Mitigating the risks facing large-scale computational science.

Douglass E. Post

post@ieee.org

Chief Scientist:

DoD High Performance Computing Modernization Program

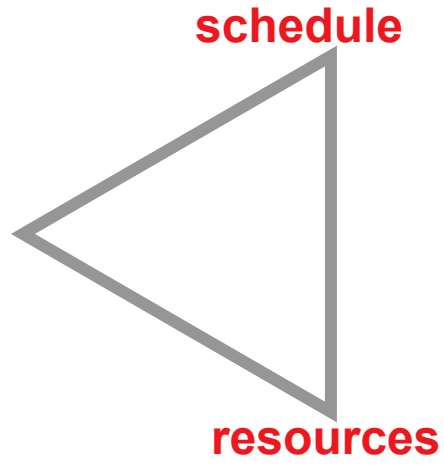
Acknowledgements: R.P.Kendall, J. Grosh, L.G.Votta

High Performance Embedded Software Conference

Lexington, MA Sept. 21, 2005

Computational Science and Engineering can transform DoD warfighting technologies.

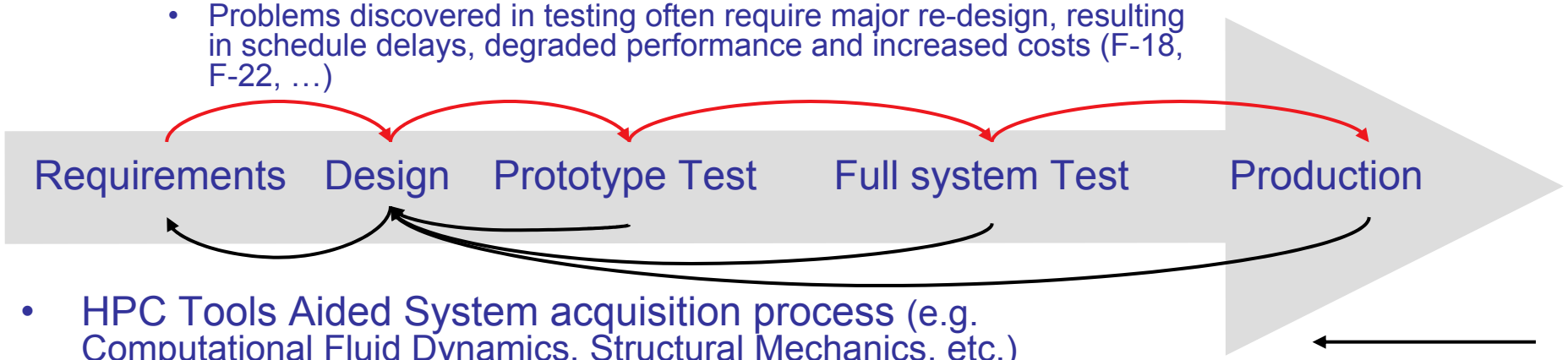
- To meet its evolving mission, the DoD must respond quickly to a rapidly changing world
- It must design and procure **better** weapons **faster and cheaper**
- A weapon project must have consistent **requirements**, **schedules** and **resources (the iron triangle)**
- Existing projects utilize:
  - Engineering design,
  - Theoretical analysis (including some computing) and
  - Experimental testing
- Breaking the iron triangle requires a new problem solving methodology
- Computational science and engineering using high performance computing offers the promise of such a new and very powerful methodology
- Definition: **A high performance computing application is one that exploits a significant fraction of the most powerful computers today.**



Computational Science and Engineering can improve the process.

- Conventional system acquisition process
  - Initial design developed with engineering tools
  - Prototypes built and tested (e.g.wind tunnel)
  - Full system built and tested (e.g. flight tests)
  - Full Production
    - Problems discovered in testing often require major re-design, resulting in schedule delays, degraded performance and increased costs (F-18, F-22, …)

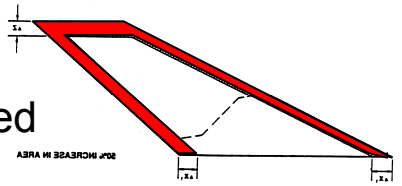
Ideal Plan



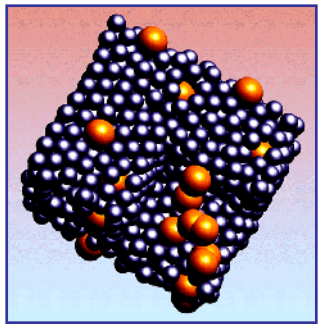
Reality

- HPC Tools Aided System acquisition process (e.g. Computational Fluid Dynamics, Structural Mechanics, etc.)
  - Initial design developed with aid of HPC tools
    - Improved design optimization, greater exploration of design options, edge-of-envelope operations
  - Prototypes built and tested (e.g.wind tunnel)
    - Testing process more effective
  - Full system built and tested (e.g. flight tests)
    - Fewer full system tests needed
  - Full Production
    - Fewer problems discovered in testing that require major re-design, fewer schedule delays, less performance degradation and lower costs

F-117
50% increase required
for tailfin

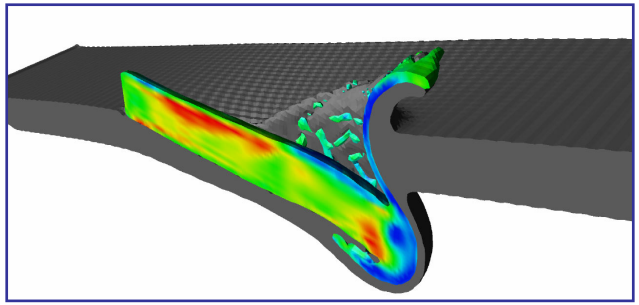


Important DoD Problems are being addressed with high performance computing applications.



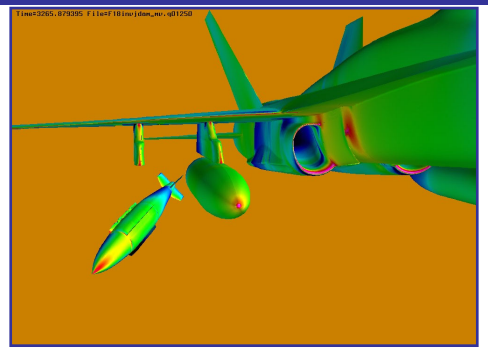
Basic Research

Simulating High-Energy Density Rocket Fuels



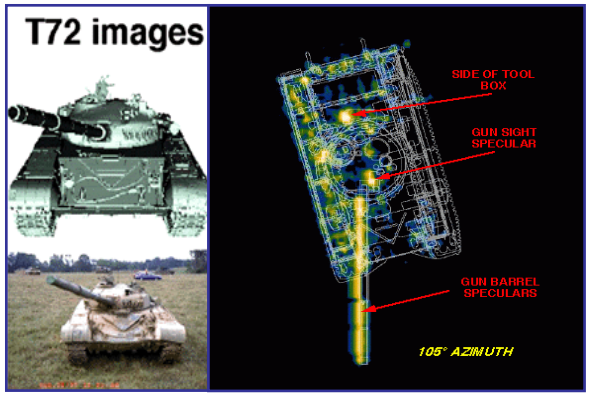
Advanced Technology

Armor and Projectile Design



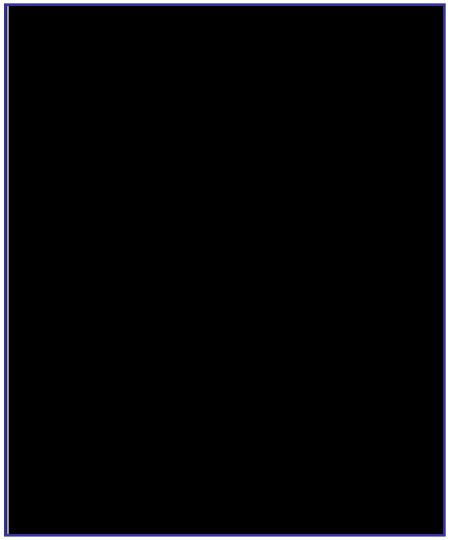
Developmental T&E

Support of Aircraft-Store Compatibility and Weapons Integration



Intelligence

Radar Cross-Sections Predictions



Operations

Ocean/wave forecasting

- J. Grosh

Next generation of computers can enable a “transformational change” in DoD design and testing methodologies to break the “iron triangle”

Estimated DARPA HPCS System and the largest HPCMP System assuming normal rates of technology improvement

Capability	Present HPCMP Systems (2006)	Evolving HPCMP Systems (2010)	DARPA HPCS System (2010)
PetaFlops/s (Linpack Top 500)	**0.02**	**0.2**	**4+**
Bandwidth (PetaBytes/s)	**0.0005**	**0.05**	**Up to 6**
Processor count	**4k**	**~10k**	**30-70k, up to 1M**
Memory PBytes	**0.008**	**~0.05**	**1-4**
Approximate measure of maximum increased capability (speed x bandwidth x memory)	**1**	**6,000**	**1,000,000,000**

- **We will be able to:**
  - **Achieve adequate spatial and temporal resolution**
  - **Develop and employ more accurate models**
  - **Include a more complete set of models**
  - **Model a complete system**
- **If we can meet the development challenge (DARPA HPCS emphasis!)**

Computational Science and Engineering has Four Major Elements.

Computers	Codes	V&V	Users
Making enormous progress but at cost of complexity	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

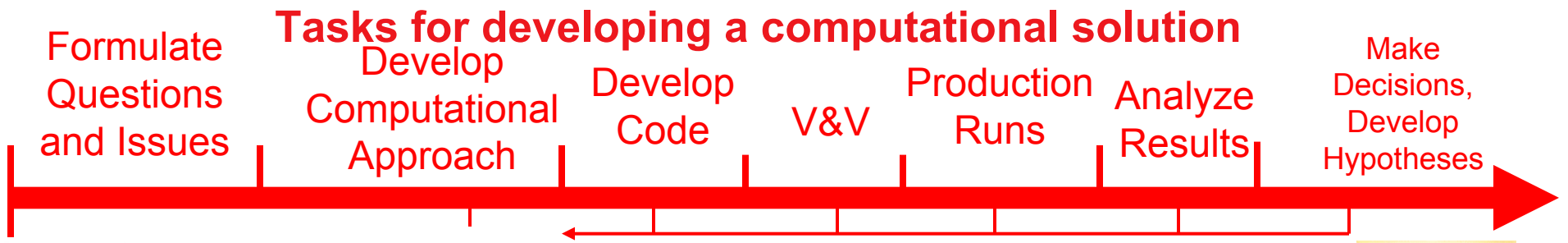
Sponsors

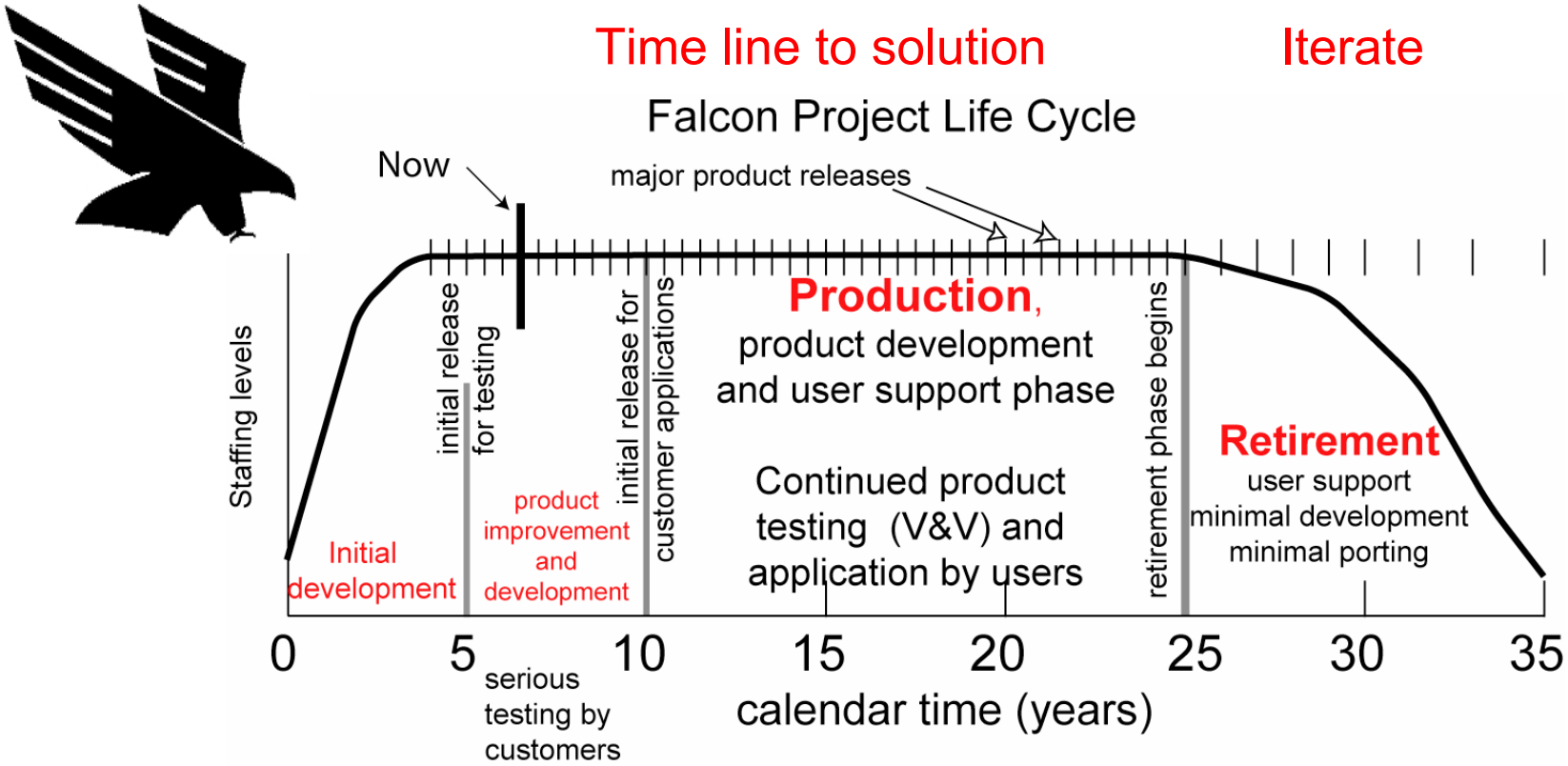
Code Development will be the major bottleneck in the future

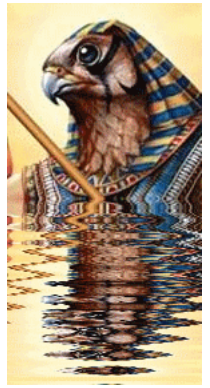
- Codes need to scale to many thousands of processors
- Low-hanging fruit has been gathered (porting of serial codes to parallel computers)
- Opportunities:
  - Better spatial and temporal resolution
  - More accurate models
  - Inclusion of a more complete set of effects
  - Codes that can address whole system
- Greatest opportunities are for integrated codes that couple many multi-scale effects to model a complete system
- Success requires large (10 to 30 professionals) teams and 5 to 10 years of development time

Requirements for computers and computer science strongly influenced by code project life cycle and workflows.

Case study of Falcon Code Project

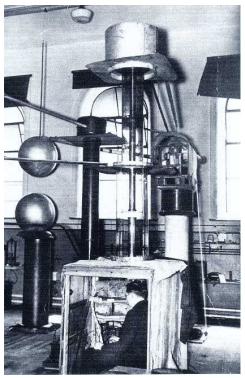




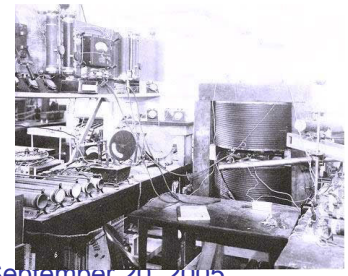


Computational Science making the same transitions that experimental science made in 1930 through 1960.

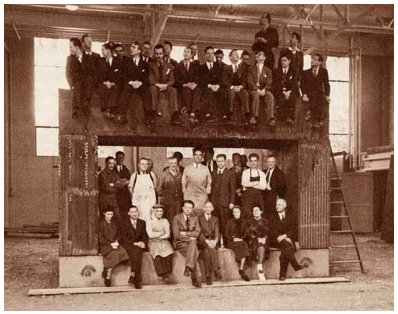
- Computational science moving from “few-effect” codes developed by small teams (1 to 3 scientists) to “many-effect” codes developed by larger teams (10, 20 or more).
- Analogous to transition that experimental science made in 1930-1960 time frame from small-scale science experiments involving a few scientists in small laboratories to “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

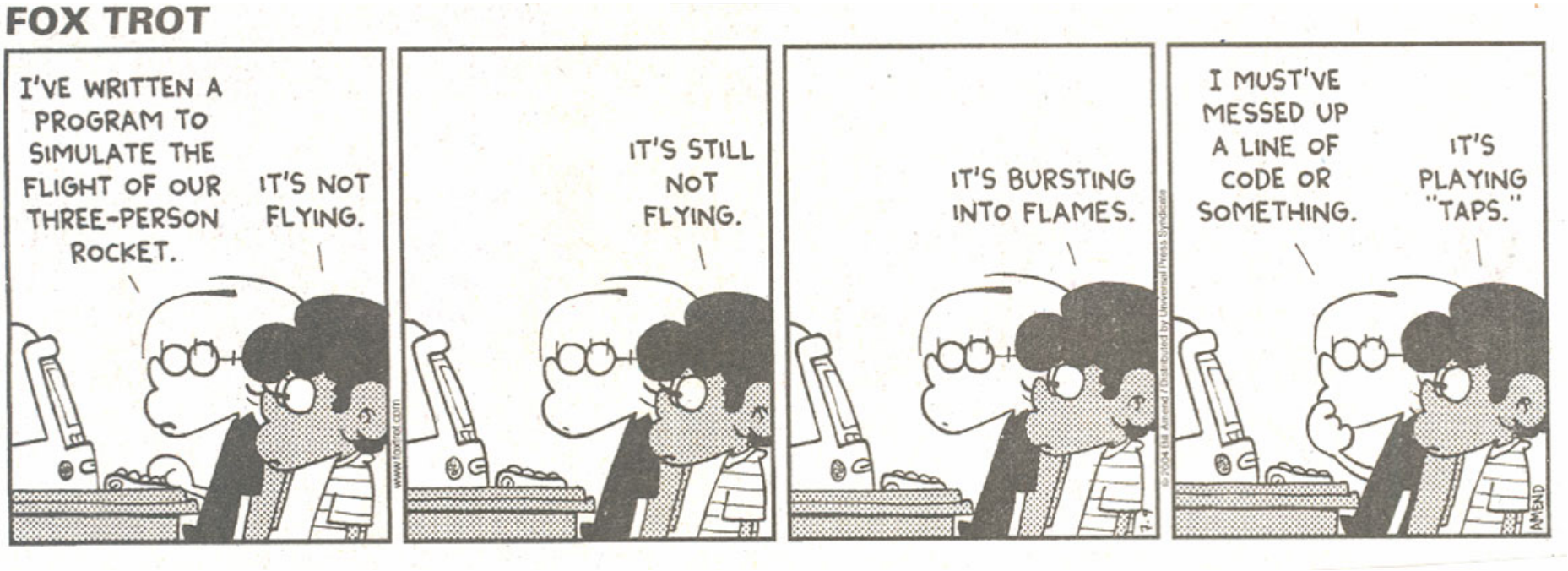


Late 1930’s



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

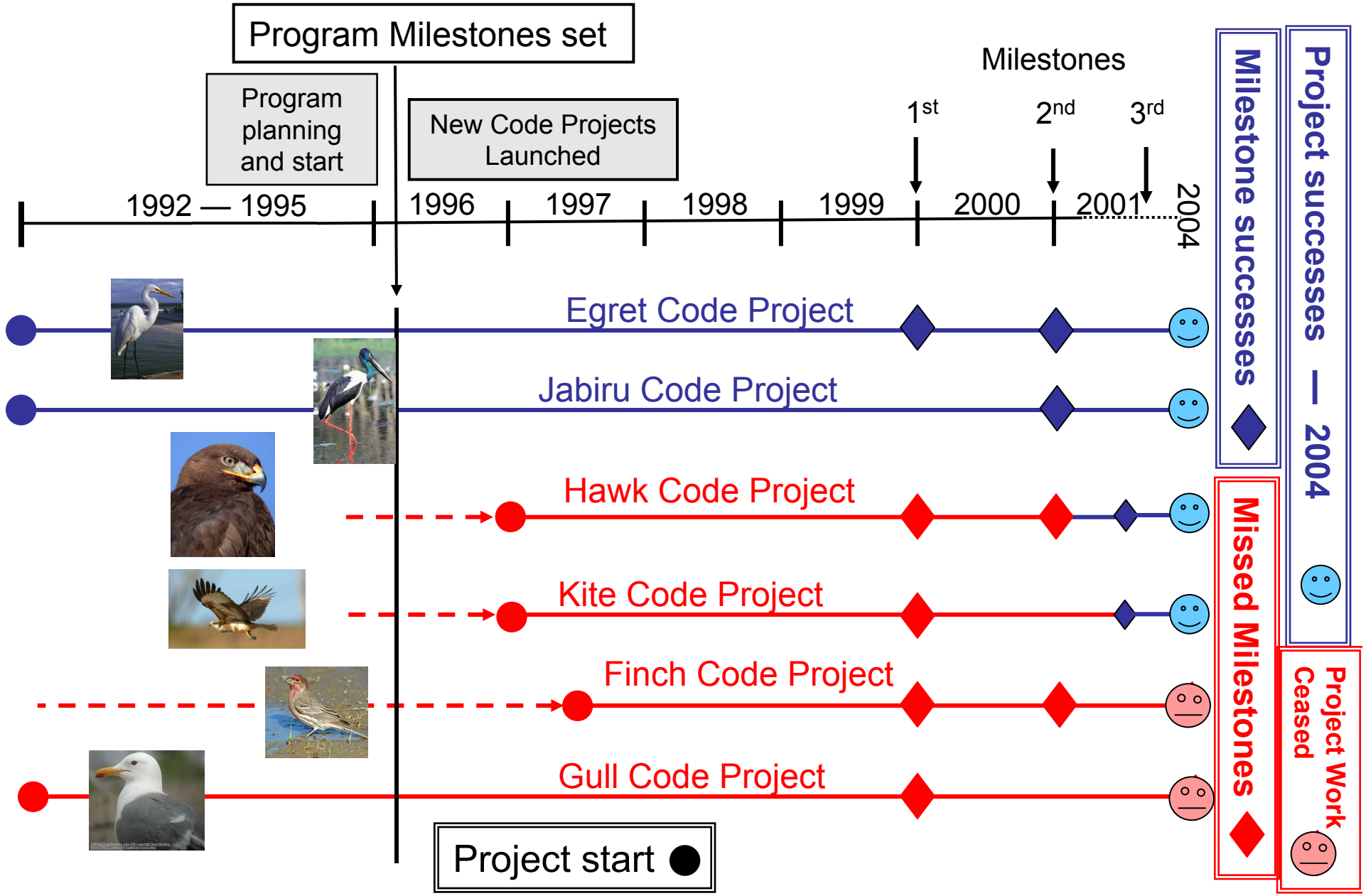
- While software failures are commonly acknowledged in the IT industry*, not much is heard about them in the technical HPC community.
- But they exist.



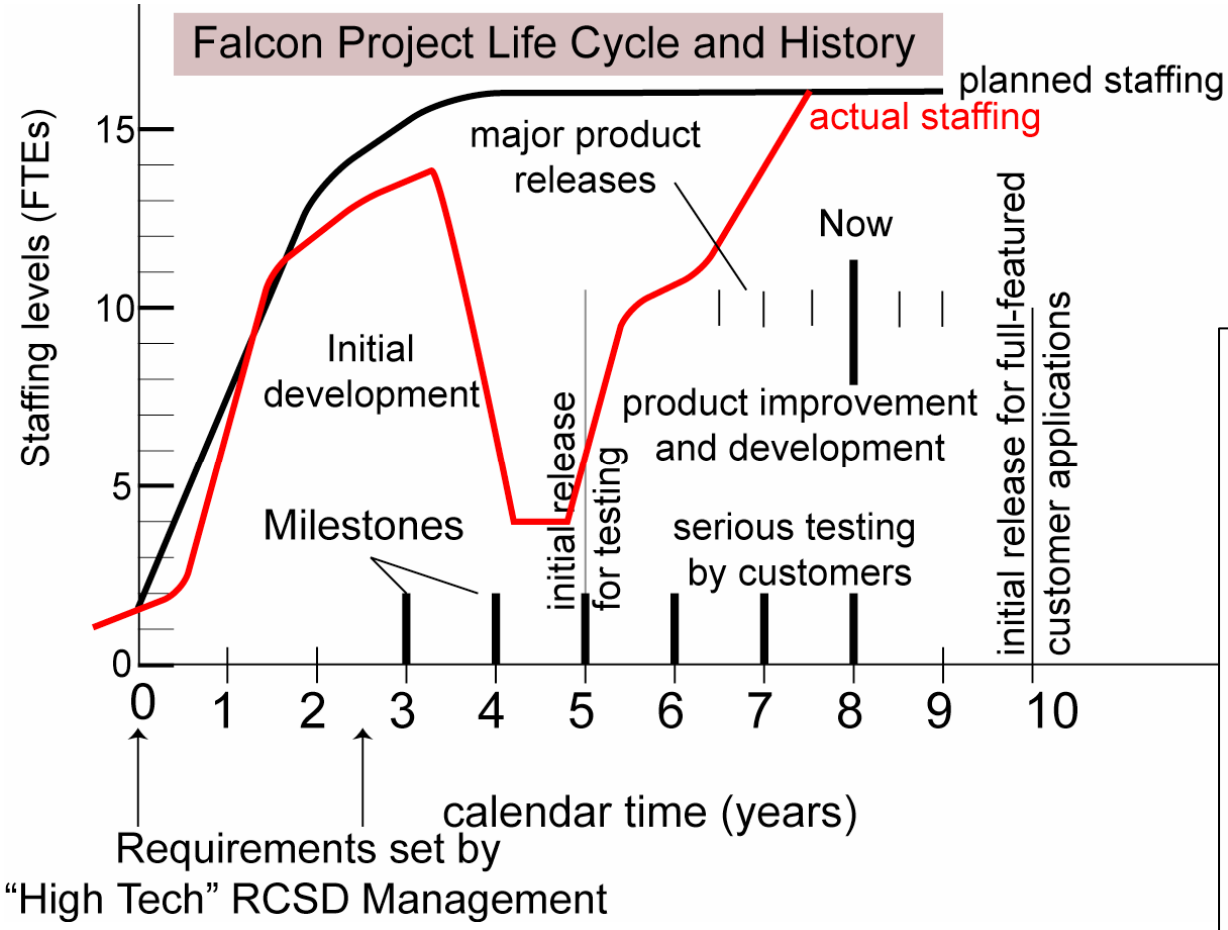
*Ewusi-Mensah, K., *Software Development Failures: Anatomy of Abandoned Projects*. 2003, Cambridge, Massachusetts: MIT Press: Glass, R.L., *Software Runaways: Monumental Software Disasters*. 1998, New York: Prentice Hall PTR.

Large scale code development is risky.

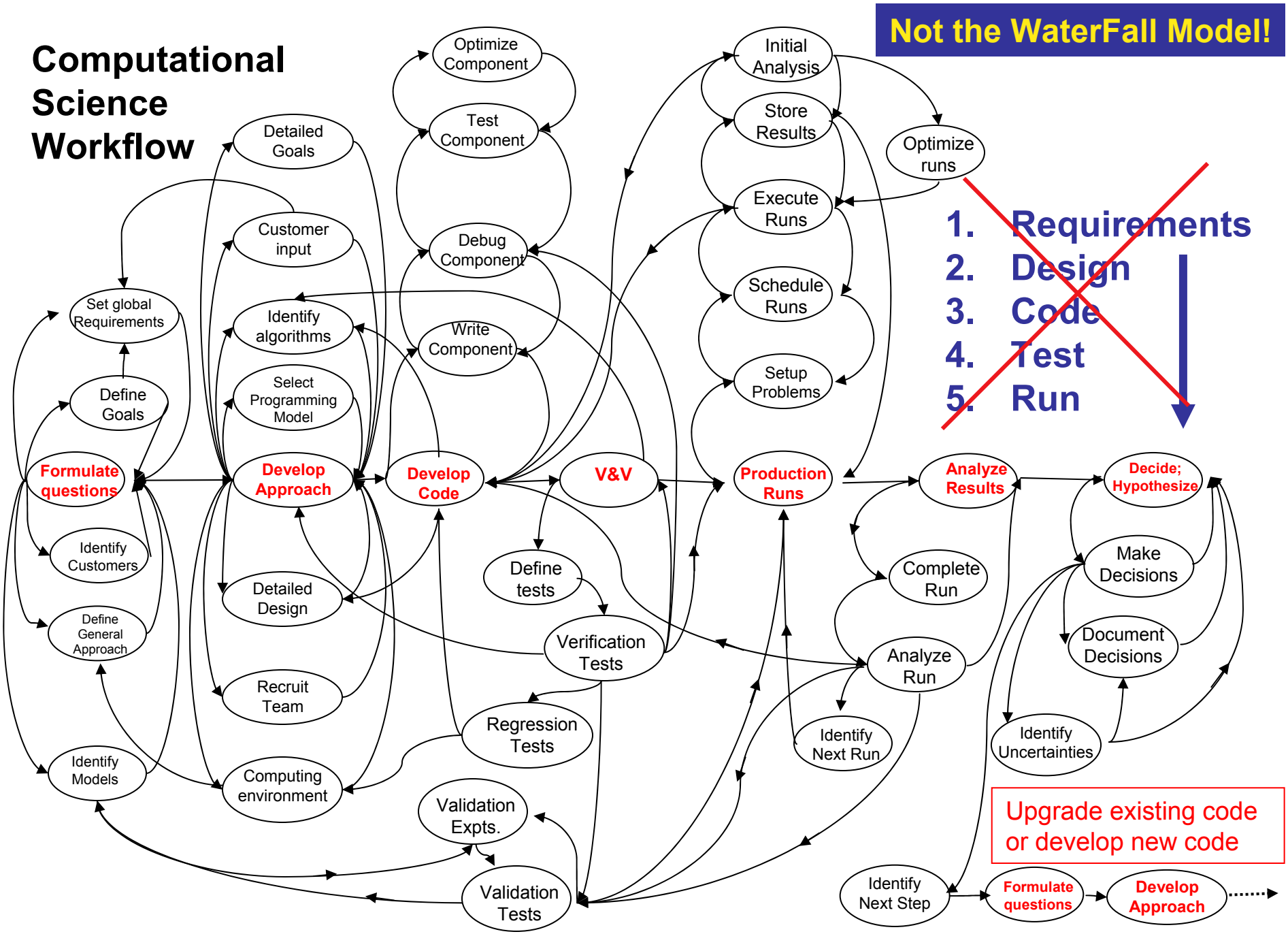
Six Large Code Project Schedule



Falcon project had turbulent beginning largely due to initial requirements.



Computational Science Workflow



How can we succeed? Case Studies point the way!

- 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.

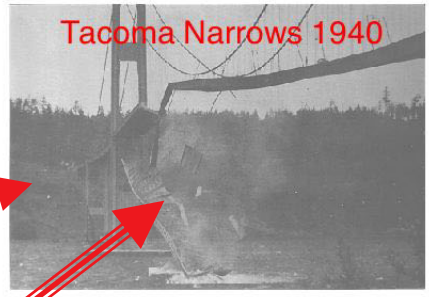
Tacoma Narrows Bridge buckled and fell 4 months after construction!

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

1

2

3



Lessons Learned
Case Studies

4

time

Comparative case study of six projects with the same goals and resources identified the *"Lessons Learned*"*

The projects that were successful emphasized:

- Minimizing risks
  - Build on successful code development history and prototypes .
  - Invest in better physics and computational mathematics before better computer science.
  - Use modern software engineering and computer science methods; and, do not do computer science research in a large code project—adds too much risk.
- Sound Software Project Management.
  - 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
  - Need for improved V&V methods became very apparent.

*D.E.Post and R.P.Kendall, International Journal of High Performance Computing, **18**(2004), pp.399-416.

The projects and their institutions that were unsuccessful didn't emphasize these sufficiently!

Verification and Validation

- Customers want to know why they should believe code results.
- Without adequate V&V, they shouldn't believe the code results.
- Codes are not reality, only a model of reality.
- 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.
    - The agency that funded the Falcon project is funding a large experimental program to provide validation data.
- Our case studies indicate that a stronger intellectual basis is needed for V&V.
- More investment is needed in Verification and Validation if computational science is to be economical and credible.
- **DoD testing facilities well suited for validation**.

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

Summary Conclusions

Challenge	Goal (and risks)	Roadblocks	Status
Performance	Powerful Computers	Limits on power, memory latency, ….	Successful but at cost of complexity
Codes	Build fast, accurate codes that can address the important problems	Complexity of computers and difficulty of science means that rapid development and accurate integration takes a large team and many years.	Due to technical challenges and long development schedules, not enough codes are being developed.
Production	Engineers and scientists use the code to solve problems	Whole system (computer, code, V&V, production system) must work	Limited by available codes and by computer complexity.
Senior Leadership	Sponsor initiates effort to solve strategic problem	Requires foresight, vision, patience, and risk.	Few sponsors are supporting code development.

There is a path forward to realize this opportunity.

- The computer industry, with help from DARPA HPCS and market forces, is continuing to develop and deliver increasingly more powerful computers.
- The computer industry, **partially due to DARPA HPCS emphasis on productivity,** is beginning to recognize the necessity of making it easier to develop and run codes, but much remains to be done.
- The scientific and engineering community needs to identify the opportunities for high performance computer applications to solve strategic problems and successfully make a case to prospective sponsors that computational applications can make a unique contribution toward solving strategic problems.
- The sponsors need to provide the resources to develop the codes, buy and support the computers, and support the V&V and application of codes.
- The code development community must utilize their experience (both individual and community from case studies) and domain knowledge to develop the needed tools.
- Users and developers must verify and validate the codes and then employ the codes to solve strategic problems.

100 processors

100 ft. rocks
Sea level

10^5 processors

5000 ft. cliffs
el.15,000 ft.

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*.

Summary

- If Computational Science is to fulfill its promise for society, it must become as mature as theoretical and experimental methodologies.
- **Preformance Risk**
  - Being met, but at expense of complexity which leads to increased programming and prediction risk
- **Programming Risk**
  - HPC community needs to reduce the difficulty of developing codes for modern platforms—DARPA HPCS developing new benchmarks, performance measurement methodologies, encouraging new development tools, etc.
- **Prediction Risk**
  - Mitigation requires learning from past experiences, successes and failures, develop "lessons learned" and implement them—DARPA HPCS doing case studies of ~ 20 major US code projects (DoD, DOE, NASA, NOAA, academia, industry,…)
  - Major lesson is that we need to improve:
    - Verification
    - Validation
    - Software Project Management and Software Quality

For papers and talks send email to post@ieee.org