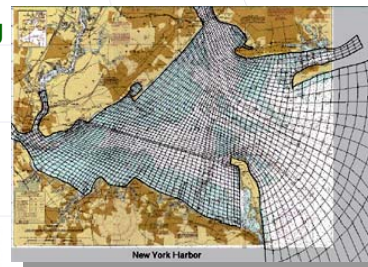


Computational Fluid Dynamics – (CFD)

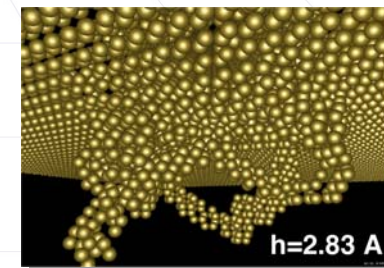


Environmental Quality Modeling & Simulation – (EQM)

Climate/Weather/Ocean Modeling & Simulation – (CWO)

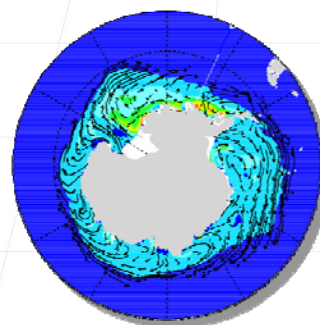
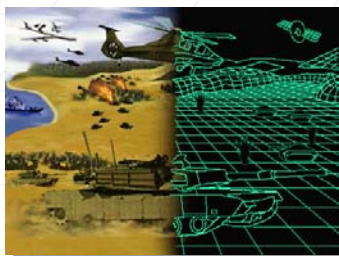


Computational Chemistry, Biology & Materials Science – (CCM)

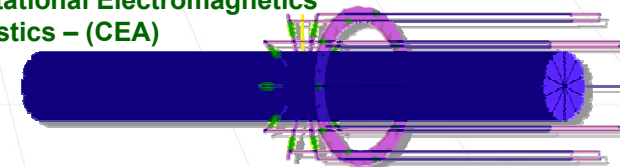


$h=2.83 \text{ \AA}$

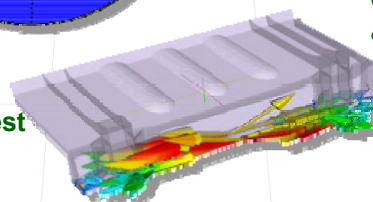
Forces Modeling & Simulation – (FMS)



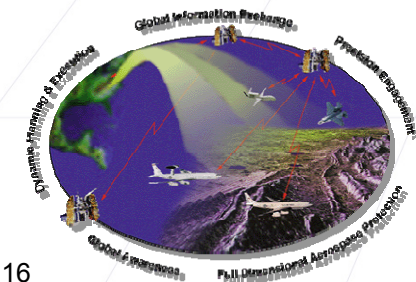
Computational Electromagnetics & Acoustics – (CEA)



Integrated Modeling & Test Environments – (IMT)



Signal/Image Processing – (SIP)





We sent surveys to our top 40 codes (ordered by time requested), with 15 responses so far.



Application Code	Hours	Application Code	Hours
CTH	93,435,421	DMOL	5,200,100
HYCOM	89,005,100	ICEM	4,950,000
GAUSSIAN	49,256,850	CFD++	5,719,000
ALLEGRA	32,815,000	ADCIRC	4,100,750
ICEPIC	26,500,000	MATLAB	4,578,430
CAML	21,000,000	NCOM	5,080,000
ANSYS	17,898,520	Loci-Chem	5,500,000
VASP	18,437,500	GAMESS	5,142,250
Xflow	15,165,000	STRIPE	4,700,000
ZAPOTEC	12,125,857	USM3D	4,210,000
XPATCH	23,462,500	FLUENT	3,955,610
MUVES	10,974,120	GASP	4,691,000
MOM	18,540,000	Our DNS code (DNSBLB)	2,420,000
OVERFLOW	8,835,500	ParaDis	4,000,000
COBALT	14,165,750	FLAPW	4,050,000
Various	8,125,000	AMBER	4,466,000
ETA	11,700,000	POP	3,800,000
CPMD	5,975,000	MS-GC	3,500,000
ALE3D	5,864,500	TURBO	3,600,600
PRONTO	5,169,100	Freericks Solver	2,600,000



Characteristics Aren't Surprising.



	Team size FTEs	# users	Total SLOC(k)	SLOC Fortran 77 (k)	SLOC Fortran 90, 95 (k)	SLOC C (k)	SLOC C++ (k)	other
Mean	38	5,038	820	24%	34%	17%	13%	13%
Median	6	27	275					

- Even now, codes are developed by teams
- Most codes have more users than just the development team
- Codes are big
- 58% of the codes are written in Fortran.
- New languages with higher levels of abstraction are attractive, but they will have to be compatible and interoperable with Fortran with MPI.



Further Data Isn't Surprising Either.

	Total Age (yr)	age production version	total number of different platforms	Largest Degree of Parallelism	Typical minimum # of processors	Typical Maximum # of processors	Is memory a limitation ?	Memory processor GBytes /proc
Mean	19.8	15.1	6.9	1000 to 3000	225	292	Sometimes	0.75-4
Median	17.5	15.5	7.0	1000 to 3000	128	128		

- Most codes are at least 15 years old
- Most codes run on at least 7 different platforms
- Most codes can run on ~1000 processors, but don't
- Most users want at least 1 GByte / processor of memory.



HPCMP Acquisition Application Benchmark Codes Perform Differently On Different Platforms.

- **Aero – Aeroelasticity CFD code**
(Fortran, serial vector, 15,000 lines of code)
- **AVUS (Cobalt-60) – Turbulent flow CFD code**
(Fortran, MPI, 19,000 lines of code)
- **GAMESS – Quantum chemistry code**
(Fortran, MPI, 330,000 lines of code)
- **HYCOM – Ocean circulation modeling code**
(Fortran, MPI, 31,000 lines of code)
- **OOCore – Out-of-core solver**
(Fortran, MPI, 39,000 lines of code)
- **CTH – Shock physics code (SNL)**
(~43% Fortran/~57% C, MPI, 436,000 lines of code)
- **WRF – Multi-Agency mesoscale atmospheric model**
(Fortran and C, MPI, 100,000 lines of code)
- **Overflow-2 – CFD code originally developed by NASA**
(Fortran 90, MPI, 83,000 lines of code)



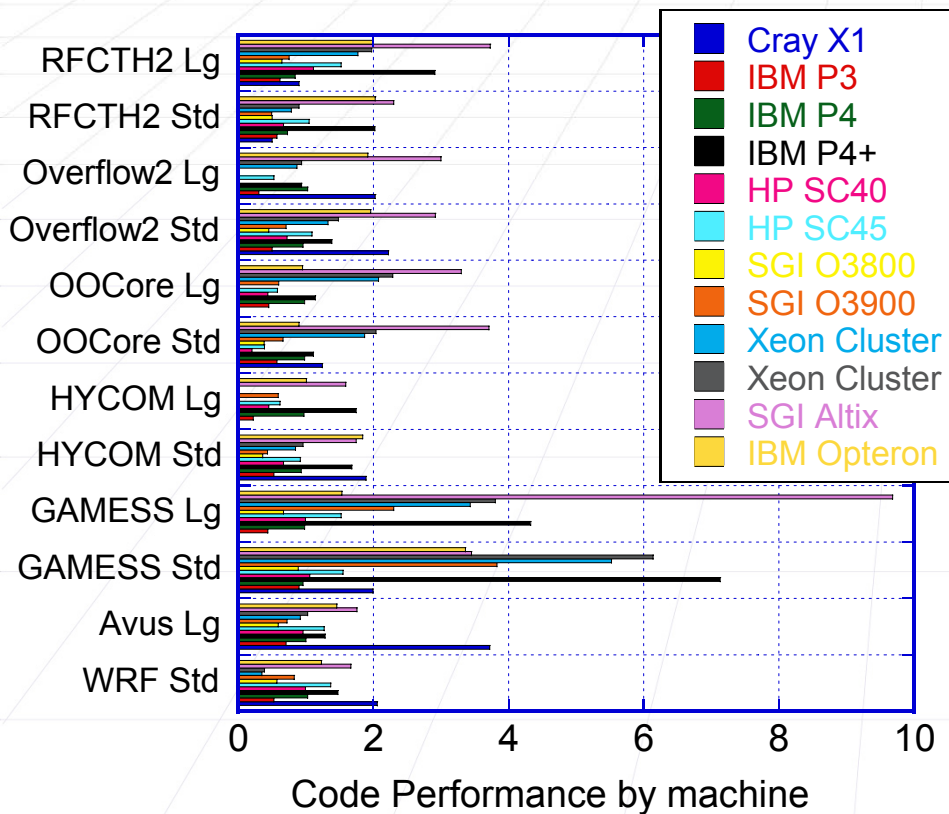
Performance Depends On The Computer And On The Code.



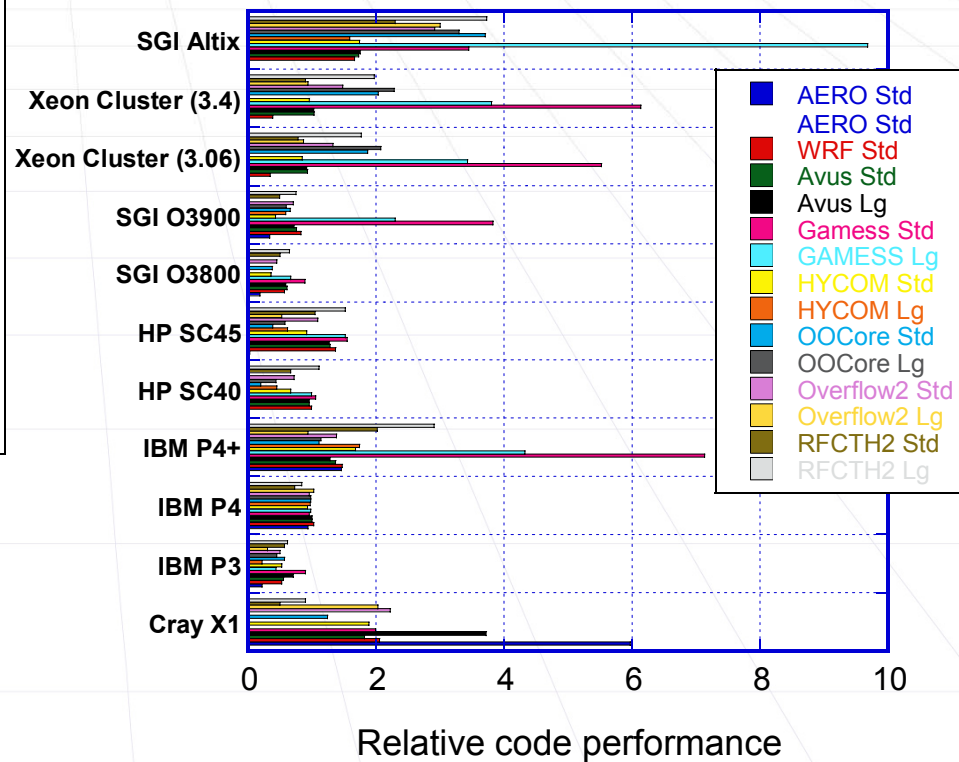
- Normalized Performance = 1 on the NAVO IBM SP3 (HABU) platform with 1024 processors (375 MHz Power3 CPUs) assuming that each system has 1024 processors.
- GAMESS had the most variation among platforms.

Substantial variation of codes for a single computer.

Code Performance (by machine)



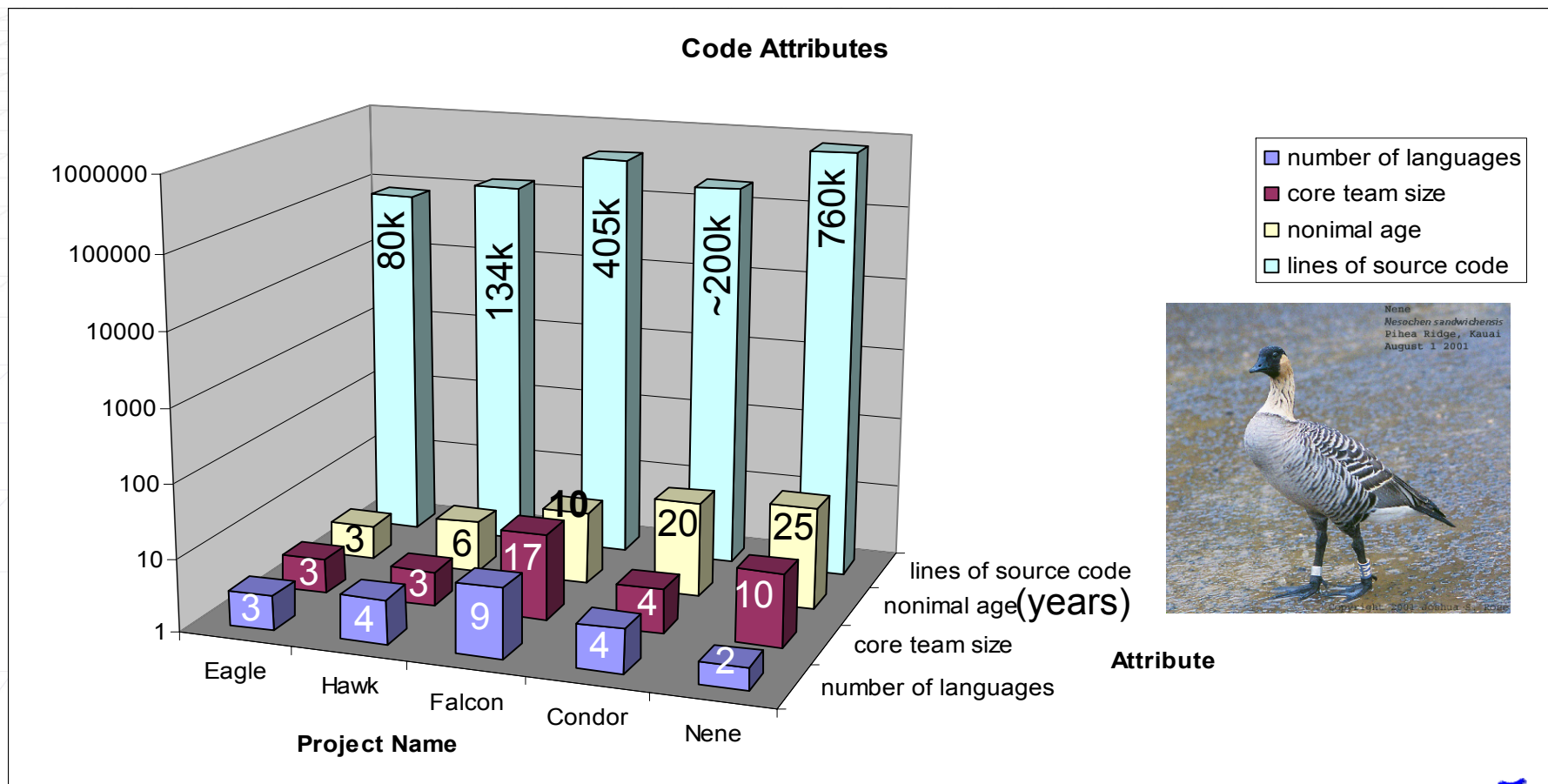
Code performance (grouped by machine)



—SC 2005 panel Tour de HPCylces



Also Did Detailed Case Studies Of First 6 Large US Federal Agency CSE Codes And Then Another Set Of 5 Large-scale CSE Codes



5 CSE codes (academia and lab)





Nine Cross-Study Observations



1. Once selected, the primary languages (typically Fortran) adopted by existing code teams do not change.
2. The use of higher level languages (e.g. Matlab) has not been widely adopted by existing code teams except for "bread-boarding" or algorithm development.
3. Code developers in existing code teams like the flexibility of UNIX command line environments.
4. Third party (externally developed) software and software development tools are viewed as a major risk factor by existing code teams.
5. The project goal is scientific discovery or engineering design. "Speed to solution" and "execution time" are not highly ranked goals for our existing code teams unless they directly impact the science.
6. All but one of the existing code teams we have studied have adopted an "agile" development approach.
7. For the most part, the developers of existing codes are scientists and engineers, not computer scientists or professional programmers.
8. Most of the effort has been expended in the "implementation" workflow step.
9. The success of all of the existing codes we have studied has depended most on keeping their customers (not always their sponsors) happy.



—R. Kendall, A. Mark, J. Carver, D. Post, et al





- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**

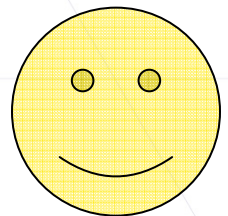
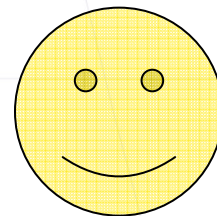
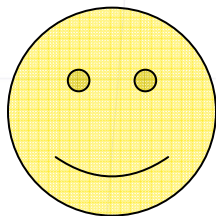


What do CSE Application People Need?



- To develop and run complex applications, they want and need:
 - Sufficient performance to enable large problems, fast turnaround, simple code development, accurate answers
 - Fast integer and floating point arithmetic (with fast divides)
 - Simple memory hierarchies that are fast, globally addressable, reliable, have low latencies and high bandwidth, and lots of data storage
 - Stable, long-lived and reliable platforms and architectures
 - Stable, long-lived and reliable software development and production tools that provide the needed capability and are simple and easy to use
 - Developers want something like a Unix/LINUX or Mac or even PC workstation development environment, or better

Summary: Users and developers want to solve their scientific or engineering problem and not worry about the architectural details of computers



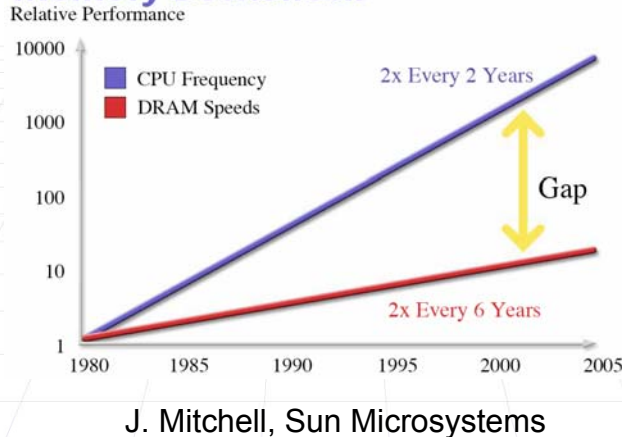


What Are They Getting?

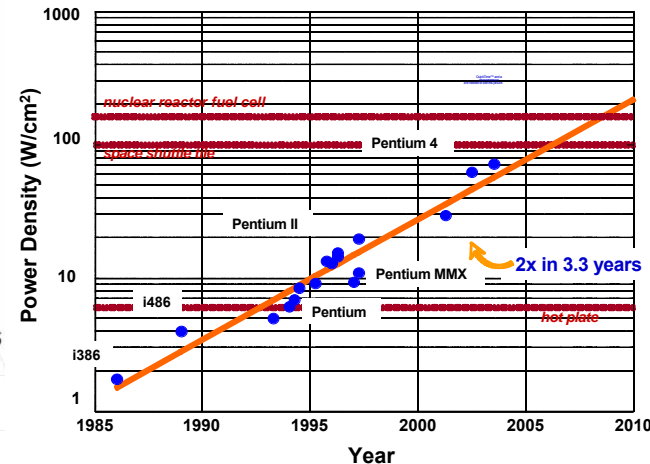


- Complex, distributed memory with only very slowly improving memory bandwidth
- Slowing rate of processor speed growth

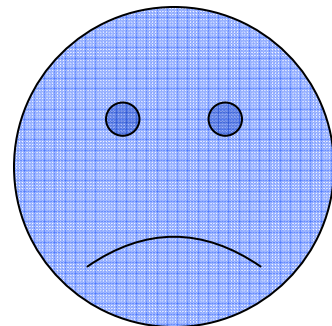
Memory Bottleneck



Growth in Power Density



- Distributed processor and memory systems linked together in ever larger and more complex networks plus heterogeneous, multi-core challenges (e.g. Blue Gene, Roadrunner,...)
- Rapid turnover in machines and machine architectures (2-4 years)
- Unreliable parallel file systems
- Unstable development and production environment
- Highly complex programming environment and challenges
 - Complex architectures—>Complex programming
 - Performance that is poor (a few % of peak) and hard to optimize
 - Frequent and challenging ports to new platforms
 - Inadequate and immature tools to develop and run codes





Computational Engineering Code Developer's World – Six Major Challenges and Risks



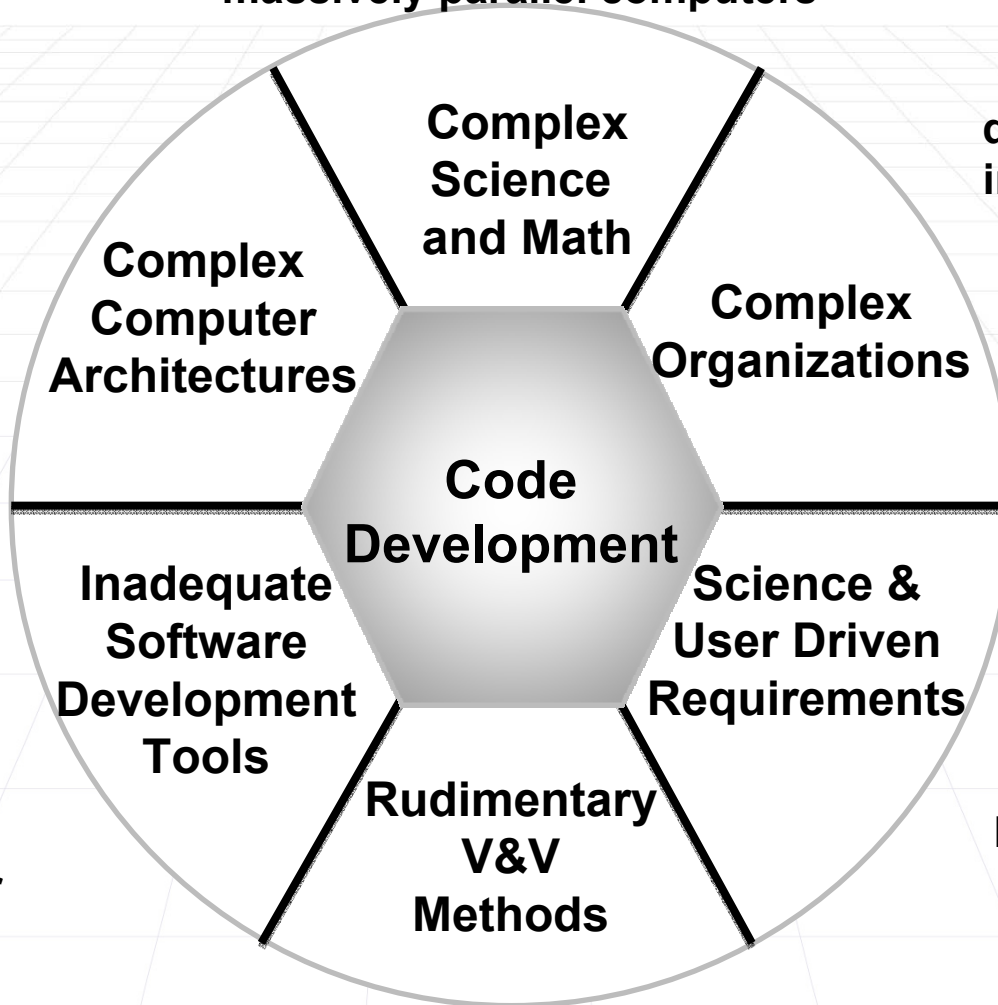
Many strongly coupled effects and massively parallel computers

Large, multi-disciplinary, multi-institutional teams



Laws of nature & user needs win every time

Immature methods and few validation experiments



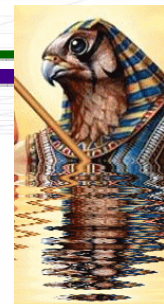
Zillions of processors linked with complicated and slow networks

Plus different kinds of processors: X86, Power-n, GPUs, FPGAs, vector, CELL.....

Little help for dealing with complex computer architectures



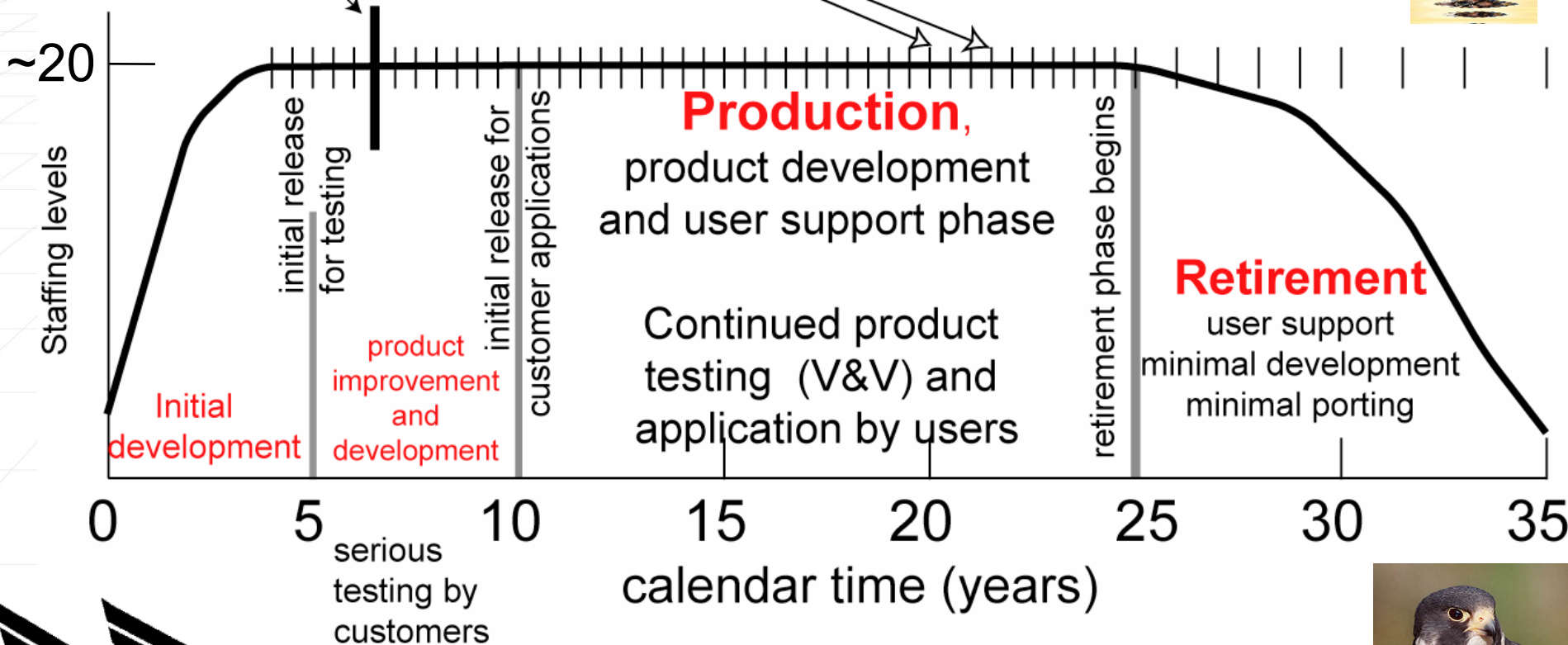
Developing a Large, Multi-scale, Multi-effect Code Takes a Large Team a Long Time



Falcon Project Life Cycle

2003

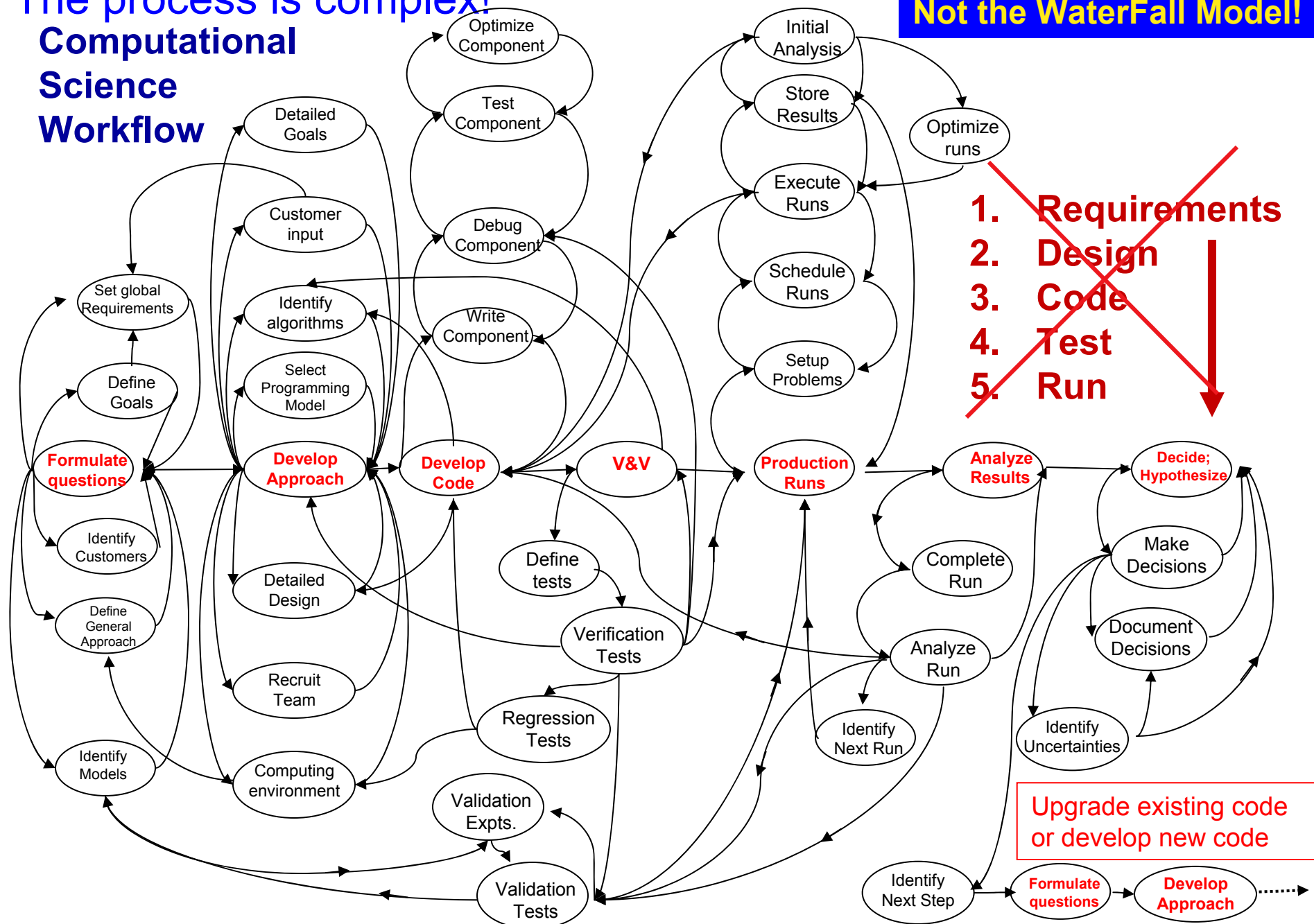
major product releases



The process is complex!

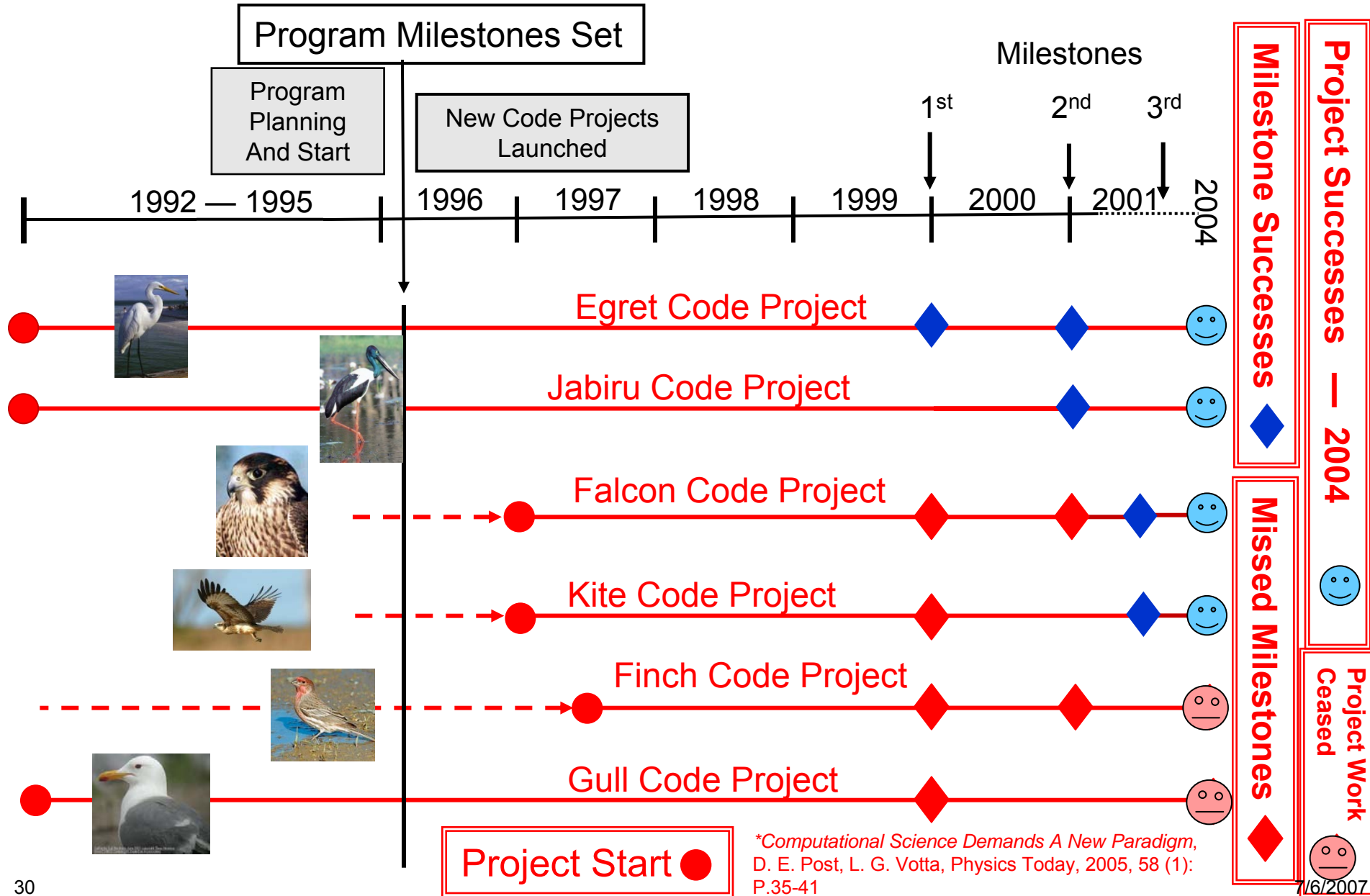
Computational Science Workflow

Not the WaterFall Model!



It's Risky!*

Code Project Schedule For Six Large-scale Physics Codes





- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**

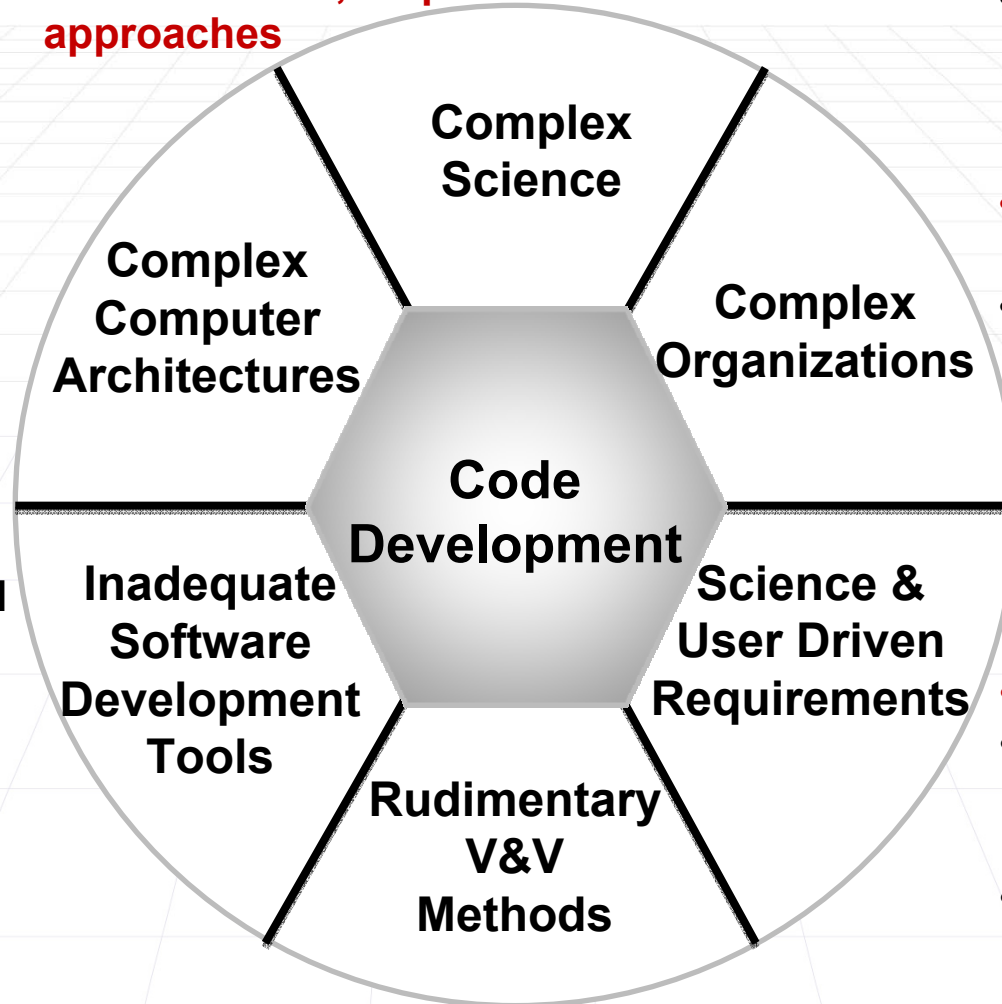


What can we do to minimize risks and address the challenges?



- Emphasize science and engineering
- **Minimize risks, emphasize conservative approaches**

- **Minimize risks**
- Conservative choices:
 - Computers
 - Algorithms
 - Parallel programming models



- Develop collaboration methods and technologies
- **Emphasize building teams**
- Emphasize flexible and agile software project management

- **Listen to users**
- Design code for engineering design analysis
- Establish early connection with engineers

- **Develop methods and connect to experimental community**

- Emphasize agile and flexible software engineering
- **Establish a community-wide program for tool development and deployment**
- **Support tool use by code developers**



Issues Summarized In January 2005 Physics Today Article*.



- Three Challenges
 - Performance Challenge
 - Programming Challenge
 - Prediction Challenge
 - Where case studies are important
- Case Studies are needed for success
 - The Scientific Method
- Paradigm shift needed
 - Computational Science moving from few effect codes developed by small teams to many effect codes developed by large teams
 - Similar to transition made by experimental science in 1930—1960
 - Software Project Management and V&V need more emphasis

Computational Science Demands a New Paradigm*, D.E. Post and L.G. Votta, *Physics Today*, **58(1), 2005, p.35-41.

Email post@ieee.org to get a copy.

Computational Science Demands a New Paradigm

The field has reached a threshold at which better organization becomes crucial. New methods of verifying and validating complex codes are mandatory if computational science is to fulfill its promise for science and society.

Douglass E. Post and Lawrence G. Votta

Computers have become indispensable to scientific research. They are essential for collecting and analyzing experimental data, and they have largely replaced pencil and paper as the theorist's main tool. Computers let theorists extend their studies of physical, chemical, and biological systems by solving difficult nonlinear problems in magnetohydrodynamics; atomic, molecular, and nuclear structure; fluid turbulence; shock hydrodynamics; and cosmological structure formation.

Beyond such well-established aids to theorists and experimenters, the exponential growth of computer power is now launching the new field of computational science. Multidisciplinary computational teams are beginning to develop large-scale predictive simulations of highly complex technical problems. Large-scale codes have been created to simulate, with unprecedented fidelity, phenomena such as supernova explosions (see figures 1 and 2), inertial-confinement fusion, nuclear explosions (see the box on page 38), asteroid impacts (figure 3), and the effect of space weather on Earth's magnetosphere (figure 4).

Computational simulation has the potential to join theory and experiment as a third powerful research methodology. Although, as figures 1–4 show, the new discipline is already yielding important and exciting results, it is also becoming all too clear that much of computational science is still troublingly immature. We point out three distinct challenges that computational science must meet if it is to fulfill its potential and take its place as a fully mature partner of theory and experiment:

- ▶ the performance challenge—producing high-performance computers,
- ▶ the programming challenge—programming for complex computers, and
- ▶ the prediction challenge—developing truly predictive complex application codes.

The performance challenge requires that the exponential growth of computer performance continue, yielding ever larger memories and faster processing. The programming challenge involves the writing of codes that can

efficiently exploit the capacities of the increasingly complex computers. The prediction challenge is to use all that computing power to provide answers reliable enough to form the basis for important decisions.

The performance challenge is being met, at least for the next 10 years. Processor speed continues to increase, and massive parallelization is augmenting that speed, albeit at the cost of increasingly complex computer architectures. Massively parallel computers with thousands of processors are becoming widely available at relatively low cost, and larger ones are being developed.

Much remains to be done to meet the programming challenge. But computer scientists are beginning to develop languages and software tools to facilitate programming for massively parallel computers.

The most urgent challenge

The prediction challenge is now the most serious limiting factor for computational science. The field is in transition from modest codes developed by small teams to much more complex programs, developed over many years by large teams, that incorporate many strongly coupled effects spanning wide ranges of spatial and temporal scales. The prediction challenge is due to the complexity of the newer codes, and the problem of integrating the efforts of large teams. This often results in codes that are not sufficiently reliable and credible to be the basis of important decisions facing society. The growth of code size and complexity, and its attendant problems, bears some resemblance to the transition from small to large scale by experimental physics in the decades after World War II.

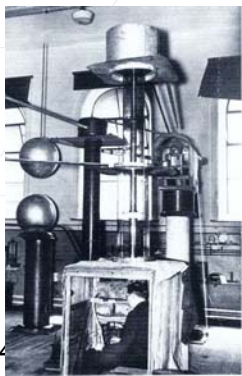
A comparative case study of six large-scale scientific code projects, by Richard Kendall and one of us (Post),¹ has yielded three important lessons. Verification, validation, and quality management, we found, are all crucial to the success of a large-scale code-writing project. Although some computational science projects—those illustrated by figures 1–4, for example—stress all three requirements, many other current and planned projects give them insufficient attention. In the absence of any one of those requirements, one doesn't have the assurance of independent assessment, confirmation, and repeatability of results. Because it's impossible to judge the validity of such results, they often have little credibility and no impact.

Part of the problem is simply that it's hard to decide whether a code result is right or wrong. Our experience as referees and editors tells us that the peer review process in computational science generally doesn't provide as effective a filter as it does for experiment or theory. Many things that a referee cannot detect could be wrong with a computational-science paper. The code could have hidden defects, it might be applying algorithms improperly, or its spatial or temporal resolution might be inappropriately coarse.

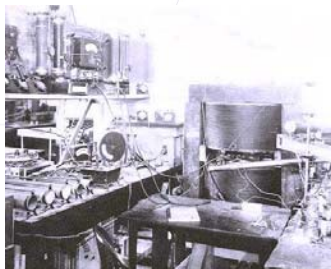
Douglass Post is a computational physicist at Los Alamos National Laboratory and an associate editor-in chief of *Computing in Science and Engineering*. Lawrence Votta is a Distinguished Engineer at Sun Microsystems Inc in Menlo Park, California. He has been an associate editor of *IEEE Transactions on Software Engineering*.

Computational Science And Engineering Is Making The Same Transition That Experimental Science Made In 1930 Through 1960.

- Computational Science and Engineering moving from “few-effect” codes developed by small teams (1 to 3 scientists) to “many-effect” codes developed by larger teams (10, 20 or more).
- Analogous experimental science transition made in 1930-1960 time frame
- Small-scale science experiments involving a few scientists in small laboratories —> “big science” experiments with large teams working on very large facilities.
- “Big Science” experiments require greater attention to formality of processes, project management issues, and coordination of team activities than small-scale science.
- Experimentalists were better equipped than most computational scientists to make the transition and they had more time to make the transition.
 - Small scale experiments require much more interaction with the outside world than small-scale code development.
 - Experimentalists had ~20 years, while computational scientists are doing the transition much more quickly.



Early 1930's



Late 1930's





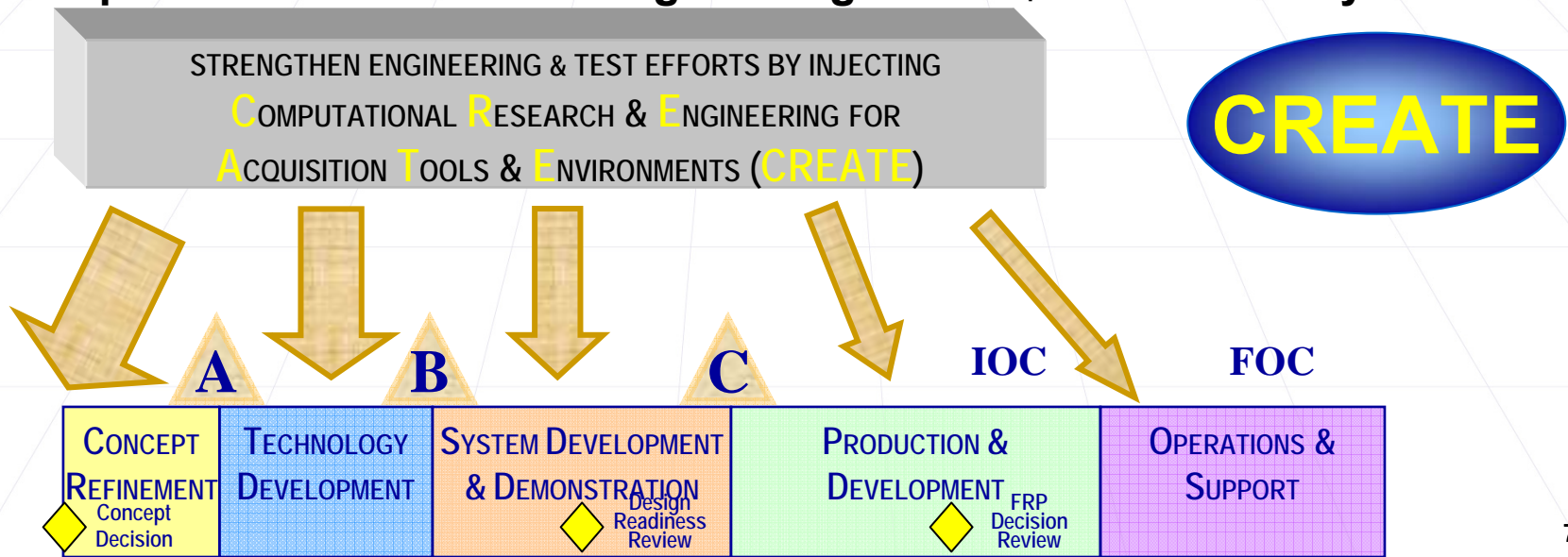
- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**



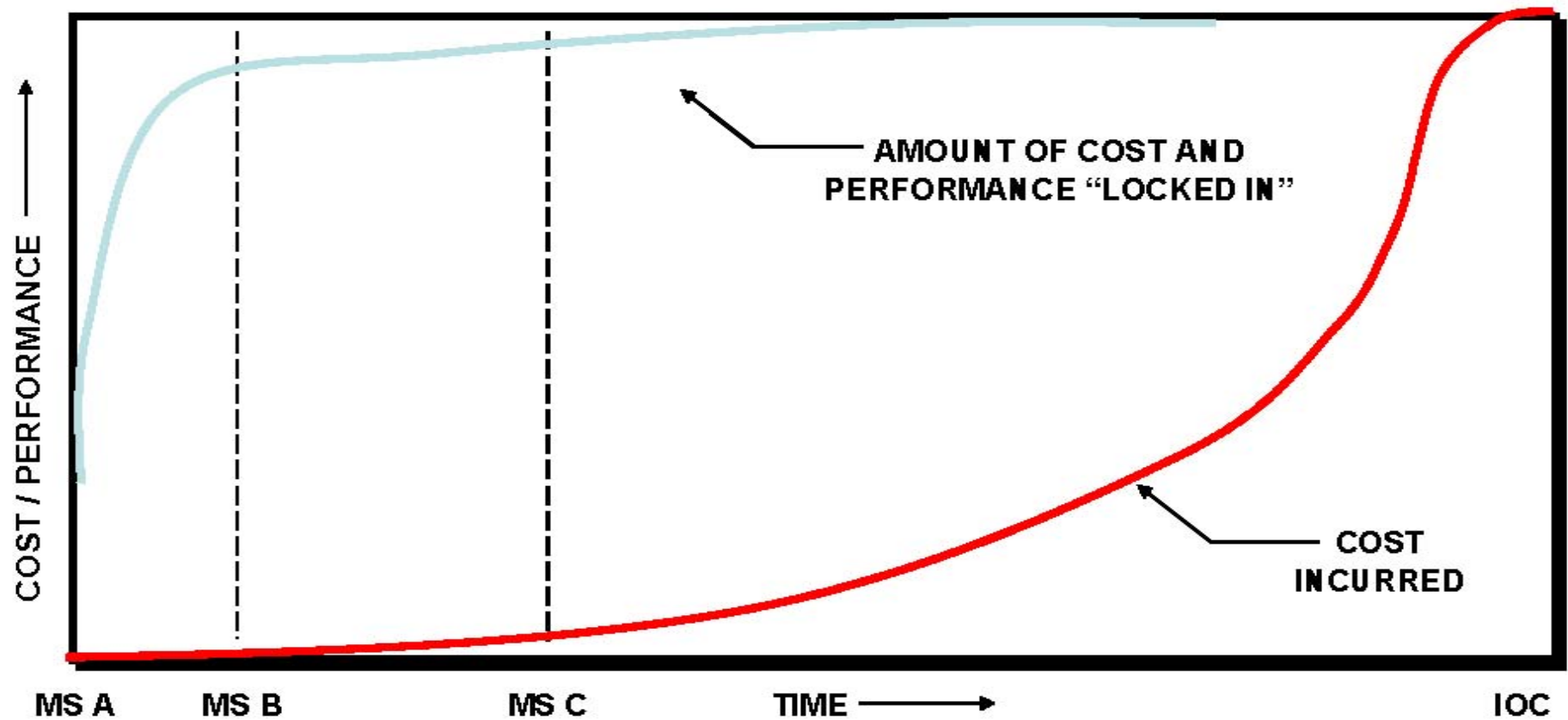
CREATE Goal



- **CREATE goal is to enable major improvements in the DoD Acquisition Process**
 - Detect and fix design flaws early in the design process before major schedule and budget commitments are made
 - Begin system integration earlier in acquisition process
 - Increase acquisition program flexibility and agility to respond to rapidly changing requirements
- **Improve the ability of DoD institutions to develop and exploit large-scale computational science and engineering tools - \$360M over 12 years**



COST PERFORMANCE LOCKED IN EARLY

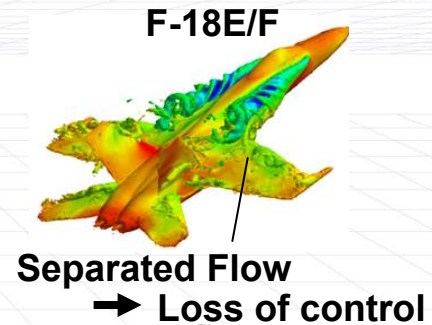




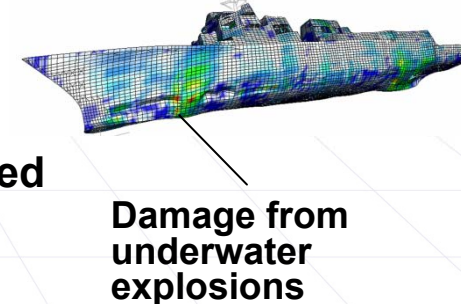
Computational Research and Engineering Acquisition Tools and Environments—CREATE



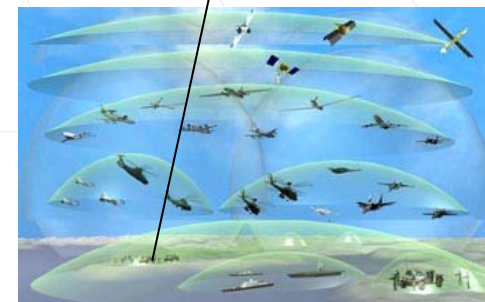
- CREATE will develop and deploy three computational engineering tool sets for acquisition program engineers to exploit the exponential growth in supercomputer power:
 - Aircraft tools (Aerodynamics & Structures)
 - Ship tools (Hydrodynamics & Structures)
 - Antenna Integration tools (Electromagnetics)
- Aircraft design tool capability:
 - Aerodynamic control & stability and loads for complete airframes including propulsion and control systems integrated with the airframe response
- Ship design tool capability:
 - Early stage design and shock hydrodynamics coupled to structural mechanics for full ship shock certification integrated with hull form optimization and capsizing stability analysis with full ship motion in ocean waves
- Antenna Integration and design tool capability:
 - Coupling of antenna radiation with platforms for minimal interference (space and frequency) and simultaneous full power multi-antenna operation
- + Software Infrastructure-Meshing, Collaboration tools.....



DDG-1000



C4ISR and sensing
antennas in Network
Centric Warfare
Battlespace





What will CREATE Need To Succeed?



Successful computational engineering and scientific projects emphasize*:

- **Verification and Validation**
 - Accurate, reliable results
- **Computational Engineering Software Project Management**
 - Large teams (~30 professionals) need a project orientation to organize and coordinate the work; single investigator paradigm doesn't work
- **Computational Engineering Software Engineering**
 - Computational Engineering software development is a complex process for producing a complex system
 - Success requires effective methods and tools that balance the need for structured development with the required degree of flexibility and agility
 - Strong connection to the customers is required to meet their evolving needs
 - Good team dynamics: trust, respect, cooperation and commitment

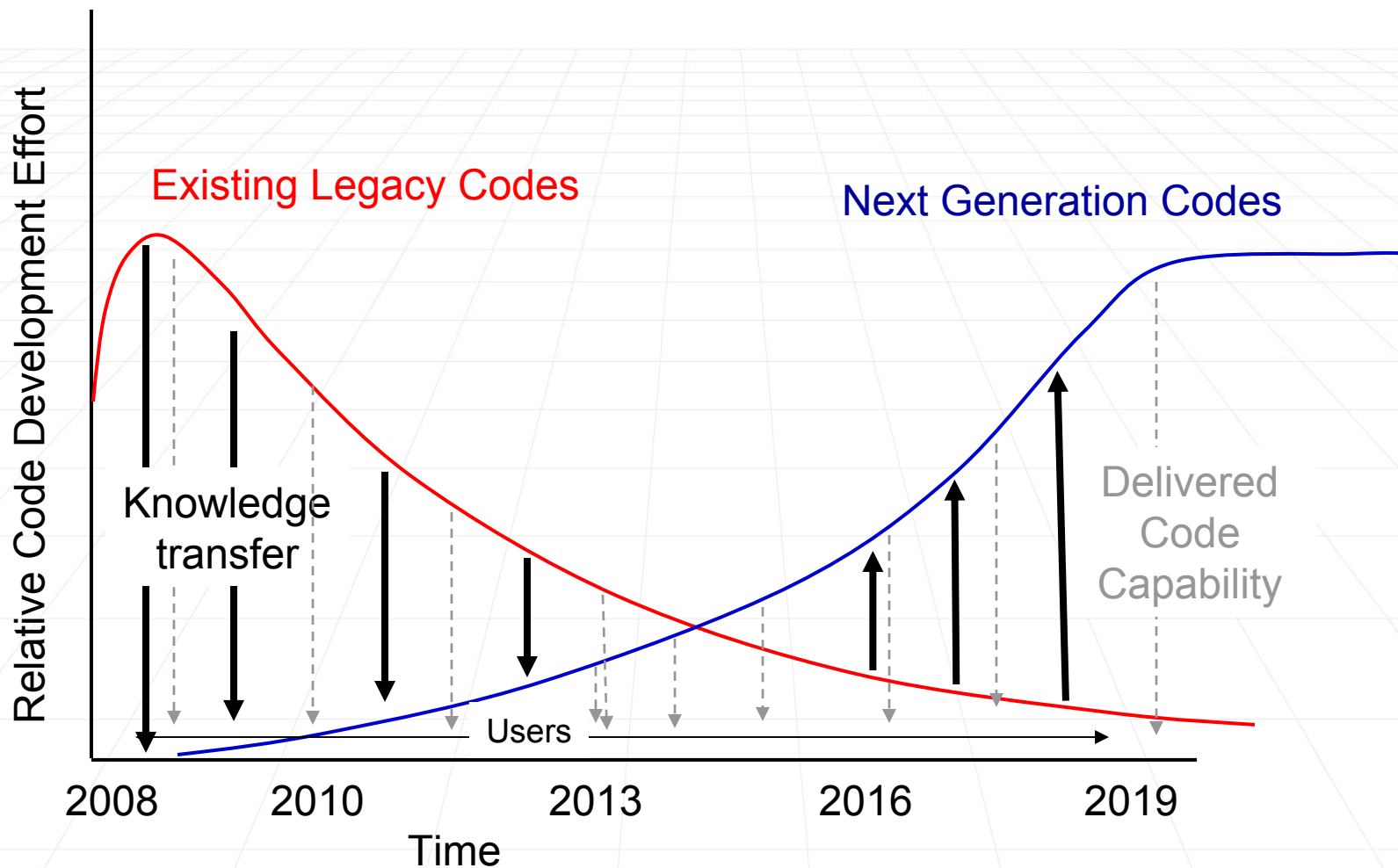
DoD Challenge:

- Establish a set of well integrated, multi-institutional, multi-disciplinary computational engineering code development teams

**Software Project Management and Quality Engineering Practices for Complex, Coupled MultiPhysics, Massively Parallel Computational Simulations, D. E. Post and R. P. Kendall, The International Journal of High Performance Computing Applications, 18(2004), pp. 399-416*

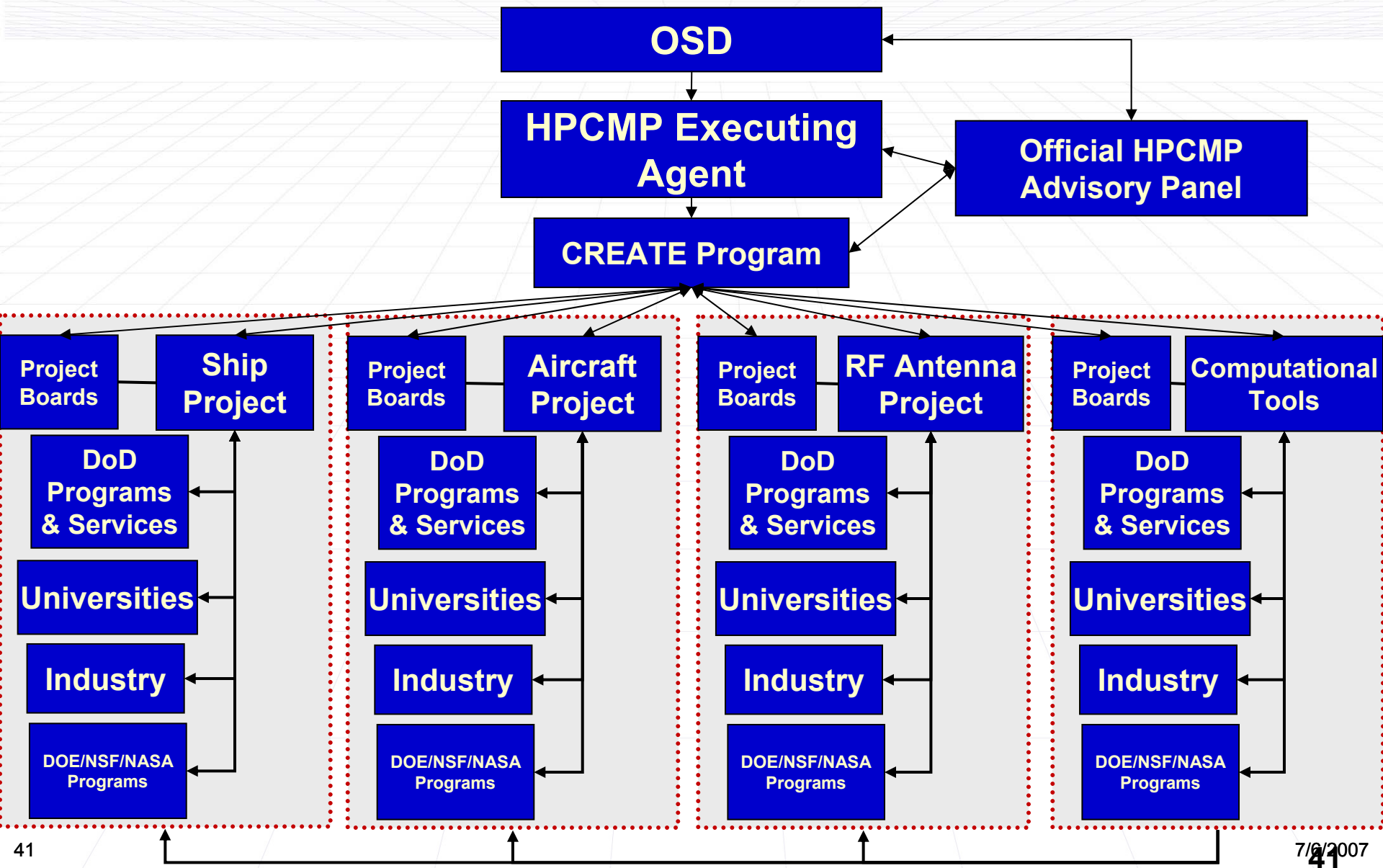


Prototype Codes Will Be Gradually Replaced With Next Generation Codes





CREATE is a Multi-Institutional Program

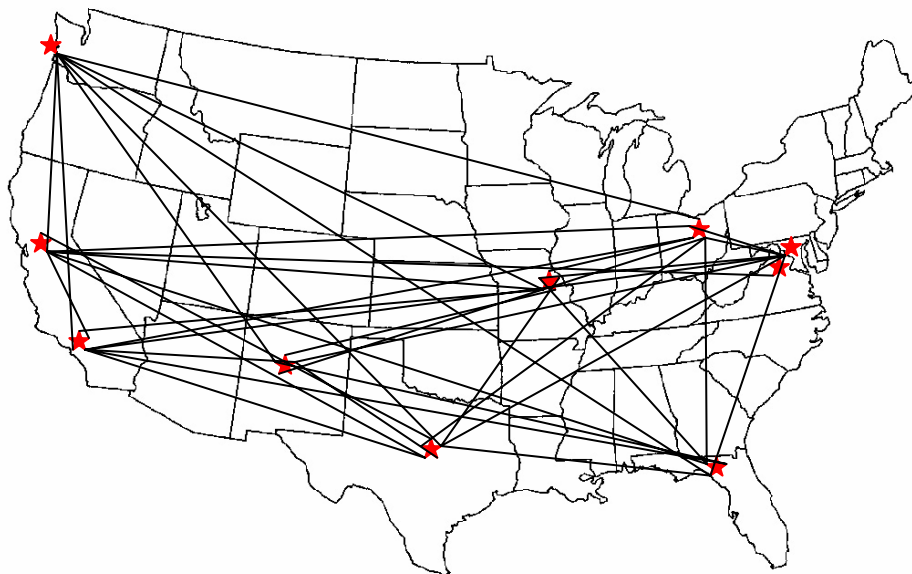




Establishing a Multi-Institutional, Multi-Disciplinary Collaboration Is a Daunting Challenge



- Coordinating colocated code development by one institution has proven very challenging
- Coordinating non-colocated code development by multiple institutions will be even more challenging
- Establish the right culture, behavior and control
- Form a team whose members have trust and respect for each other and a strong commitment to the success of the project
- Provide support for collaboration tools (hardware, software, and user help).
- Effective desktop video communication
- Effective daily communication
- Propose an aggressive program to develop and deploy collaboration tools and methods, budgeting up to \$750k/year





- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**



“HPC needs a tools strategy! **”



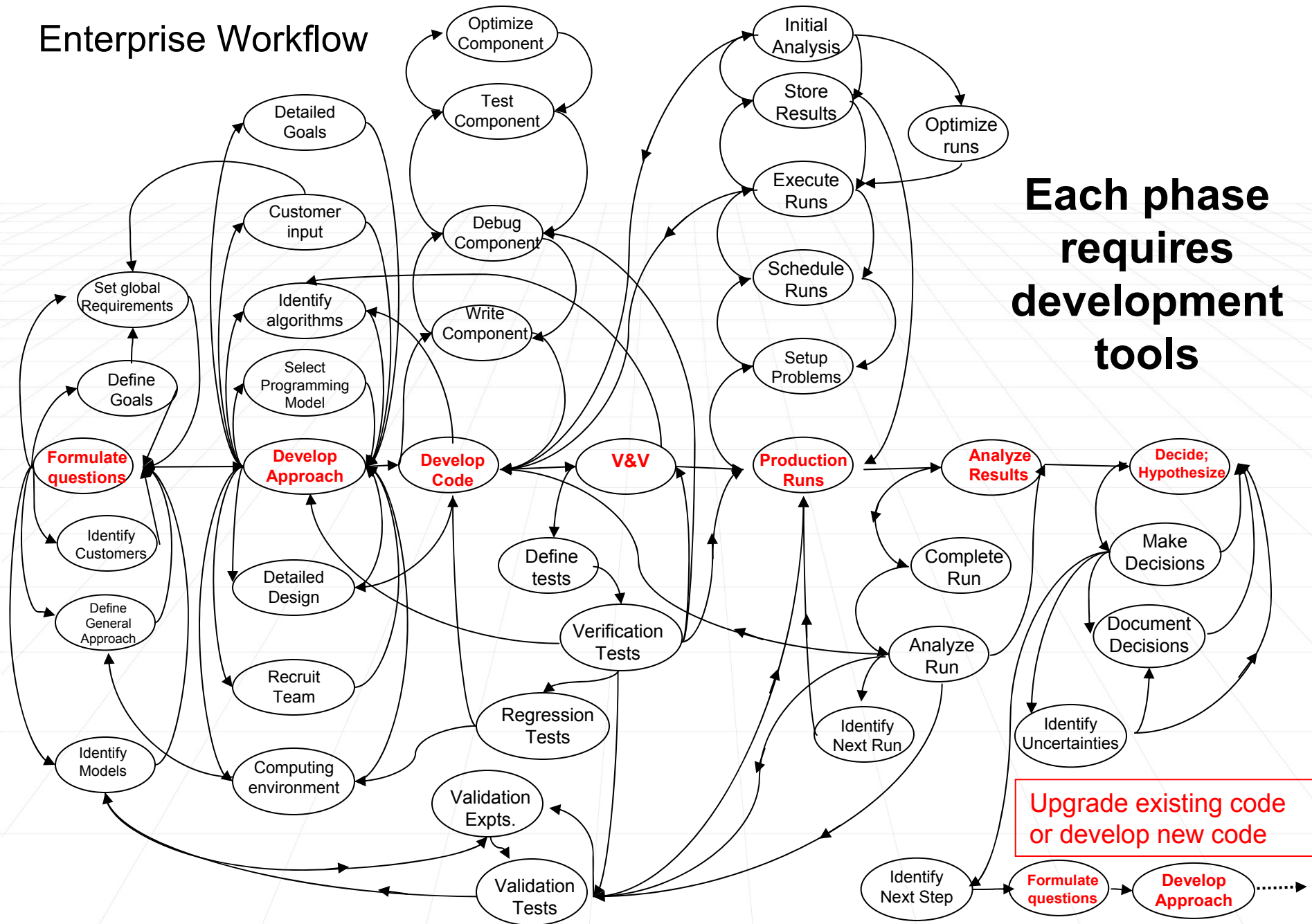
- There is not a good business model for code development tools!
- Academia has developed some good tools, but the support is generally meager
- Highly successful tools developers get bought by INTEL or Microsoft or..., and then don't serve the general community (Kuck and Associates, Pallas.....)
- Unsuccessful tool developers go out of business
- Modestly successful tool developers seem to survive, but there aren't many (Totalview, CEI, Portland Group...)
- Government support for tool development is waning
- “HCP needs a tools strategy! **”
- VI-HPS can play an important role!
- Pick a few tools and do a good job, build the user base
- Emphasize utility and user support

“Last Man Standing”



*HPC Needs a Tool Strategy, M. VanDeVanter, D.E. Post, M.E. Zosel, 2nd Workshop on HPC applications, ACM/IEEE International Conference on Software Engineering, St. Louis, MO, May 22, 2005

Enterprise Workflow





Four HPC Code Development Tool Categories



- **Category I: Code development environment—**
Operating system, text editors, compilers, scripting languages, debuggers, syntax checkers, static analysis tools, job schedulers, performance analysis, memory trace and analysis tools, bug trackers, issue trackers, IDEs...
 - **Examples: MatLab —► Octave, TotalView, Eclipse, ~~Vampir~~, Tau, Open Speedshop.....**
- **Category II: Production tools—**mesh generators, data analysis and assessment, visualization, checkpoint restart, runtime documentation, V&V...



Tools we need (continued)

- **Category III: Software project management and collaboration tools—configuration management, code design, documentation, web design, project management, issue tracking, collaboration tools...**
- **Category IV: Libraries—computational math libraries, data libraries, parallel programming libraries...**
 - **Possibly best option for scaling and reuse**



How do people optimize CSE codes now?



- Most start with the design of the code, take into account the domain science and engineering, and the solution algorithms as well as optimizing for the computer
 - Minimizing latency penalties is the first priority
 - Load balancing is the second priority
- Now many, if not most, large-scale CSE codes are designed with massive parallelization in mind
- The number of codes that begin as large serial codes that are refactored into parallel codes is diminishing
- Codes are “tuned” and reconfigured to some extent for individual platforms, but more emphasis is given to portability
- Better and easier to use tools would be a great help



Qualitative Performance Gains Also Come From Better Computers and Algorithms



- **Faster, bigger computers**
- + **Optimization of code performance for the computers**
- + **Faster, more accurate computational mathematics (FFT, Metropolis, Krylov iteration schemes..)**
- + **Improved domain science algorithms (capture key features of domain science in an algorithm without computing the most basic phenomena)**
 - **Come from basic insights into the physics, chemistry, biology, etc.**
 - **Flux limiters, average ion model, flux surfaces, phenomenological models...**
 - **Symmetries, conservation laws, thermodynamic constraints, detailed balance...**
- **More emphasis on computational mathematics for massively parallel machines is needed since they are the future**



- **Promise**
- **HCPMP**
- **Code characterization**
- **Challenges**
- **Solutions**
- **CREATE - 3 new projects testing the solutions**
- **Tools part of solution**
- **Conclusions and recommendations**



“Top 500” list based on LINPACK is a major impediment for good performance.



- **Charles Holland, DARPA IT head, terms it “Worshipping false gods”**
- **LINPACK benchmarks are not a good indicator of the performance of real CSE codes on today’s massively parallel computers where latency is a big problem**
- **Yet LINPACK performance is the “de facto” metric for computer vendors**
- **The vendors are “forced” by the market to develop computers to achieve high Linpack performance even if it involves penalizing the performance of real applications**
- **As long as the top 500 list based on LINPACK continues to be the community metric, performance for real codes will not be strongly emphasized**
 - **Result: Code development will continue to get more challenging**
- **Need new metrics for computer performance, but it’s a lot of work!**
 - **HPC Challenge.....**
 - **Real codes as benchmarks (HPCMP does this! NASA and NSF adopting this)**
 - **Synthetic benchmarks**
 - **Performance analysis**
- **Europe should learn from the mistakes of others, it shouldn’t “worship false gods”**



Recommendations

- Computational science and engineering has a great potential to solve many problems important to society, but it is challenging, takes a long time and resources, and has risks
- The growing complexity of computers provides both opportunities and challenges
- **Success requires good tools for code development and production**
- The tools haven't kept up with computer hardware, and the business model is broken
- Large opportunity, and challenge, for VI-HPS to have a real impact
- **Recommend that VI-HPS study the user community and the new hardware architectures, identify a few crucial opportunities where good tools will make a big difference, then develop and support those tools as long as they are useful .**
- Emphasize a few good tools rather than many tools.
- Multi-core, heterogeneous computing tools – emphasize ease of use



The Future

- In the words of ancient curse, we live in “exciting times”, full of both opportunities, challenges and dangers
- CSE offers tremendous promise to address and solve important problems
 - The potential to tackle and solve problems that we couldn’t until now
- CSE faces many challenges just like every other new problem solving methodology has faced
- Don’t be discouraged if it takes a long time for the methodology (CSE) to mature
- It will take time and a lot of hard work to overcome the challenges, but we will eventually prevail