The Promise and Challenges of Large-Scale Computational Science and Engineering

Douglass Post

Chief Scientist

DoD High Performance Computing Modernization Program

(IPA from CMU Software Engineering Institute)

12th workshop on the Use of High Performance Computing in Meteorology, 30 Oct - 3 Nov 2006,
European Centre for Medium-Range Weather Forecasts (ECMWF)
Reading, UK

HPC Modernization Program

Army HPCMP Resources

ARL & ERDC MSRCs
AHPCRC & SMDC ADCs
ATC, DPG, & CERDEC DHPIs
1,147 Users/25 Locations/87 Projects
51 DREN Sites
10 Challenge Projects
3 Institutes/1 Portfolios

Navy HPCMP Resources

NAVO MSRC
FNMOC, SSCSD, NAWCWD, NSWCCD, & NUWC DHPIs
1,035 Users/25 Locations/205 Projects
30DREN Sites
10 Challenge Projects
3 Institutes

Air Force HPCMP Resources

ASC MSRC
MHPCC ADC
AEDC/AFSEO, AFWA, & MHPCC DHPIs
1,161 Users/36 189Projects
20 DREN Sites
13 Challenge Projects
2 Institutes/1 Portfolio

Defense Agencies

DARPA, DTRA, JNIC, JFCOM, MDA, & OTE
JFCOM DHPI
405 Users/4 Locations/11 Projects
29 DREN Sites
3 Challenge Projects

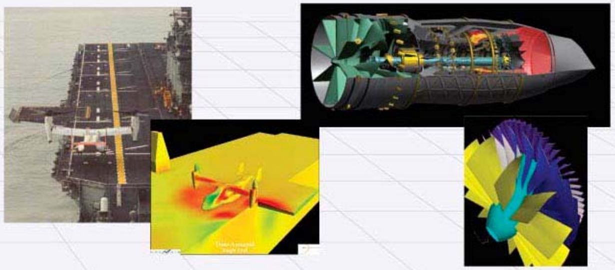
Other

ARSC ADC
MIT-LL DHPI

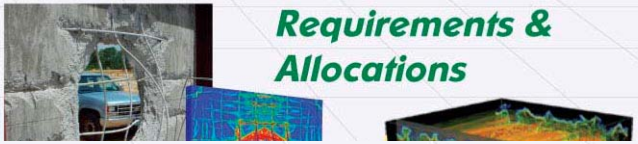
HPC Centers

Resource Mgmt

DoD Challenge Projects



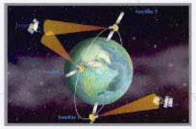
Requirements & Allocations



Networking

Software Applications Support

HSAIs/Portfolios



SPI

SOFTWARE PROTECTION INITIATIVE
Application Software Security Spanning Desktops to Supercomputers

PET

Support to the Warfighter
Enabling Technologies
Outreach
Training
Computational Environments
Education
Support for Computational Technology Areas

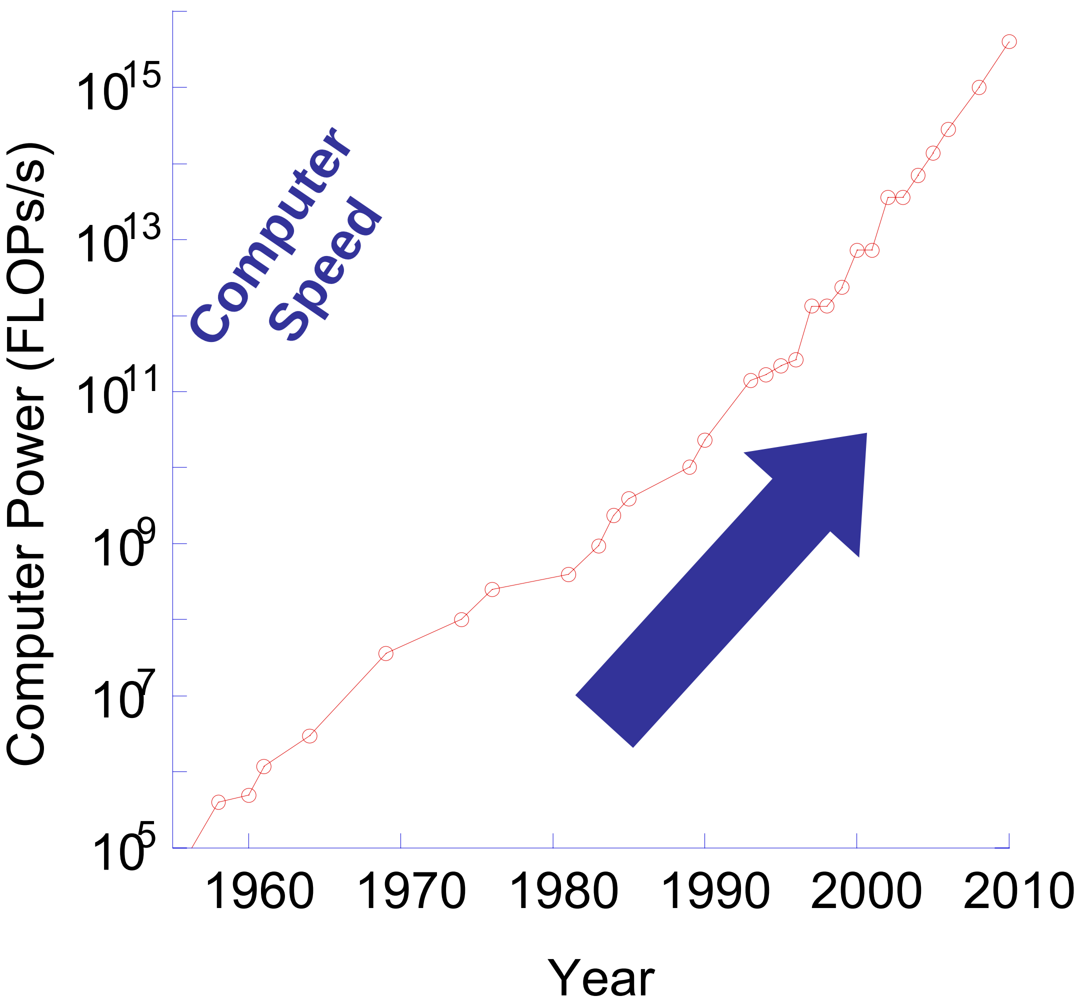
Education Outreach

8/16/2006

Exponential Growth In Supercomputer Speed And Power Is Making It A “Disruptive” Technology.

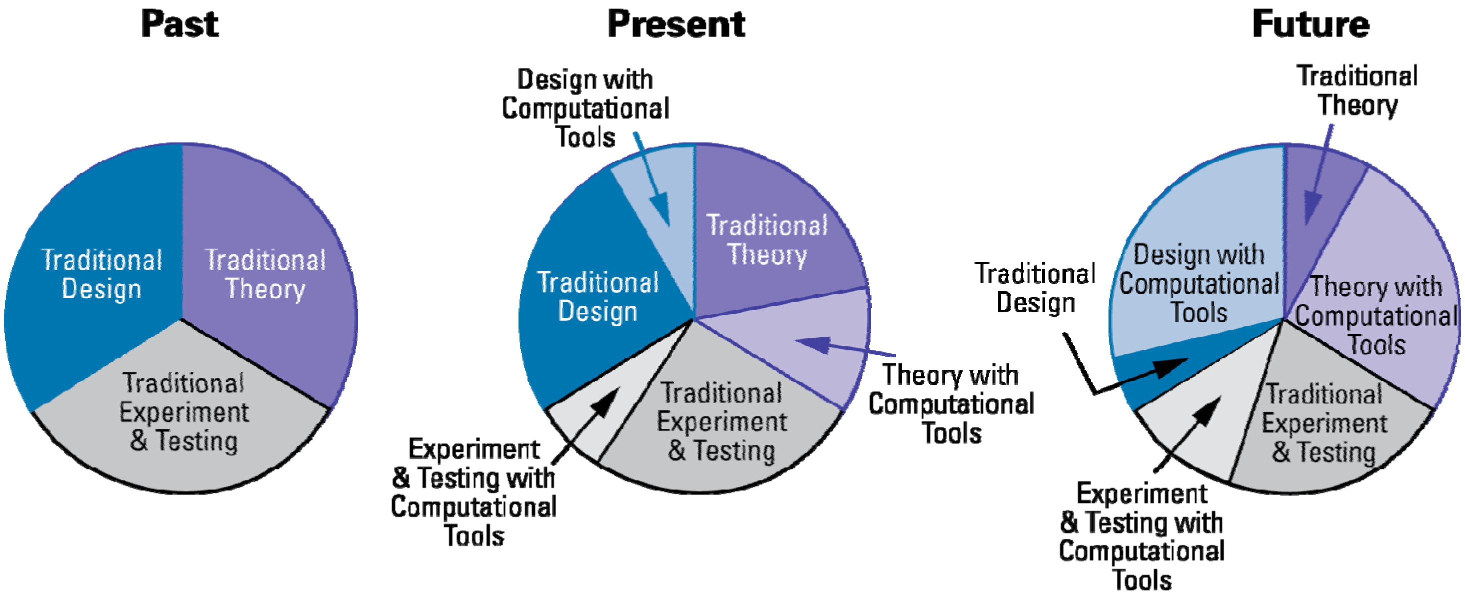
Enable paradigm shift

- 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!



Computer power comes at the expense of complexity!

Computational Tools are becoming widely used in Science and Science



	Past	Present	Future
Theory	Pencils, paper; slide rules	New: symbolic math; computational solutions	New: Almost all computional
Experiments	Physical hardware; notebooks; chart recorders; polariod 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 governing individual interactions are known

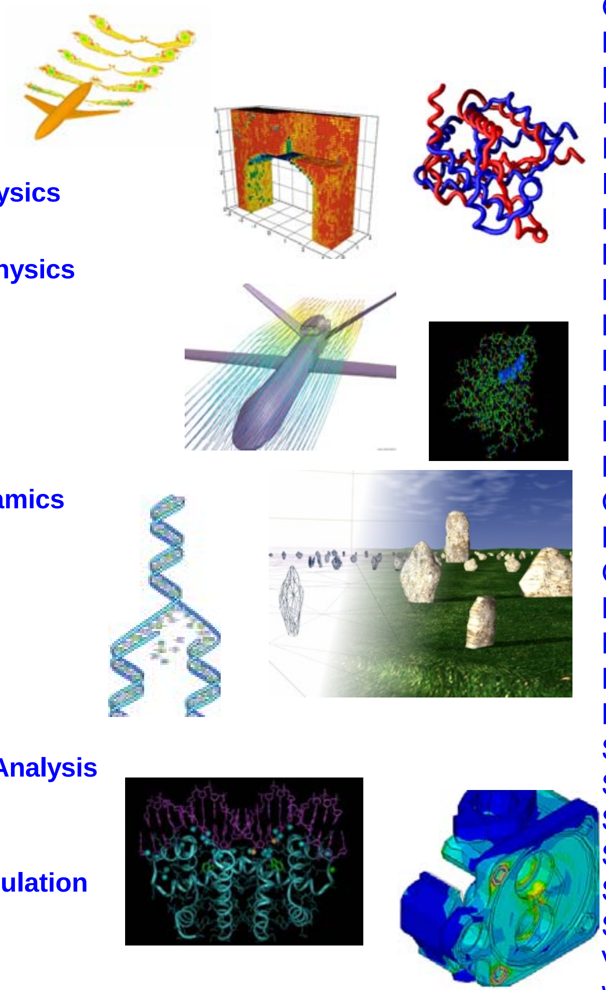
1. Scientific discovery
2. Experimental analysis and design
3. Prediction of operational conditions
4. Scientific design and analysis
5. Engineering design and analysis

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

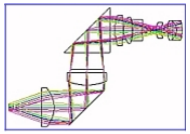
6. Data collection, analysis & mining
   - Social sciences, medicine, education, research
7. Heuristic simulations and decision tools (economic forecasts, war and strategy simulations,..)

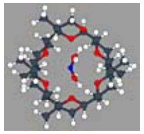
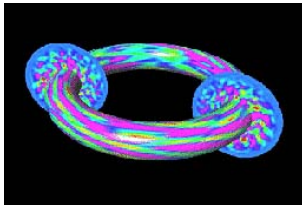
Computational Science and Engineering is becoming ubiquitous in science and engineering

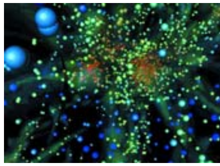
Accelerator Design
Aircraft Design
Archaeology
Armor Design
Astrophysics
Atomic And Molecular Physics
Automobile Design
Bioengineering And Biophysics
Bioinformatics
Chemistry
Civil Engineering
Climate Prediction
Computational Biology
Computational Fluid Dynamics
Cosmology
Cryptography
Data Mining
Drug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

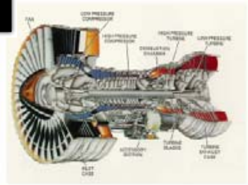
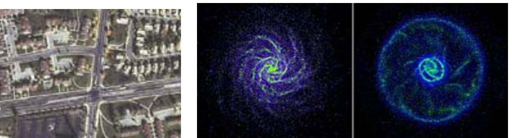
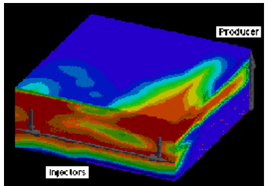


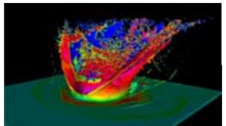
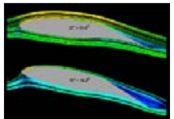
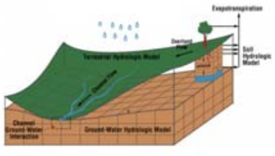
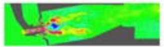
Groundwater And Contaminant Flow
High Energy Physics Research
Hydrology
Image Processing
Inertial Confinement Fusion
Integrated Circuit Chip Design
Magnetic Fusion Energy
Manufacturing
Materials Science
Medicine
Microtomography
Nanotechnology And Nanoscience
Nuclear Reactor Design And Safety
Nuclear 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





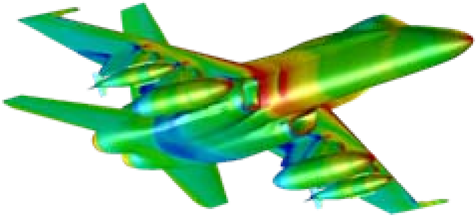




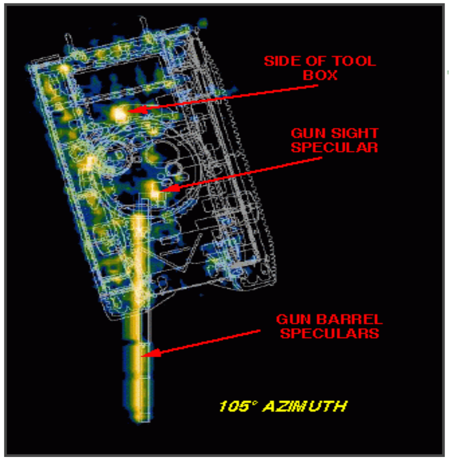
Computational Science and Engineering contributes today

- Computational Science and Engineering is making major contributions today
- DoD— Stores separation, weather and ocean prediction, materials, armor penetration, RF antenna and radar signatures, aircraft and ship design and analysis, bio-warfare countermeasures, and many more
- DOE, NSF, NIST, NASA, NOAA, EPA,…— high energy and nuclear physics, nuclear weapons design, controlled fusion, materials, nuclear reactor, fuel efficiency, geophysics, astrophysics, space physics, and many more
- Industry— Crash design (GM,…), Tire design (Goodyear), chip design (Intel,..), consumer products (P&G,..), aircraft (Boeing, Airbus), structural design, drug design and data searches (Merck,…), oil exploration, and many more

Hurricane tracking

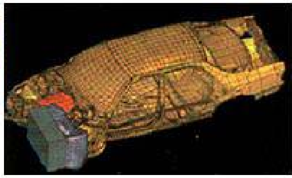


Aircraft Design

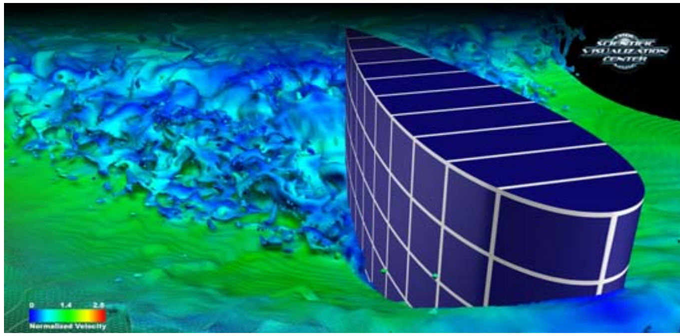


Radar Signature of a Tank

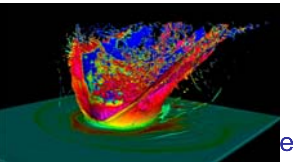
Vehicle crash



Nuclear Weapons



Breaking Waves Around a Ship

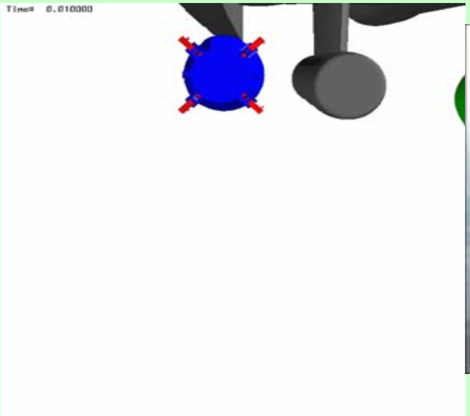


Asteroid Impact

Stores Integration & Certification
Supercomputing Improves the Test Process

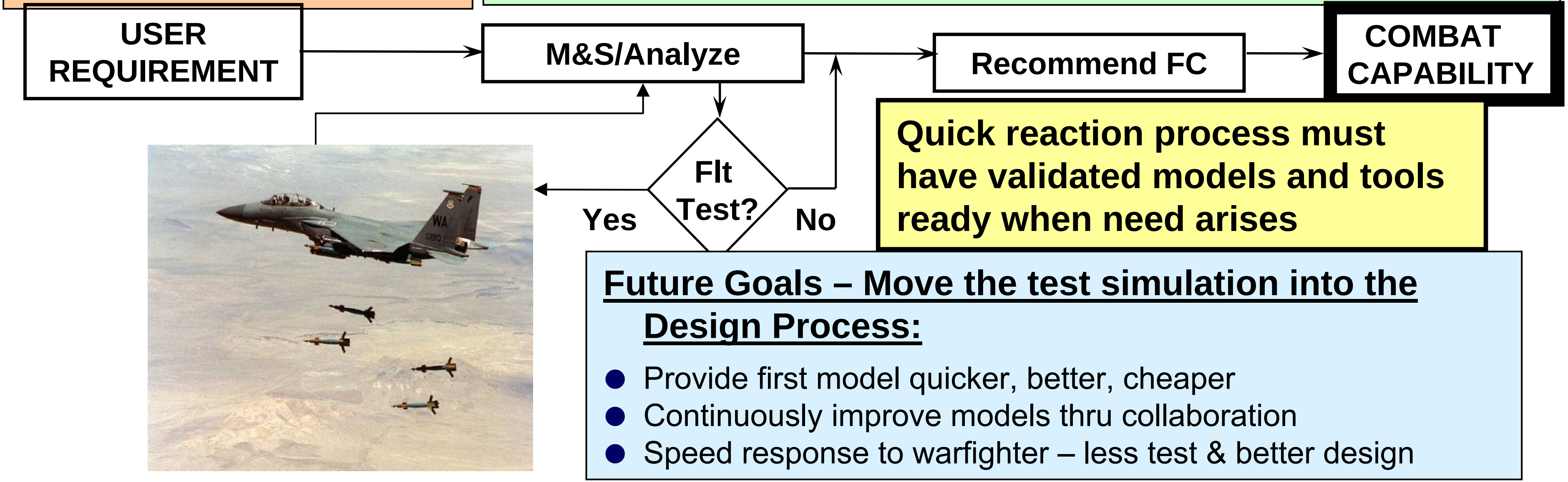
Old Way - Flight test it

Today's Way - Computationally simulate the test and run much reduced flight test



Benefits:

- Faster
- Cheaper
- More technical insight
- Safer



Quick reaction process must have validated models and tools ready when need arises

Future Goals – Move the test simulation into the Design Process:

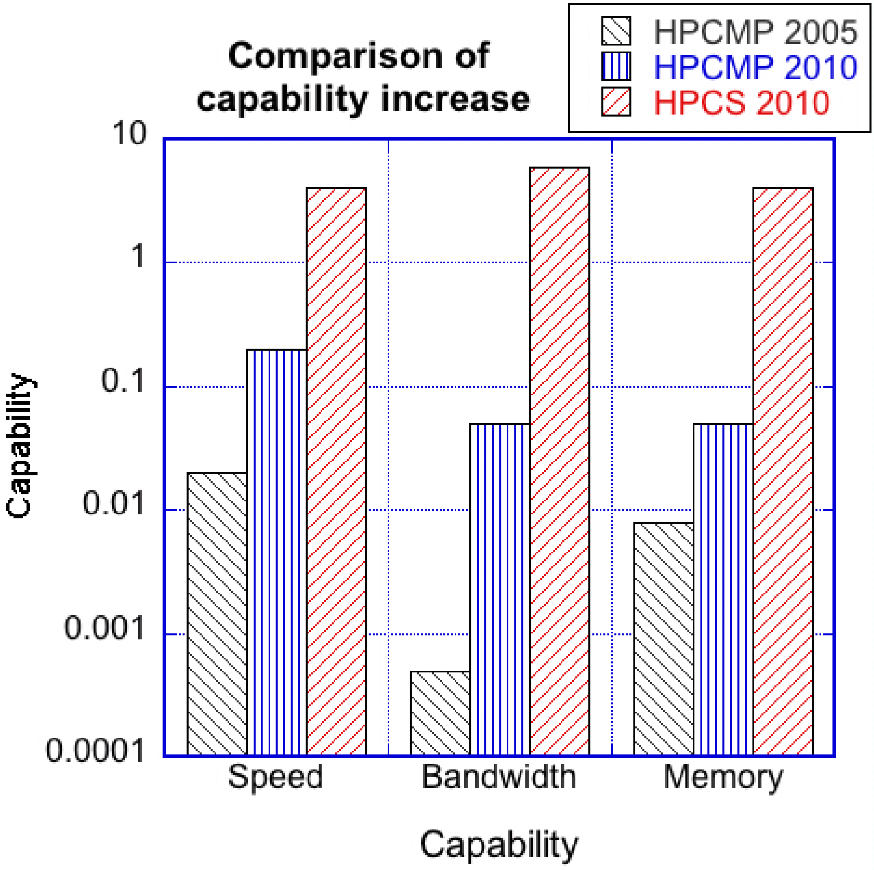
- Provide first model quicker, better, cheaper
- Continuously improve models thru collaboration
- Speed response to warfighter – less test & better design

PetaFlop computers are coming

NSF, DOE Science and Defense Systems, NASA, NOAA, DoD (DARPA) all plan petaflop computers for 2008-2012

- DARPA High Productivity Computing Systems a bright light
  - Faster computing but also
  - Higher bandwidth and lower memory latency (64k GUP/s)
  - Flatter memory hierarchy with globally addressable memory
  - + many more

But are we ready to use them?



To succeed, Computational Science and Engineering faces immense challenges*

- Scientific and Engineering:
  - Calculate the trade-off of many different strongly interacting effects across many orders of magnitude of multiple time and distance scales
  - Verify and validate highly complex applications
  - Develop problem generation and setup methods for larger and more complex problems
  - Analyze and visualize larger and more complex datasets
- Project:
  - Evolve from small code development teams to large teams
  - Successfully deploy multi-disciplinary and multi-institutional code development teams
- Programming:
  - Develop codes for computers that don’t yet exist.
  - Develop codes for computers that will be 10^2 to 10^4 faster and contain 10^2 to 10^3 times more processors than today
  - Develop codes with adequate performance levels
  - Cope with relatively immature tools for developing and running massively parallel applications

*c.f. The Opportunities, Challenges and Risks of High Performance Computing in Computational Science and Engineering, *D.E. Post, R.P. Kendall and R.F. Lucas, Advances in Computers, Quality Software Development,* ***66***, *(2006), M. Zelkowitz, Ed., Academic Press pp. 239-301.*

Lessons Learned are the way forward!!!

- 4 stages of design maturity for a methodology to mature—Henry Petroski—*Design Paradigms*
- Suspension bridges—case studies of failures (and successes) were essential for reaching reliability and credibility
- The Scientific Method!

Tacoma Narrows Bridge buckled and fell 4 months after construction!

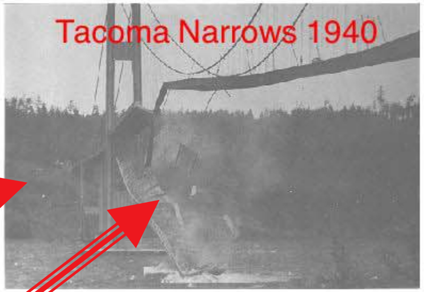
- Case studies conducted after each crash.
- Lessons learned identified and adopted by community
- Computational Science is at stage 3

time

1

2

3



Lessons Learned Case Studies

4

What do CSE applications look like?

Surveyed DoD and other codes to verify characterizations of CSE codes.

- Identify general characteristics
- Preamble (anonymity guaranteed)

Questionnaire asked for:

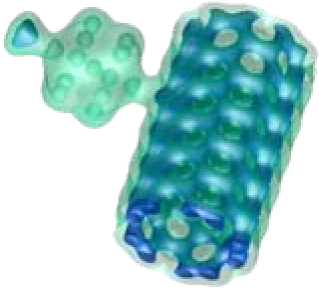
- Contact information
- Code purpose
- Team size, number of users
- Domain Science area and sponsor
- Code size (slocs)
  - Total and for each language
- Code history
  - How long did the code take to develop and how old is it now?)
- Platforms
- Degree of parallelism
- Computer time usage
- Memory requirements
- Algorithms

What kind of cods are we talking about?

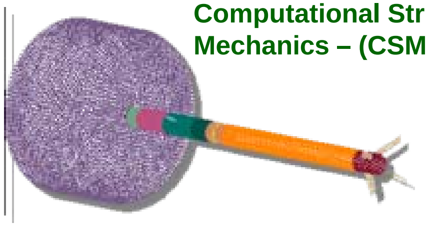
We surveyed our Large, Diverse DoD HPC Community to characterize our codes

- **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)**

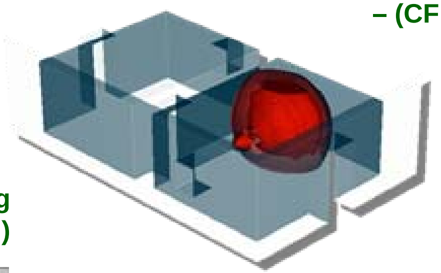
Electronics, Networking, and Systems/C4I – (ENS)



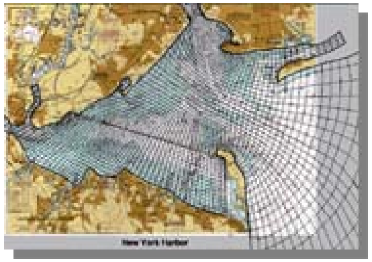
Computational Structural Mechanics – (CSM)



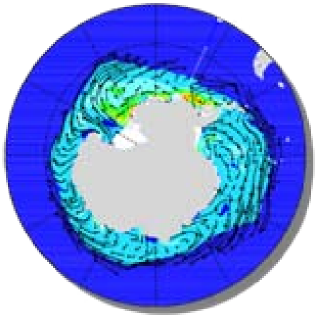
Computational Fluid Dynamics – (CFD)



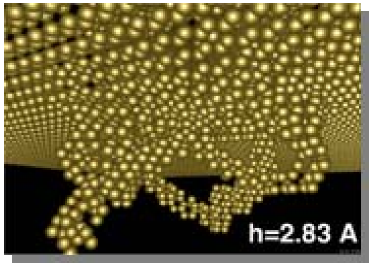
Environmental Quality Modeling & Simulation – (EQM)



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

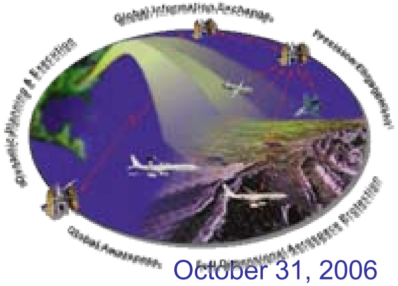


Computational Chemistry, Biology & Materials Science – (CCM)

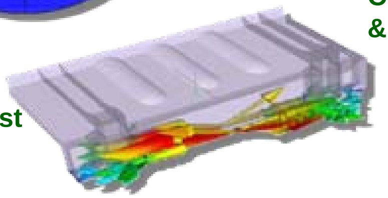


Forces Modeling & Simulation – (FMS)

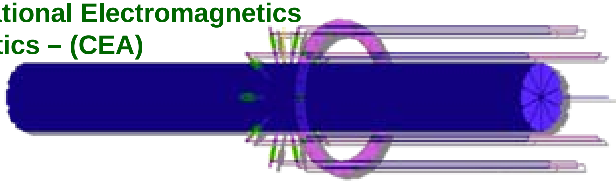
Signal/Image Processing – (SIP)



Integrated Modeling & Test Environments – (IMT)



Computational Electromagnetics & Acoustics – (CEA)



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

Application Code	Hours
CTH (SNL)	93,435,421
HYCOM (30% DoD)	89,005,100
GAUSSIAN (Commercial)	49,256,850
ALLEGRA (SNL)	32,815,000
ICEPIC (100% DoD)	26,500,000
CAML (100% DoD)	21,000,000
ANSYS (Commercial)	17,898,520
VASP (U.ofVienna)	18,437,500
Xflow (Commercial)	15,165,000
ZAPOTEC (SNL)	12,125,857
XPATCH (DoD commercial)	23,462,500
MUVES	10,974,120
MOM	18,540,000
OVERFLOW (NASA)	8,835,500
COBALT (commercial)	14,165,750
ETA	11,700,000
CPMD (MPI & IBM)	5,975,000
ALE3D (LLNL)	5,864,500
PRONTO (SNL)	5,169,100

Application Code	Hours
DMOL	5,200,100
ICEM (commercial)	4,950,000
CFD++ (commercial)	5,719,000
ADCIRC (DoD + academia)	4,100,750
MATLAB (commercial)	4,578,430
NCOM	5,080,000
Loci-Chem	5,500,000
GAMESS (Iowa State)	5,142,250
STRIPE	4,700,000
USM3D	4,210,000
FLUENT (commercial)	3,955,610
GASP	4,691,000
Our DNS code (DNSBLB)	2,420,000
ParaDis	4,000,000
FLAPW	4,050,000
AMBER	4,466,000
POP (LANL)	3,800,000
MS-GC	3,500,000
TURBO	3,600,600
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 inter-operable with Fortran with MPI.

Further data isn’t surprising either.

	Total project age	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.

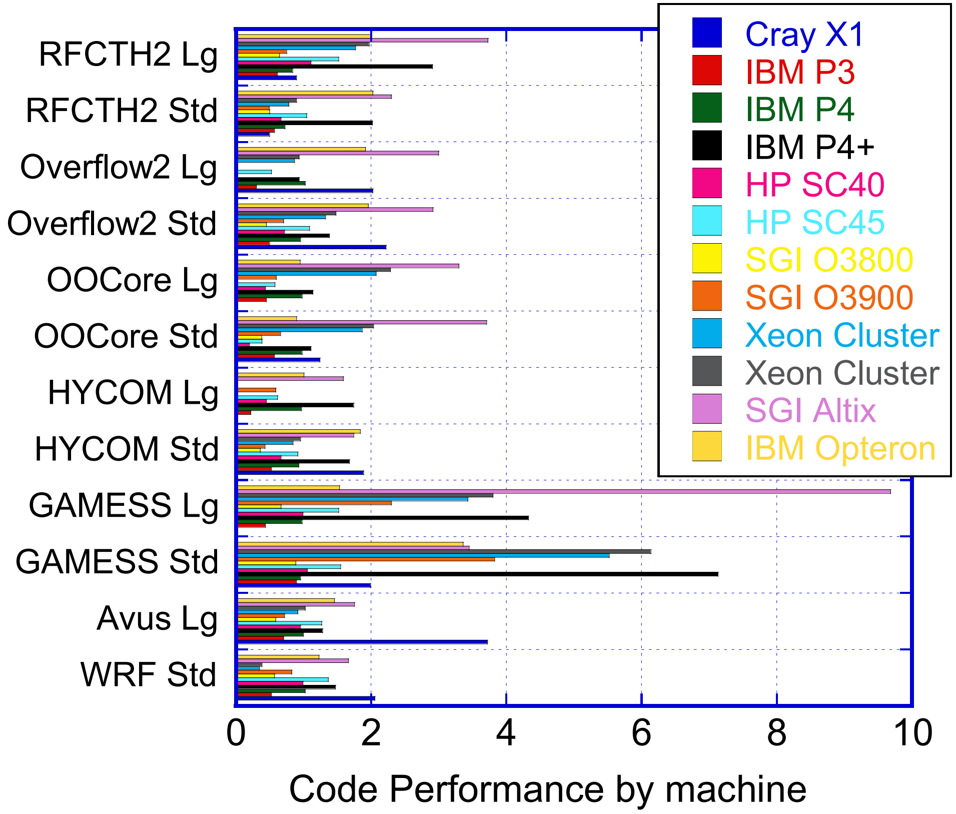
HPCMP TI-05 Application Benchmark Codes perform differently on different platforms.

- **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 modeling code
  (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.

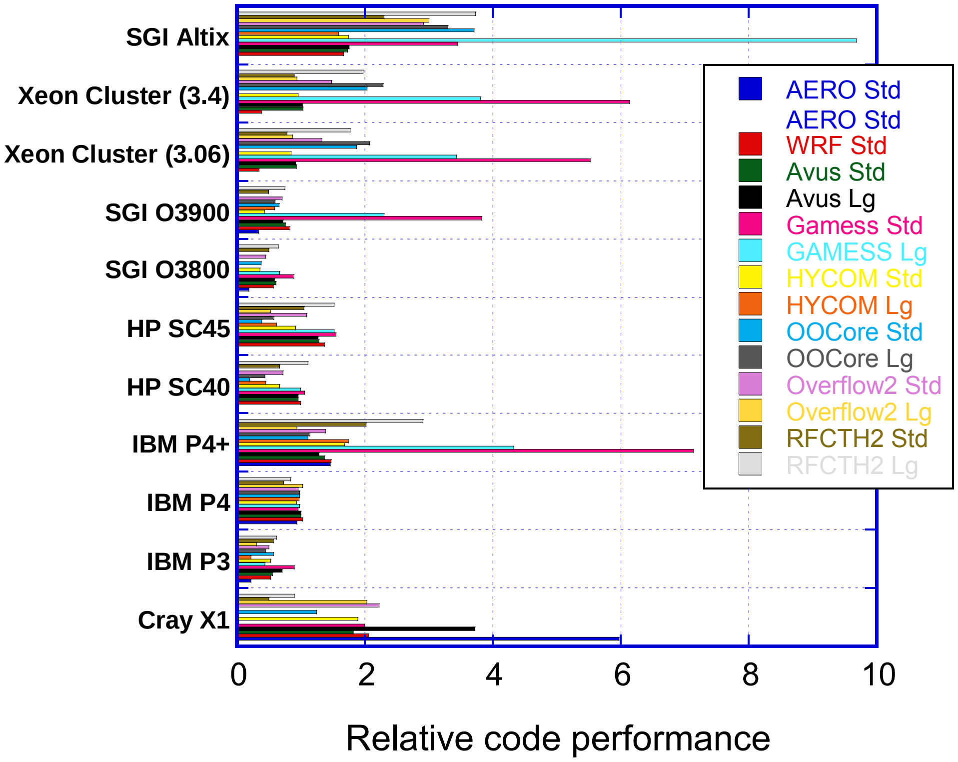
- Normalized Performance = 1 on the NAVO IBM SP3 (HABU) platform with 1024 processors (375 MHz Power3 CPUs) assuming that each system has 1024 processors.
- GAMESS had the most variation among platforms.

Code Performance (by machine)



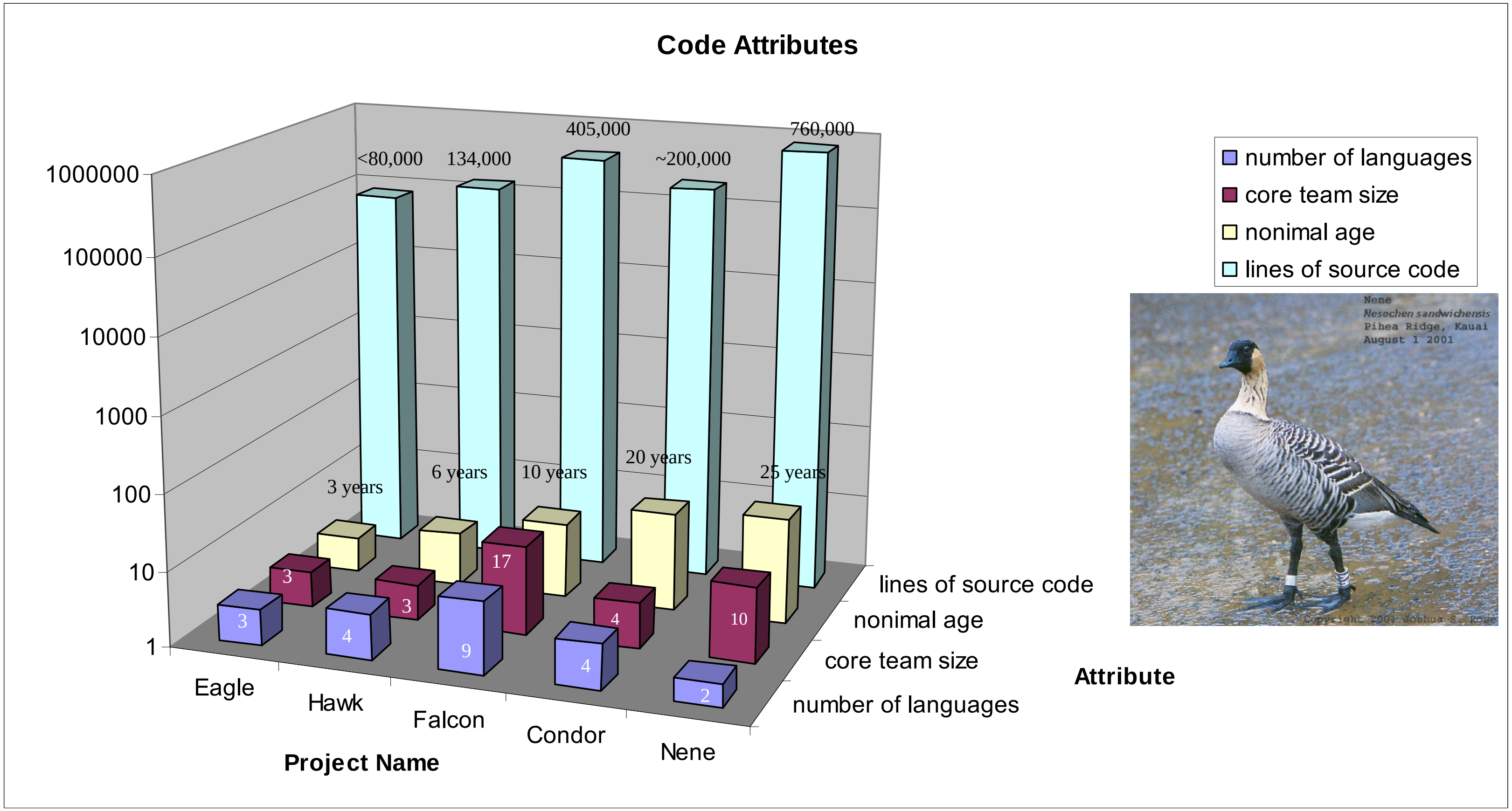
Substantial variation of codes for a single computer.

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)

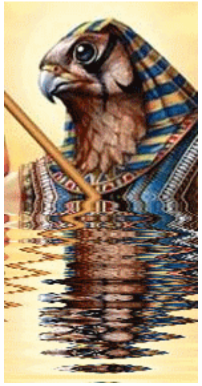
Use of Higher-Level Languages

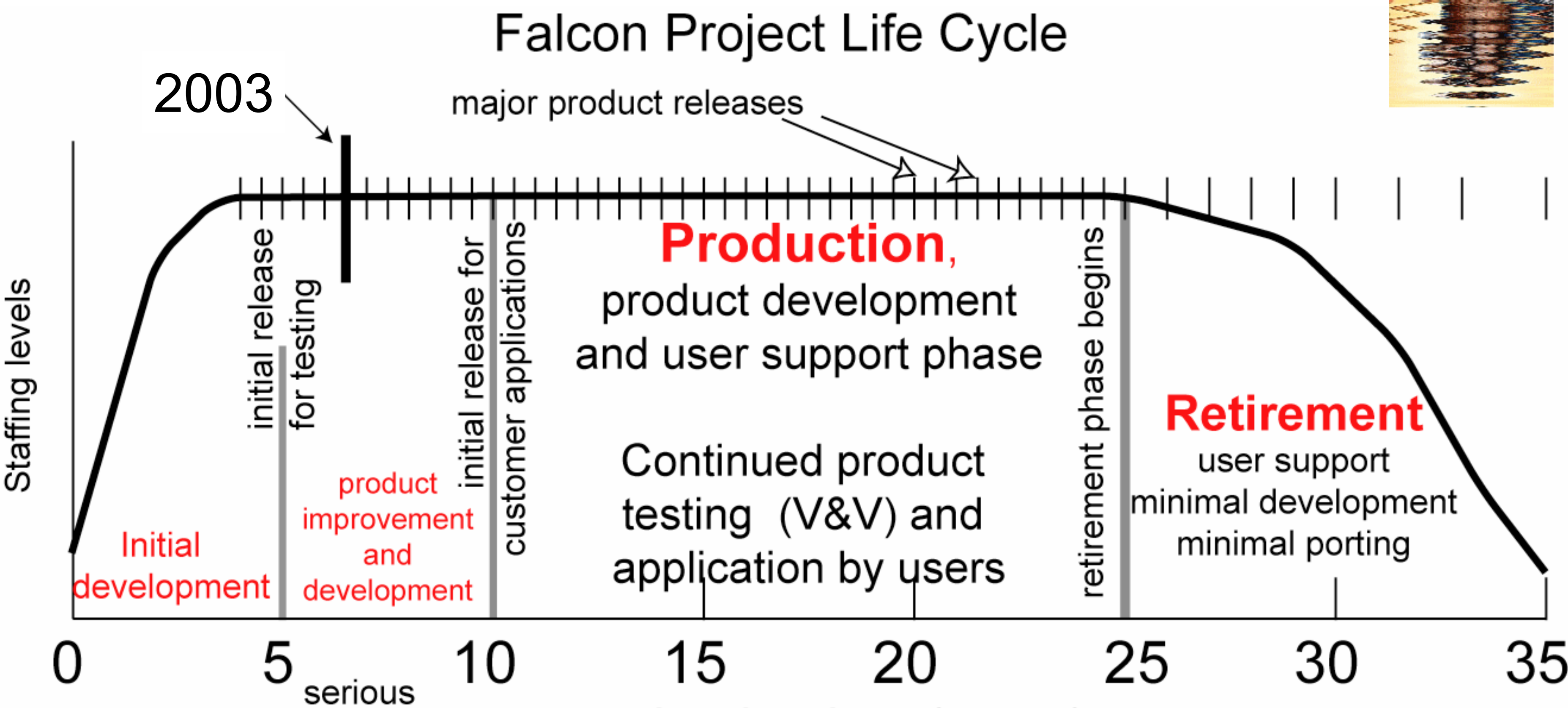
	Falcon	**Hawk**	**Condor**	**Eagle**	Nene
Application Domain	Product Performance	Manufacturing	Product Performance	Signal Processing	Process Modeling
Project Duration	~10 years (since 1995)	~6 years (since 1999)	~20 years (since 1985)	~3 years	~25 years (since 1982)
Number of Releases	9 Production	1	7	1	?
Earliest Predecessor	1970s	early 1990s	1969	?	1977-78
Staffing	15 FTEs	3 FTEs	3-5 FTEs	3FTEs	~10FTEs+100s of contributors
Customers	<50	10s	100s	Demonstration code	~100,000
Nonimal Code Size	~405,000	~134,000	~200,000	<100,000	750,000
Primary Languages	F77 (24%), C (12%)	C++ (67%), C (18%)	Fortran 77 (85%)	C++, Matlab	Fortran 77 (95%)
Other Languages	**F90,Python,Perl,ksh/csh/sh**	**Python, Fortran 90**	**Fortran 90, C, Slang**	**Java Libraries(~70%)**	**C (1%)**
Target Hardware	Parallel Supecomputers	Parallel Supercomputers	PCs to Parallel Supercomputers	Embedded App	PCs to Parallel Supercomputers
Status	Production	Production ready	Production	Demonstration code	Production
Sponsors	DOE	DoD	DoD	DoD	DoD, DOE, NSF

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.

Developing a large, multi-scale, multi-effect code takes a lot of people a long time, and development continues through the entire life cycle of the code.

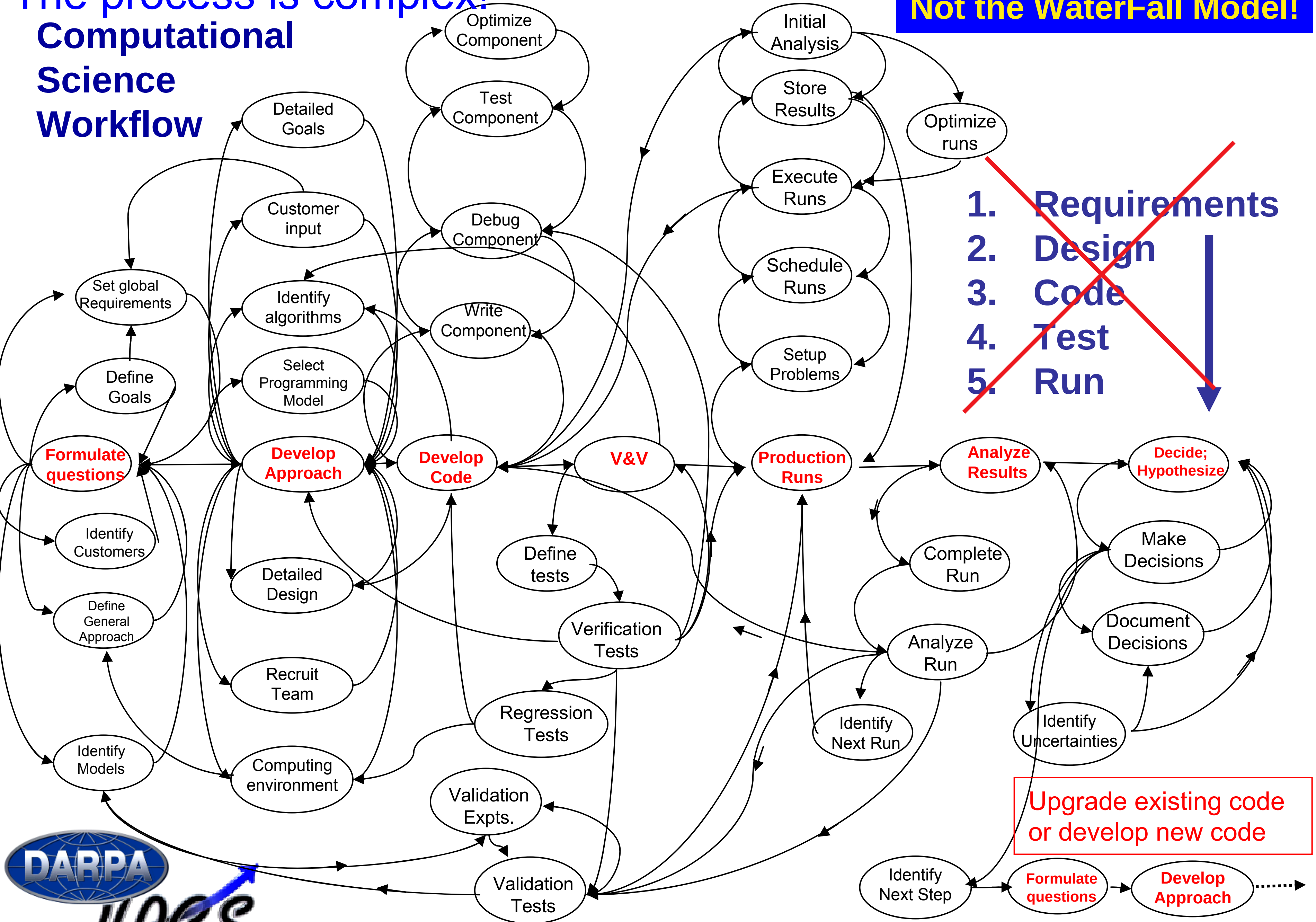




The process is complex!

Computational Science Workflow

Not the WaterFall Model!



Set global Requirements

Define Goals

Formulate questions

Identify Customers

Define General Approach

Identify Models

Detailed Goals

Customer input

Identify algorithms

Select Programming Model

Develop Approach

Detailed Design

Recruit Team

Computing environment

Optimize Component

Test Component

Debug Component

Write Component

Develop Code

Regression Tests

Validation Expts.

Validation Tests

V&V

Define tests

Verification Tests

Initial Analysis

Store Results

Execute Runs

Schedule Runs

Setup Problems

Production Runs

Optimize runs

Identify Next Run

Analyze Results

Complete Run

Analyze Run

Decide; Hypothesize

Make Decisions

Document Decisions

Identify Uncertainties

Identify Next Step

Formulate questions

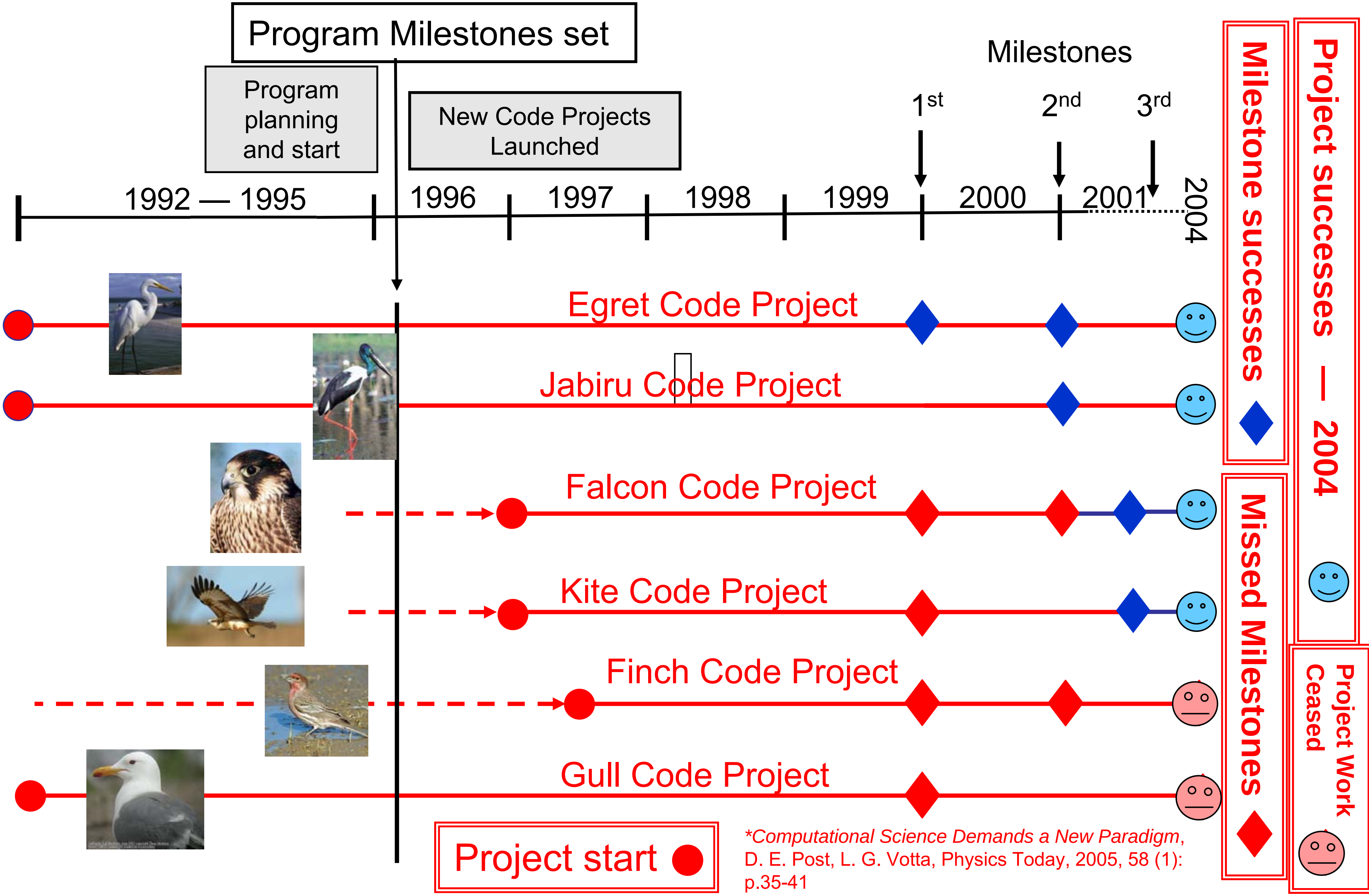
Develop Approach

1. ~~Requirements~~
2. ~~Design~~
3. ~~Code~~
4. ~~Test~~
5. ~~Run~~

Upgrade existing code or develop new code

Risk mitigation often requires redundant projects*.

Code Project Schedule for Six Large-scale Physics Codes



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

Computational Science and Engineering has at Least Four Major Elements.

Computers	Codes	V&V	Users
Making enormous progress but at cost of complexity, particularly memory hierarchy	More complicated models + larger programming challenges	Harder due to inclusion of more effects and more complicated models	Use tools to solve problems, do designs, make discoveries
Need to reduce programming challenge	**Greatest bottleneck**	Inadequate methods, need paradigm shift	Users make connections to customers

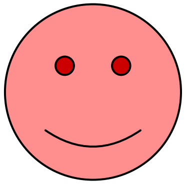
→ **Sponsors**

- **We need to develop a total capability to solve problems, not just build codes or computers.**

What are the needs of CSE Application Codes?

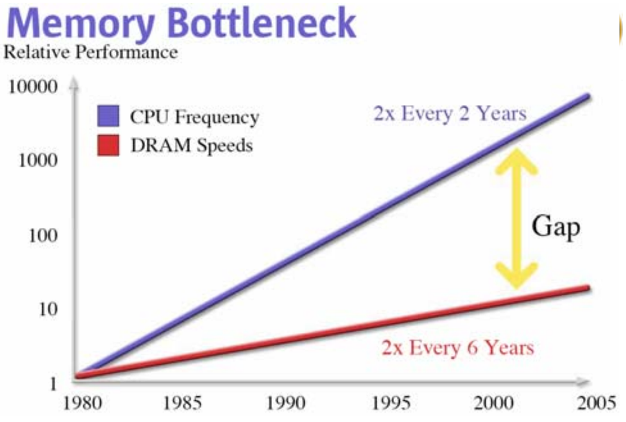
- Developers and production users want and need:
  - Fast integer and floating point arithmetic (with fast divides)
  - Fast, global addressable, reliable memory and data storage with low latency
  - Stable, long-lived and reliable platforms and architectures
  - Stable, long-lived and reliable software 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 workstation development environment or better

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



What are they getting?

- Distributed memory with only very slowly improving memory bandwidth
- Slowing rate of processor speed growth



J. Mitchell, Sun Microsystems

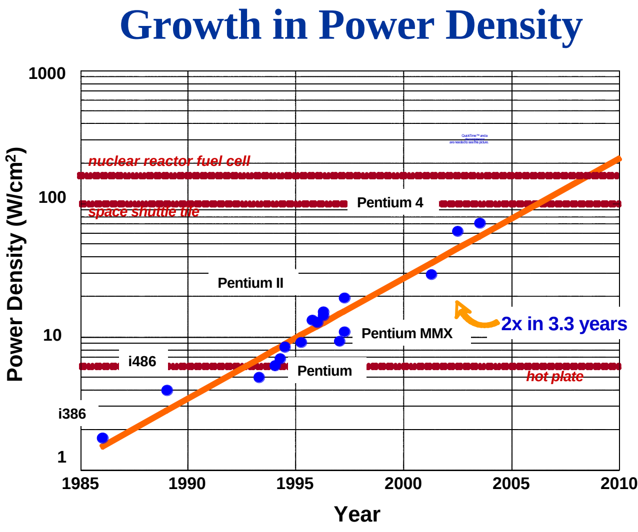


Chart courtesy of Dr. Gary Shaw, MIT/Lincoln Laboratories

- Distributed processor and memory systems linked together in ever more complex networks
- 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

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*.

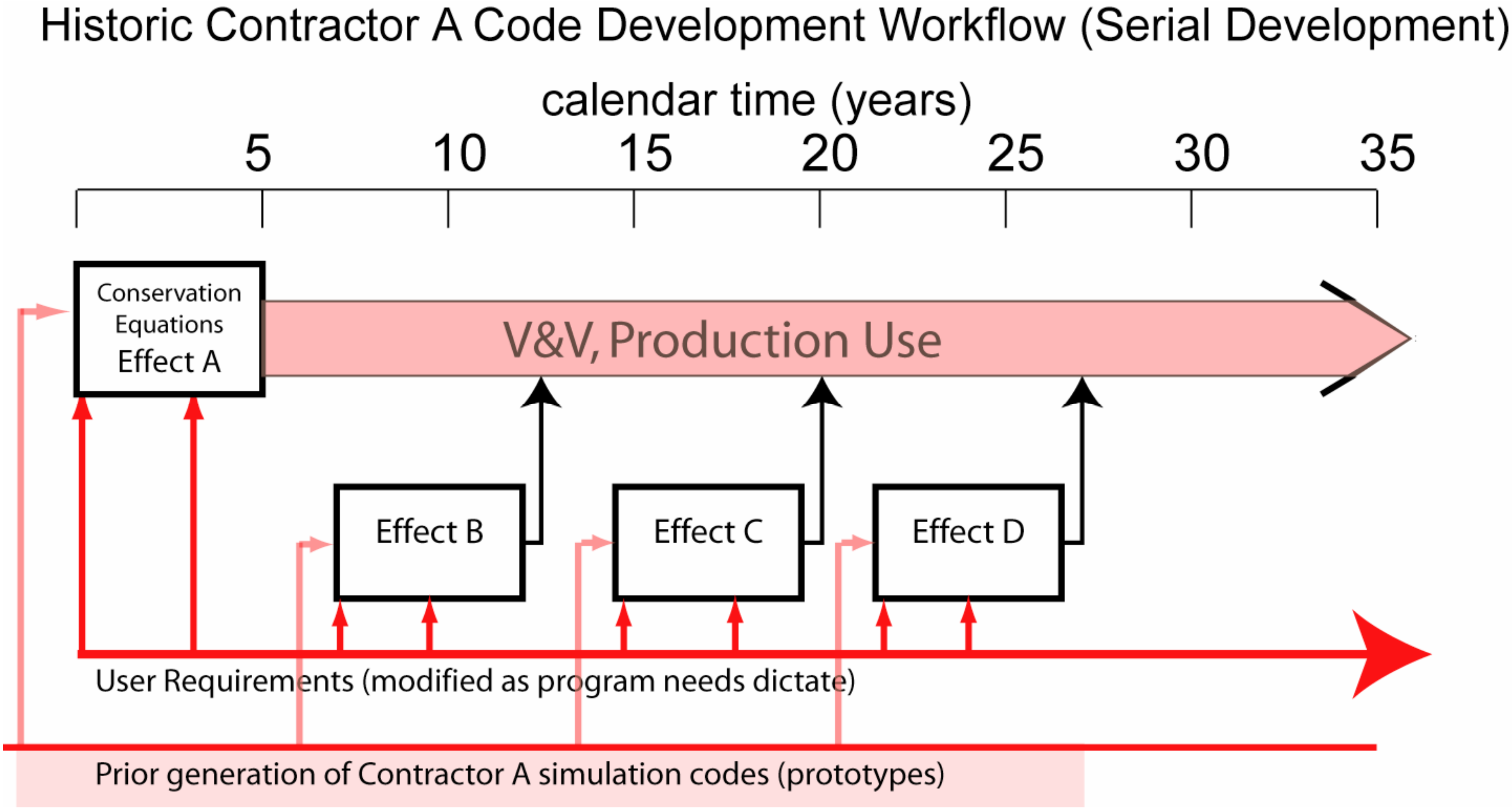
Code Development will be (is) the major bottleneck in the future (now).

- Codes need to scale to many, many thousands of processors.
- Low-hanging fruit has been gathered (porting of serial codes to parallel computers).
- Exciting opportunities to remedy present deficiencies:
  - Better spatial and temporal resolution
  - More accurate models
  - Inclusion of a more complete set of effects
    - Strongly-coupled, multi-scale effects
  - Codes that can model a whole system
  - Codes that can get answers in minutes to hours rather than days to weeks to months
- The greatest opportunities include integrated codes that couple many multi-scale effects to model a complete system.
- Success often requires large (10 to 30 professionals), multi-disciplinary, multi-institutional teams and 5 to 10 years of development time.
- It’s exciting, it’s challenging and it’s risky.

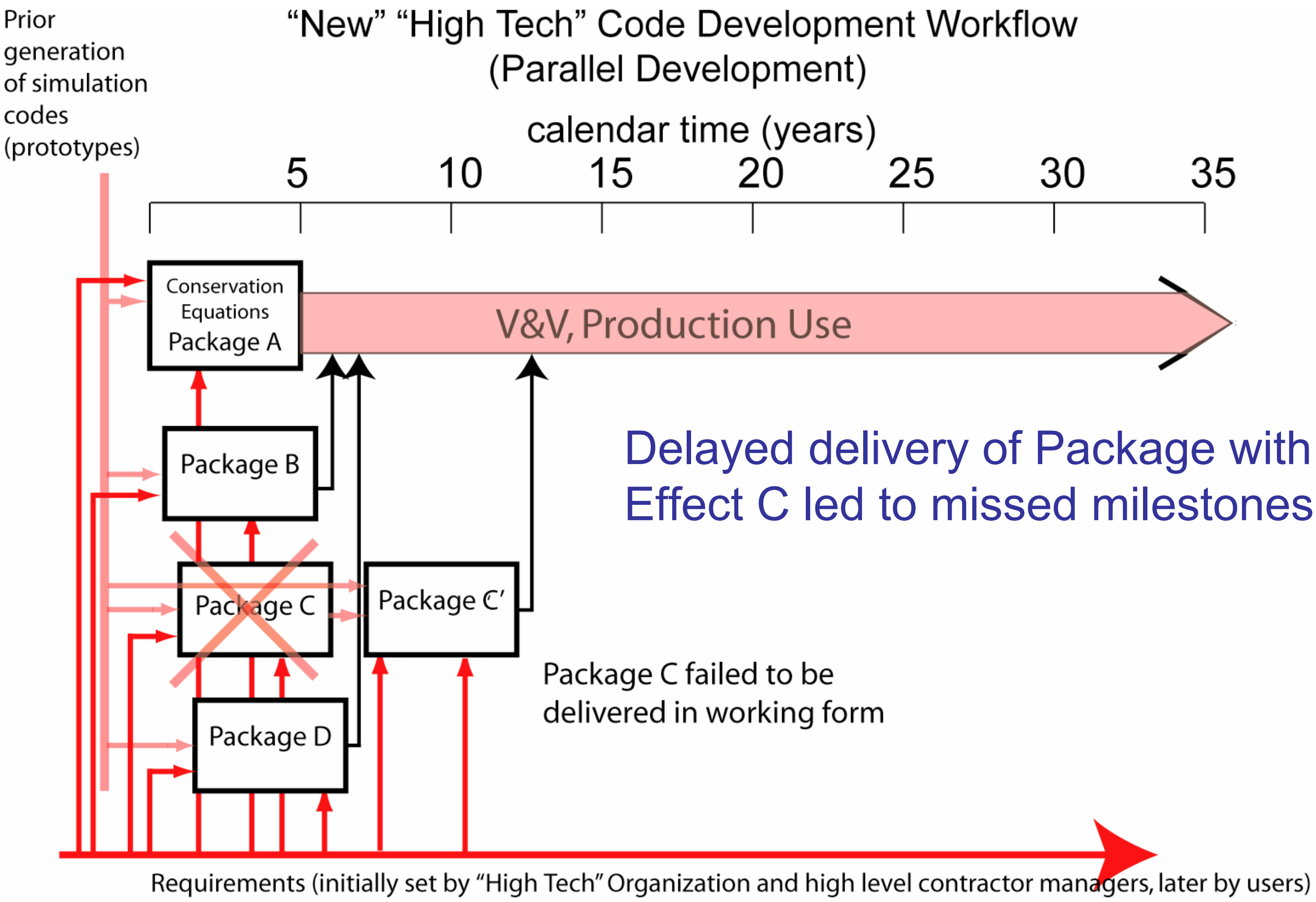
Predictive Risk is even more serious than Programming Risk.

- Programming Risk is a matter of efficiency
  - Programming for more complex computers takes longer and is more difficult and takes more people, but with enough resources and time, codes can be written to run on these computers.
- But the Predictive Risk is a matter of survival:
  - If the results of the complicated application codes cannot be believed or if the right codes are not developed and used effectively, then there is no reason to support the development and deployment of platforms or supporting software.
  - Pretty pictures are not necessarily consistent with the laws of nature!
  - Computational scientists and engineers have to be aware of all the issues:
    - Development of the application codes takes time and resources, often tens of people for tens of years plus resources for validation and testing and productions runs.
    - If the right codes are never developed, they cannot be used to solve problems.
    - If they are developed and give wrong answers, they cannot be used to solve problems.
    - If they are developed and not utilized effectively to solve problems, then the problems won’t be solved.

Proto-FALCON Workflows were initially serial



Ambitious schedule required parallel development with no contingency.



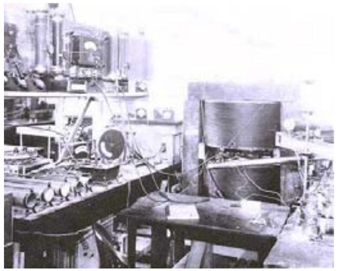
Package C failed to be delivered in working form

Delayed delivery of Package with Effect C led to missed milestones.

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

CERN 2000

We studied 6 federal agency projects to identify the *"Lessons Learned*"*

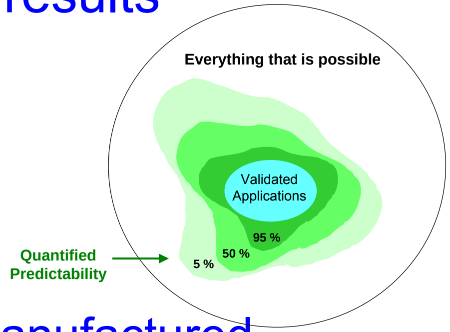
The Successful projects emphasized:

- Conservative approach - Minimize Risks!
  - Building on successful code development history and prototypes
  - Better physics and computational mathematics over better "computer science"
  - The use of proven Software Engineering rather than new Computer Science
    - Don't let the code project become a Computer Science research project!
- Sound Software Project Management - Plan and Organize the Work!
  - Highly competent and motivated people in a good team
  - Development of the team
  - Software Project Management: Run the code project like a project
  - Determining the Schedule and resources from the requirements
  - Identifying, managing and mitigating risks
  - Focusing on the customer
    - For code teams and for stakeholder support
  - Software Quality Engineering: Best Practices rather than Processes
- Verification and Validation – Correct Results are Essential!
  - Need for improved V&V methods became very apparent

The unsuccessful projects didn't emphasize these!

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

Verification and Validation



- Customers want to know why they should believe code results
- Codes are only a model of reality
- Verification and Validation are essential
- Verification
  - Verify equations are solved correctly
  - Regression suites of test problems, convergence tests, manufactured solutions, analytic test problems, code comparisons and benchmarks
- Validation
  - Ensure models reflect nature, check code results with experimental data
  - Specific validation experiments are required
    - Federal sponsor is funding multi-billion dollar validation experiments for V&V,…
- V&V experience with thes and other codes indicates that a stronger intellectual basis is needed for V&V
- More intense efforts are needed in both types of V&V if computational science is to be credible

Roach, 1998; Roache, 2002; Salari and Knupp, 2000; Lindl, 1998; Lewis, 1992; Laughliin, 2002)

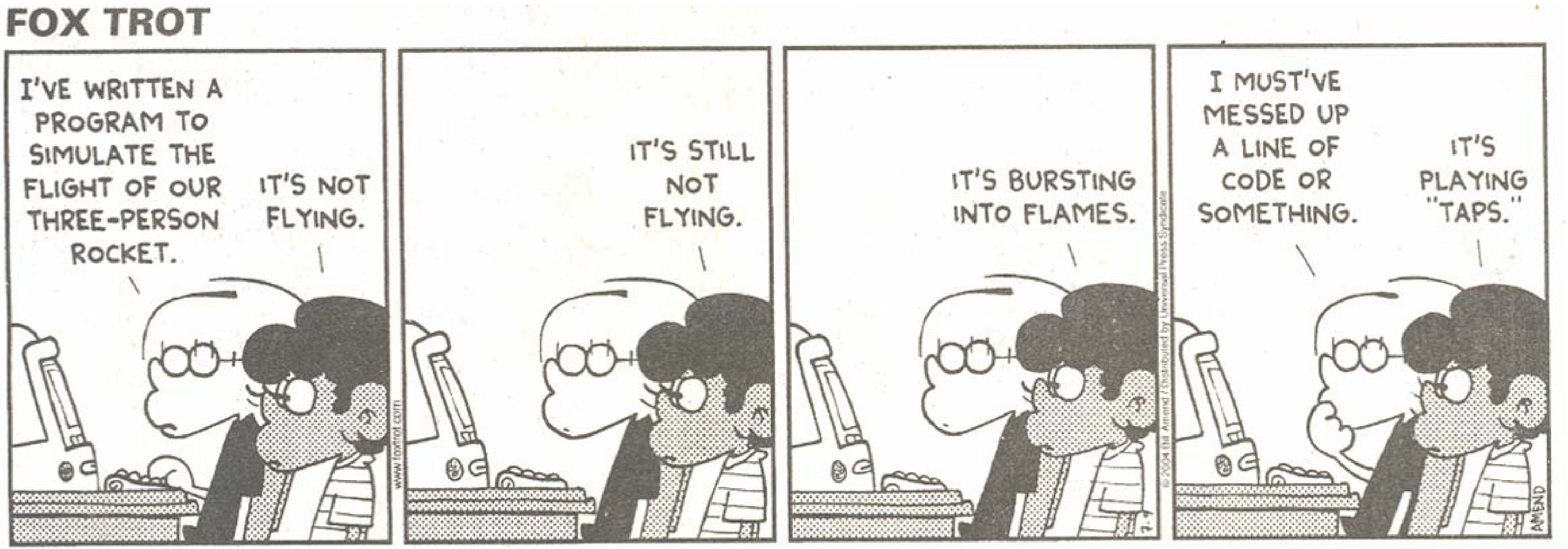
Many things can be wrong with a computer generated prediction.

- Experimental and theoretical science are mature methodologies but computational science is not.
- Hatton study* indicates that Scientific codes have ~ 6 defects per 1000 lines of code.
- Code could have bugs in either the models or the solution methods that result in answers that are incorrect.
  - e.g. 2+2=54.22, sin(90^{O})= 673.44, etc.
- Models in the code could be incomplete or not applicable to problem or have wrong data.
  - E.g. climate models without an ocean current model.
- User could be inexperienced, not know how to use the code correctly.
  - CRATER analysis of Columbia Shuttle damage.
- Many examples: Columbia Space Shuttle, Sonoluminescence, Fusion

*Hatton, L. and A. Roberts (1994). "How Accurate is Scientific Software?" IEEE Transactions in Software Engineering **20**(10): 785-797.

It’s risky. Software failures are not just in the IT industry.

- While software failures are commonly acknowledged in the IT industry*, not much is heard about them in the technical HPC community.
- But they exist.



*Ewusi-Mensah, K., *Software Development Failures: Anatomy of Abandoned Projects*. 2003, Cambridge, Massachusetts: MIT Press: Glass, R.L., *Software Runaways: Monumental Software Disasters*. 1998, New York: Prentice Hall PTR.

Technical Software Failures Continue to be in the news!

Jan., 1997 IEEE Computer

Software errors crash Ariane launch.

Object Technology

Editor: Bertrand Meyer, EiffelSoft, 270 Storke Rd., Ste. 7, Goleta, CA 93117; voice (805) 685-6869; ot-column@eiffel.com

Design by Contract: The Lessons of Ariane

Jean-Marc Jézéquel, IRISA/CNRS
Bertrand Meyer, EiffelSoft

Several contributions to this department have emphasized the importance of *design by contract* to the construction of reliable software. Design by contract, as

made up of respected experts from major European countries, which produced a report in hardly more than a month. These agencies are to be commended for the speed and openness with which they han-

several hours (at least in earlier versions of Ariane), it was better to let the computation proceed than to stop it and then have to restart it if liftoff was delayed. So the SRI computation continues for 50 seconds after the start of flight mode—well into the flight period. After takeoff, of course, this computation is useless. In the Ariane 5 flight, however, it caused an exception, which was not caught and—boom.

The exception was due to a floating-point error during a conversion from a 64-bit floating-point value, representing the flight's "horizontal bias," to a 16-bit signed integer: In other words, the value that was converted was greater than what can be represented as a 16-bit signed integer. There was no explicit exception handler to catch the exception, so it followed the usual fate of uncaught exceptions and crashed the entire software, hence the onboard computers, hence the mission.

This is the kind of trivial error that we are all familiar with (raise your hand if you have never done anything of this sort), although fortunately the consequences are usually less expensive. How in the world

How in the world could such a trivial error have remained undetected and cause

Nov.2004 IEEE Spectrum

Software failure takes LA FAA controllers off the air.

Jan., 2005 ComputerWorld

"FBI trying to salvage $170M software package"

Nov.25,2004 Economist

Computer codes not delivering!

Economist.com SCIENCE & TECHNOLOGY

The software-development industry

Managing complexity

Nov 25th 2004
From The Economist print edition

Most software projects fail to meet their goals. Can this be fixed by giving developers better tools?

ON SEPTEMBER 14th, the radios in an air-traffic control centre in Palmdale, California shut down, grounding hundreds of flights in southern California and Nevada, and leading to five mid-air encounters between aircraft unable to talk to the ground controllers. Disaster was averted because aircraft managed to communicate with more distant back-up facilities. But why did Palmdale's radios fail? A glitch in the software running the system meant the computers had to be re-booted every 30 days, and somebody forgot to do so. But software running a mission-critical system should not have to be restarted every month. The culprit: poor design.

NEWS

PLANE VIEW: An air traffic controller monitors activity on a radar screen—on 14 September, controllers in the Los Angeles area lost all radio communication with planes in the region, making it impossible to warn them directly of impending dangers.

Lost Radio Contact Leaves Pilots On Their Own

Communications error wreaks havoc in the Los Angeles air control system

It was an air traffic controller's worst nightmare. Without warning, on Tuesday, 14 September, at about 5 p.m. Pacific daylight time, air traffic controllers lost voice contact with 400 airplanes they were tracking over the southwestern United States. Planes started to head toward one another, something that occurs routinely under careful control of the air traffic controllers, who keep airplanes safely apart. But now the controllers had no way to redirect the planes' courses.

"You could see airplanes getting awfully close but you're powerless. You can do nothing about it," said Hamid Ghaffari, an air traffic controller at the Los Angeles Air Route Traffic Control Center in Palmdale, Calif., where the crisis occurred. The center is responsible for airplanes flying above 13 000 feet (4000 meters) in 460 000 square kilometers of airspace over Southern California and parts of Arizona, Nevada, and Utah, including the busy McCarran International Airport in Las Vegas, Nev.

The controllers lost contact with the planes when the main voice communications system shut down unexpectedly. To make matters worse, a backup system that was supposed to take over in such an event crashed within a minute after it was turned on. The outage disrupted about 800 flights across the country.

In at least five cases, according to reports in *The New York Times* and elsewhere, airplanes came within the minimum separation distances mandated by the U.S. Federal Aviation Administration for planes at high altitudes: five nautical miles (9.25 kilometers) horizontally or 2000 feet (610 meters) vertically. Fortunately, there were no collisions.

Although Ghaffari, who is also president of the National Air Traffic Controllers Association local, was not in the center when the system shut down, he was able later to watch the

DIGITAL VISION/GETTY IMAGES

16 IEEE Spectrum | November 2004 | NA

You may retrieve this story by entering QuickLink# 51982
> Return to story

FBI trying to salvage $170M software package

Its Virtual Case File project is under fire for not working as planned

News Story by Grant Gross

JANUARY 14, 2005

(IDG NEWS SERVICE) - The FBI will attempt to salvage parts of a $170 million case-management software package despite growing criticism that the 4-year-old project doesn't work as expected.

The software package, called Virtual Case File, was supposed to help FBI employees more quickly share data about cases in progress, including terrorism investigations, and to help FBI agents around the country better search documents and connect leads coming from diverse sources. But this week, FBI Director Robert Mueller told reporters that he was frustrated with the progress on the software package.

Science Applications International Corp. in San Diego last month delivered about one-tenth of what the FBI had envisioned for the package, and the company didn't incorporate many of the changes recommended by a second contractor about six months ago, an FBI official said today. SAIC has worked on Virtual Case File since June 2001.

"It wasn't to where we thought it would be," said the FBI official, who asked that her name not be used.

The FBI hasn't scrapped Virtual Case File and the workflow prototype SAIC delivered, but the agency has asked another contractor to look for commercial or government off-the-shelf software that could be used instead, the FBI official said. The FBI has also begun testing the workflow package to see if "there's anything we can use in developing a future case management program," she added.

SAIC, in a statement, said it delivered the first phase of the project ahead of schedule and under budget after the terms of the contract were renegotiated. The Sept. 11, 2001, terrorist attacks on the U.S. forced changes in the project and contributed to delays, the company said. The project

Perspective:
Requirements are important after all

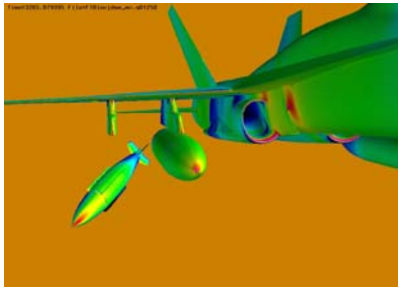
- Often said that computational science and engineering software doesn't have requirements in same sense as the IT industry
- Computational science and engineering does have highly rigid requirements
  - The laws of nature
- Computational science and engineering code development can't be planned in detail because it involves discovery of how to accurately simulate those laws

Perspective: Software Engineering and Computer Science are different and each is important

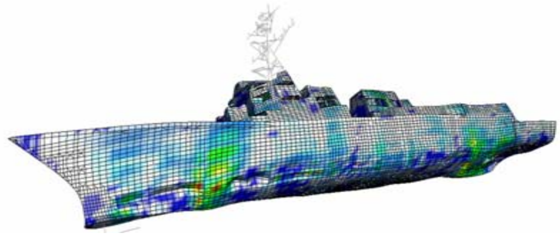
- Every successful code project needs software engineering, not computer science
- Software engineering involves the **implementation** of proven methodologies for code development
- Computer science involves **exploration, research and development** of new methodologies and concepts
- Computer science is an essential activity, but it should be an independent activity

CREATE Focuses on Design and Engineering for Acquisition

Goal—Make design and engineering a more effective contributor to acquisition



- CREATE will develop advanced computational engineering tools to optimize the design and testing of:
  1. Military aircraft (i.e., structures & aerodynamics)
  2. Naval vessels (i.e., structures & hydrodynamics)
  3. Integration of RF sensors and C4ISR antennas with platforms (i.e., electromagnetics & signatures)
- Each project: $10M/year for 10 years; total $300M
- Result:
  - Faster and more effective acquisition process
  - Better, faster and more effective design and validation
  - Fewer problems discovered in testing
  - Fewer costly delays and rework to fix flaws



Recap: What do you do you need to succeed?

Case studies* of existing computational science and engineering project indicate that increased emphasis is crucial for:

- Verification and Validation
  - Accurate, reliable results, are needed and not just pretty pictures!
- Software Project Management
  - Single investigator paradigm doesn’t work for large teams
  - Large teams need a project orientation to organize and coordinate the work
- Software Engineering
  - Software development is a highly technical 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.

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

Observations on Weather prediction

- Validation is a challenge
  - Few controlled experiments
- Who is the code architect? Where is the conceptual integrity? And who enforces it?
- All codes involve trade-offs between accuracy and time to problem completion.
  - I'm not sure that many weather/climate codes enforce the trade-off to ensure practical run times.
- Example: ASCI academic alliances:
  - Multi-physics codes, each module with the "best physics"
  - Result: Initially could only simulate 6 s of a 20 minute fire, 2 s of a 120 s rocket burn,…

Reductionism and Emergence

- Weather and climate codes include 100s of effects
  - Problem is reduced to its constituents
  - Answer depends on trade-off of many competing effects
- Robert Laughlin (Nobel Prize, 1999) and others have been pointing out that solving complex problems by calculating the trade-off of all of the detailed effects (reductionist) is an NP incomplete problem
- They claim that we only solve problems where there are a set of overarching or “emergent” principles (e.g. conservation laws, symmetry, thermodynamic principles,…)
  - We use hydrodynamics to calculate ocean flow, not molecular dynamics
- How can we sure that weather models correctly capture the relevant emergent principles?
  - Validation is the best way to ensure that the emergent principles are captured

The Future

- We live in “exciting times”
- CSE offers tremendous promise to address and solve important problems
  - The potential to tackle and solve problems that we couldn’t before now
- CSE faces many challenges just like every other new problem solving methodology has faced
- It will take time and a lot of hard work
- But if we face and overcome the challenges we can do great and important things