On Developing Effective Software Engineering Approaches to Resolving Scientific Computing's Productivity Gridlock

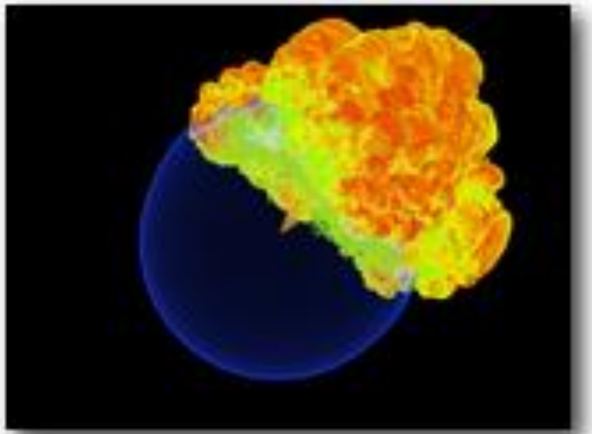
Stuart Faulk, Ph.D.
Computer and Information Science
University of Oregon

Outline

- Challenges of Scientific Computing (SC)
- SC's growing productivity problems
- Productivity studies and root causes
- Implications of the "expertise gap"
- Can SE help? Prerequisites to successful collaboration
- R&D areas where SE could contribute

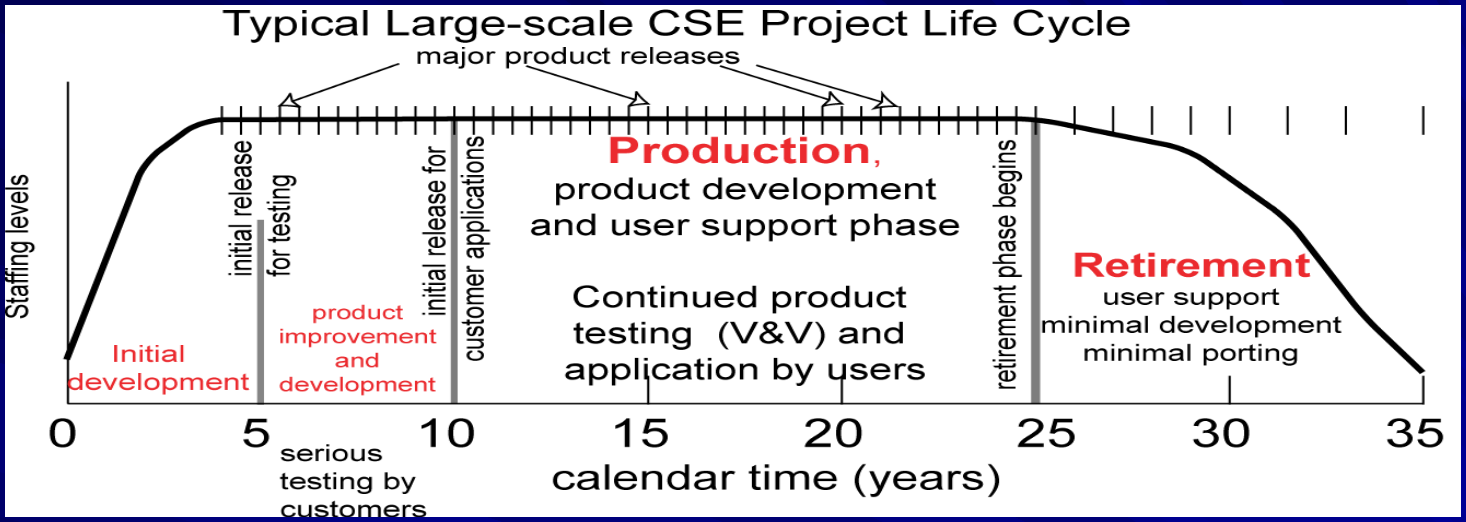
Focus on SC Community Codes

- Growing demand for “Virtual Research and Test” facilities
  - Applications providing simulation, analysis and test capabilities for science and engineering
  - Execute on large, massively-parallel platforms
    - Accurate simulation of complex physics
    - Analysis of big data, real-time results
  - Materials, fluid dynamics, climate, weather, etc.
- Shifting paradigm
  - 1960s->today: codes for and by subject-matter experts
  - 1990s->today> : codes for external expert community
- SC codes for community use present greatest challenges going forward



Thermonuclear flame plume bursting through the surface of a white dwarf: (FLASH multi-physics sim)

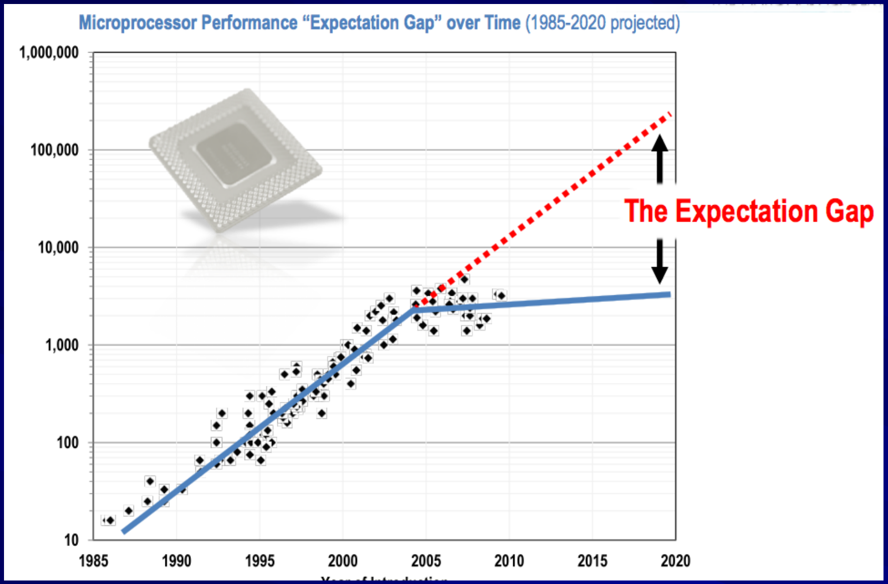
SC Distinguishing Characteristic



- Driven by the science (not software qualities)
  - Time-to-solution
  - Validity (V&V are expensive)
  - Agility (Emerging/changing requirements)
  - Performance really matters (~50% dev. effort)
- But it's not the only important quality
  - Very long life cycle (maintainability/mutability)
  - Ports are frequent (portability)

Post & Kendall, *Large-Scale Comp. Scientific and Engi. Project Development and Production Workflows*, CTWatch, 2007

The Challenge of Computer Complexity



- Machine complexity is increasing
  - Clock-speed is stuck while circuit density increases
  - Future of increasing parallelism, more special purpose processors

National Academy Study: The Future of Computing Performance (2010)

- Coding is correspondingly more difficult
  - Scaling and optimizing to massive parallelism
  - Achieving and demonstrating correctness
  - Maintaining, porting
- Observed SC development problems
  - Increasingly long and expensive development
  - Higher risk of failure
  - Growing maintenance costs
  - Increasing difficulty of porting to next generation machines

The SC Productivity Problem

- Assert that SC has a growing problem in *end-to-end productivity*
- Informal definition: use “productivity” to denote the (scientific) value produced per unit cost over time
- Using current development paradigms, it is increasingly difficult to
  - Get correct scientific solutions onto target hardware
  - Effectively exploit new hardware capabilities
  - Continue meeting evolving user needs over the life cycle
- Community concerns
  - *Large scale development of new application codes and refactoring of existing scientific numerical software are emerging as major obstacles to the effective use of extremescale computing systems* [NdousseFetter 2014]
  - *Current methods by which HPC systems are programmed and data are extracted are not expected to survive into the exascale* [DoE ASCR Workshop Report 2011]

Studies of Root Causes

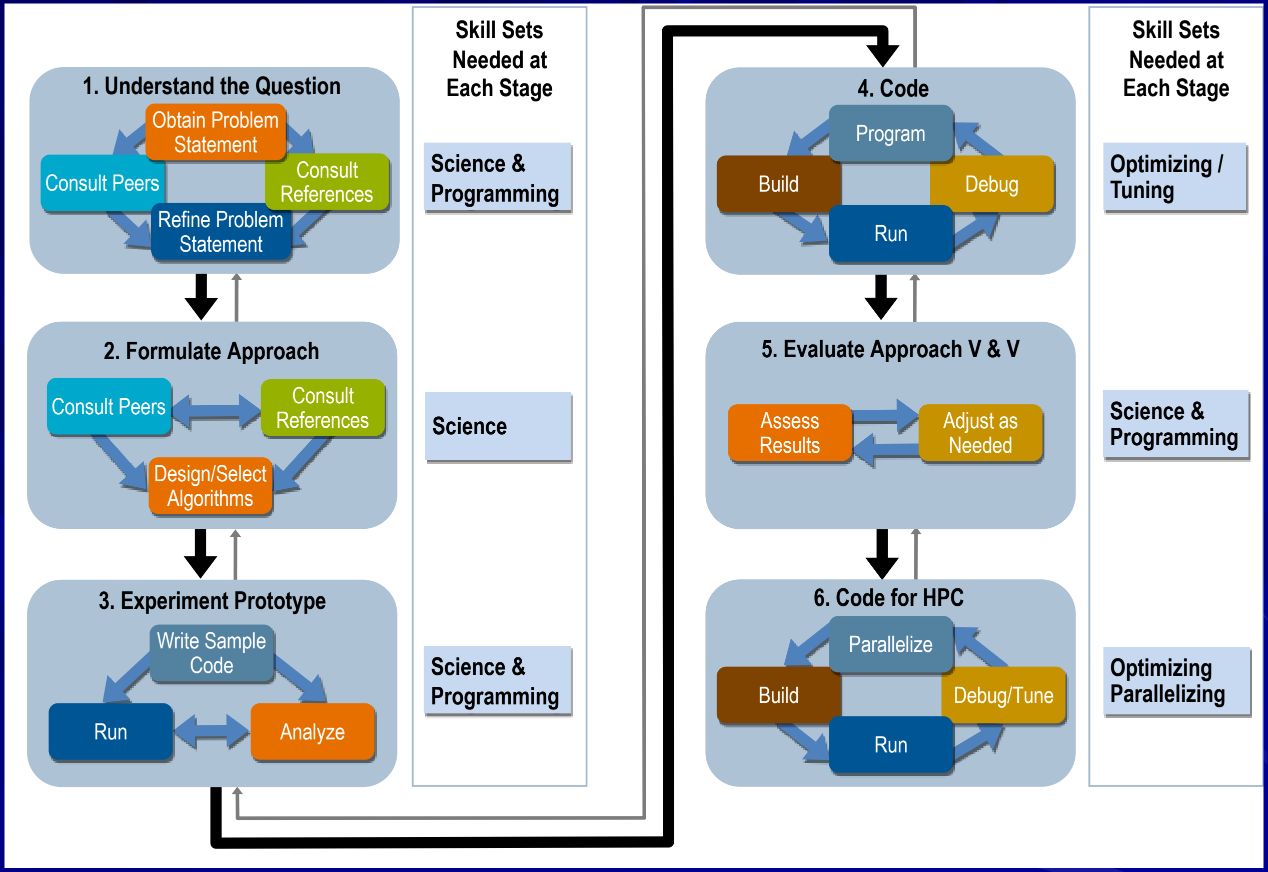
- Empirical studies aimed at understanding the source and nature of SC productivity issues
- Six case studies in DoE Accelerated Strategic Computing Initiative (ASCI) conducted at National Labs
  - Large, complex systems on massively parallel platforms
  - Extract lessons that could increase success rate
- Five case studies under DARPA HPCS program
  - Range of applications and scales (public &commercial)
  - Identify technical and organizational challenges
  - Identify lessons learned in tools and SE processes
- Sun empirical studies of SC coding and development workflows (DARPA HPCS)

Sun Workflow Studies

- Goal: understanding where current SC development practices limit end-to-end productivity
  - Interdisciplinary team from social, physical and computational sciences
  - Collected empirical data validated by multiple approaches
    - Case studies, interviews, focus groups
    - In-situ observations of developers (Hackystat)
    - Experimental studies: controlled developments, measurements
- Developed an canonical HPC workflow model
  - Identify tasks consuming the greatest resources
  - *Skill sets required for those tasks

HPC Workflow Bottlenecks

- Most resource intensive tasks
  - Developing correct scientific programs
  - Serial optimization and tuning
  - Code parallelization and organization (scaling)
  - Porting and modifying existing parallel code
- Bottlenecks result from:
  - Manual methods
    - Hand coding, scaling, optimization, verification
  - Multidisciplinary expertise
    - Most tasks demand multiple skill sets
    - Domain science, programming, parallelization, and target hardware



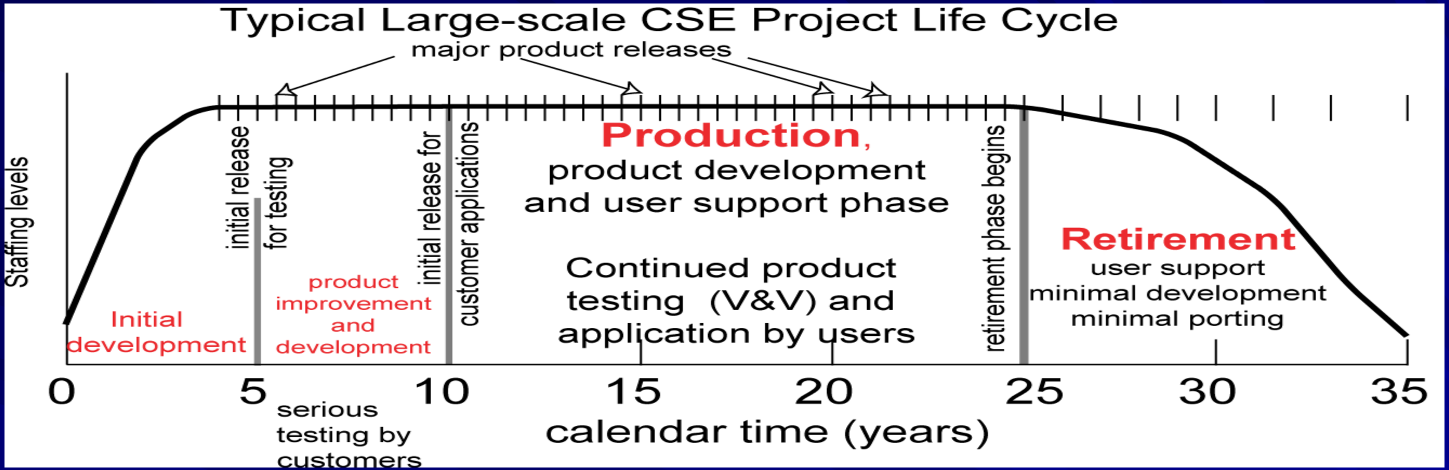
Finding: the *Expertise Gap*

- Bottom line: productivity depends on multidisciplinary experts optimizing parallel code by hand
- Key finding: there exists an *expertise gap* at the heart of the productivity crisis
  - Vanishingly few individuals with needed skills for a given scientific domain, language, and hardware set
  - Training (apprenticeship) takes years
  - Once acquired, are often not portable
- and it will only get worse…
  - Demand is growing
  - More demanding as hardware becomes more complex

Finding: Inadequate SE Methods and Tools

- SC code development is dominated by informal processes and manual methods
  - Developers are scientists first not software engineers
  - Processes largely ad hoc
  - Use of high-level languages is low
  - Limited use of current SE methods
- Tool support fragmentary*
  - Often ad hoc collections
  - Little support for most labor-intensive tasks
  - Scaling, optimization, and other critical tasks largely manual
- Upshot: Process and product quality depend on individual skills and efforts

Inadequate SE Methods and Tools



- Essentially complex and demanding development
  - Mission critical, large, long-lived, multidisciplinary, complex
  - Agility, validity, precision, performance, time-to-solution, reliability, maintainability, portability, cost, customer satisfaction
- Must concurrently satisfy conflicting goals
  - Time-to-solution (expedience) vs. maintainability (robustness)
  - Performance (machine dependent) vs. portability (machine independent)
  - Emerging requirements (agility) vs. correctness (extensive V&V)
  - But all must be addressed concurrently and at massive scale

Why not adopt current SE methods?

- Perception that "computer scientists don't address our needs"
- SE processes, methods and tools not adapted to SC's unique goals and constraints
  - Languages, methods, etc. focused on serial coding
  - Typically abstract from critical hardware properties including utilization and performance
  - Processes don't address SC's conflicting development goals
- Adoption of untried SE methods viewed as adding risk
  - Adaptation cost for is high, effort is a distraction from the science, benefits are uncertain
  - Confirmed by experience

Can SE contribute? Yes but …

- SC desperately needs new methods
  - Bottlenecks are inherent in hand-crafted paradigm
  - Cannot produce multidisciplinary experts fast enough
  - *Productivity gridlock*: resulting inability to start solving productivity problems, even as overall productivity declines
- But, SE must address the realities of SC development
  - Design processes and methods to encompass the unique SC life cycle
  - Embrace parallelism, performance as fundamental
  - Directly address tradeoffs in SC's design-time and run-time goals
  - Demonstrate effectiveness in realistic environments

R&D Areas: The Expertise Gap

- Must reduce dependence on multidisciplinary experts to improve productivity
- Keys are in abstraction and automation (SE strengths)
  - Provide computational abstractions reflecting the science and math of the problem domain
    - Reduce programming complexity (size, understandability, maintainability)
    - Ease verification
  - Provide hardware-independent abstractions for
    - Expressing algorithmic parallelization
    - Optimizing and tuning for performance, locality, latency, etc.
  - Automate mapping of abstractions to hardware (hard problem)
    - Parallelism, data layout, latency
    - Preserving sufficient performance
- Goals: reduce manual labor, allow scientists to reason in the problem domain

R&D Areas: Development

- Software processes tailored to SC goals and constraints
  - Agility to address changing requirements, time-to-solution
  - Risk mitigation
  - Distributed development
  - Project management metrics
- Requirements elicitation and specification for SC domains
- Software architecture and design
  - Concurrent design for tradeoffs in parallel performance, maintainability and portability
  - Patterns and canonical architectures
- Verification and validation
  - Establishing scientific validity in face of change
  - Automation, tracking, management, etc. over development cycle
- Strategic development
  - Optimization across multiple development cycles
  - Software product lines

Requirements for Success

- Successful research and development must address concerns for *relevance* and *risk*
  - SE community must work with SC community to identify requirements and constraints
  - Must revisit common SE assumptions, align with SC realities
  - Must re-engineer solutions (processes, methods, tools) or invent anew
  - Must validate on real problems
    - Demonstrate effectiveness in meeting developmental goals
    - Demonstrate sufficient control of run-time performance
    - Demonstrate cost effectiveness
- Success will require collaboration between the SC and SE communities

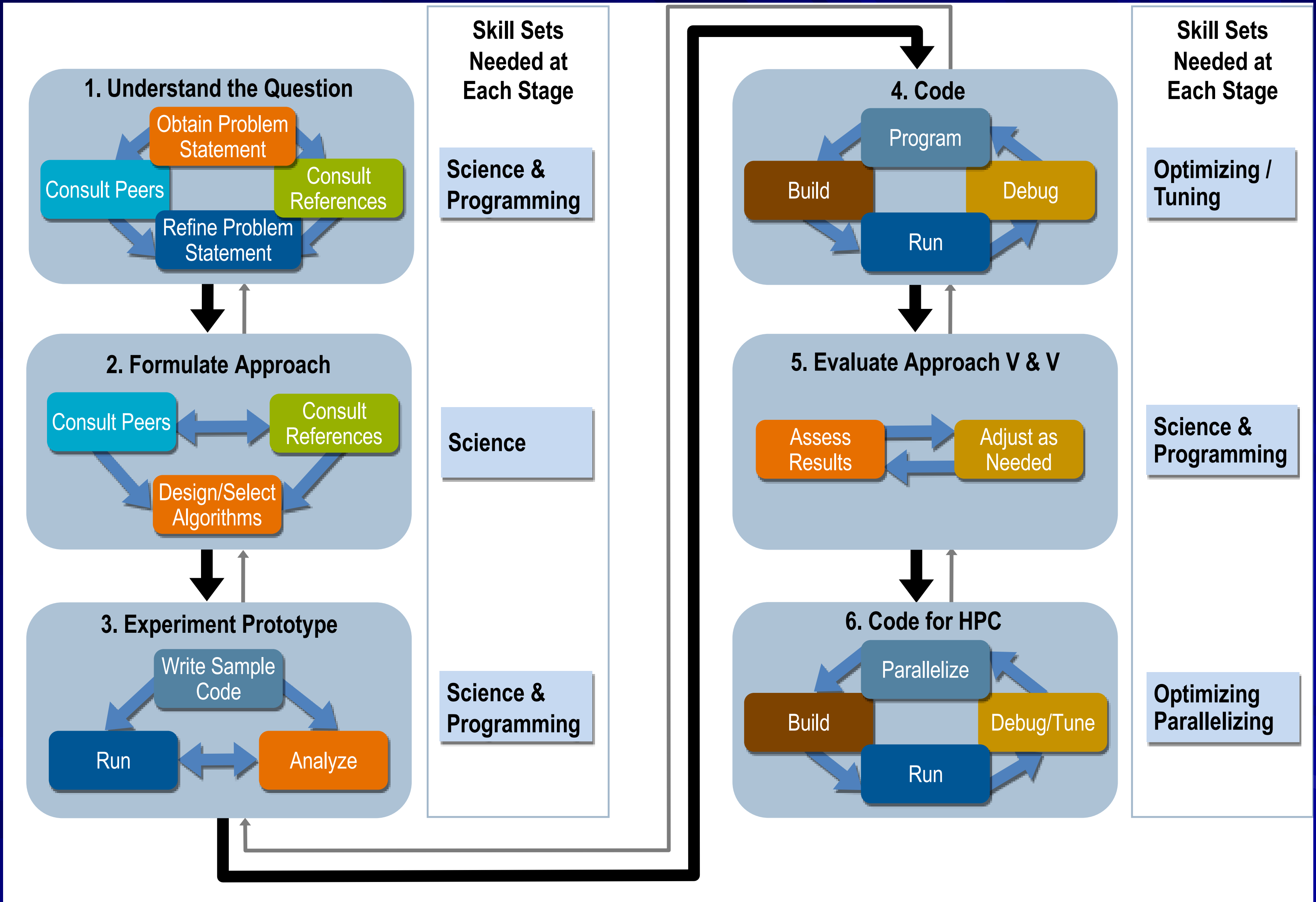
Current Resources

- DoD HPC Modernization Program: Computational Research and Engineering Acquisition Tools and Environments (CREATE)
  - Directed by Dr. Douglass Post (Dr. Larry Votta SE)
  - Demonstrating a range of SE methods in the HPSC context
    - Verification and Validation
    - Lightweight development methods
  - Ask for link
- DoE Advanced Scientific Computing Research (ASCR)
  - Exascale Computing Systems Productivity Workshop
  - http://www.orau.gov/ecsproductivity2014/

Questions?

Sources

- D. E. Post, R. P. Kendall, Internat. J. High Perf. Comput. Appl. 18(4), 399 (2004).
- Van De Vanter, M.L., Post, D., and Zosel, M.E. "HPC Needs a Tool Strategy". In Proceedings of Second International Workshop on Software Engineering for High Performance Computing System Applications (ICSE 2005). St. Louis. 2005. p. 15
- D. Post and L. Votta, “Computational Science Demands a New Paradigm,” Physics Today, vol. 58, no. 1, 2005, pp. 35–41.
- Jeffrey C. Carver, Richard P. Kendall, Susan Squires, Douglass E. Post, “Software Development Environments for Scientific and Engineering Software: A Series of Case Studies,” Proceedings of the 29th International Conference on Software Engineering, IEEE Computer Society, Washington, DC, pp. 440-559, May 20–26, 2007.
- Stuart Faulk, John Gustafson, Philip M. Johnson, Adam Porter, Walter F. Tichy, Lawrence G. Votta, “Measuring HPC Productivity,” International Journal of High Performance Computing Applications: Special Issue on HPC Productivity, J. Kepner (editor), 18(4), Winter 2004 (November).
- CREATE Link
  - http://www.hpc.mil/index.php/2013-08-29-16-03-23/software-applications-support-sas-overview/computational-research-for-engineering-and-science-cres/computational-research-for-engineering-acquisition-tools-and-environments-create



1. Understand the Question
Obtain Problem Statement
Consult Peers
Consult References
Refine Problem Statement
2. Formulate Approach
Consult Peers
Consult References
Design/Select Algorithms
3. Experiment Prototype
Write Sample Code
Run
Analyze
Skill Sets Needed at Each Stage
Science & Programming
Science
Science & Programming
4. Code
Program
Build
Debug
Run
5. Evaluate Approach V & V
Assess Results
Adjust as Needed
6. Code for HPC
Parallelize
Build
Debug/Tune
Run
Skill Sets Needed at Each Stage
Optimizing / Tuning
Science & Programming
Optimizing Parallelizing

HPC Workflow in Context

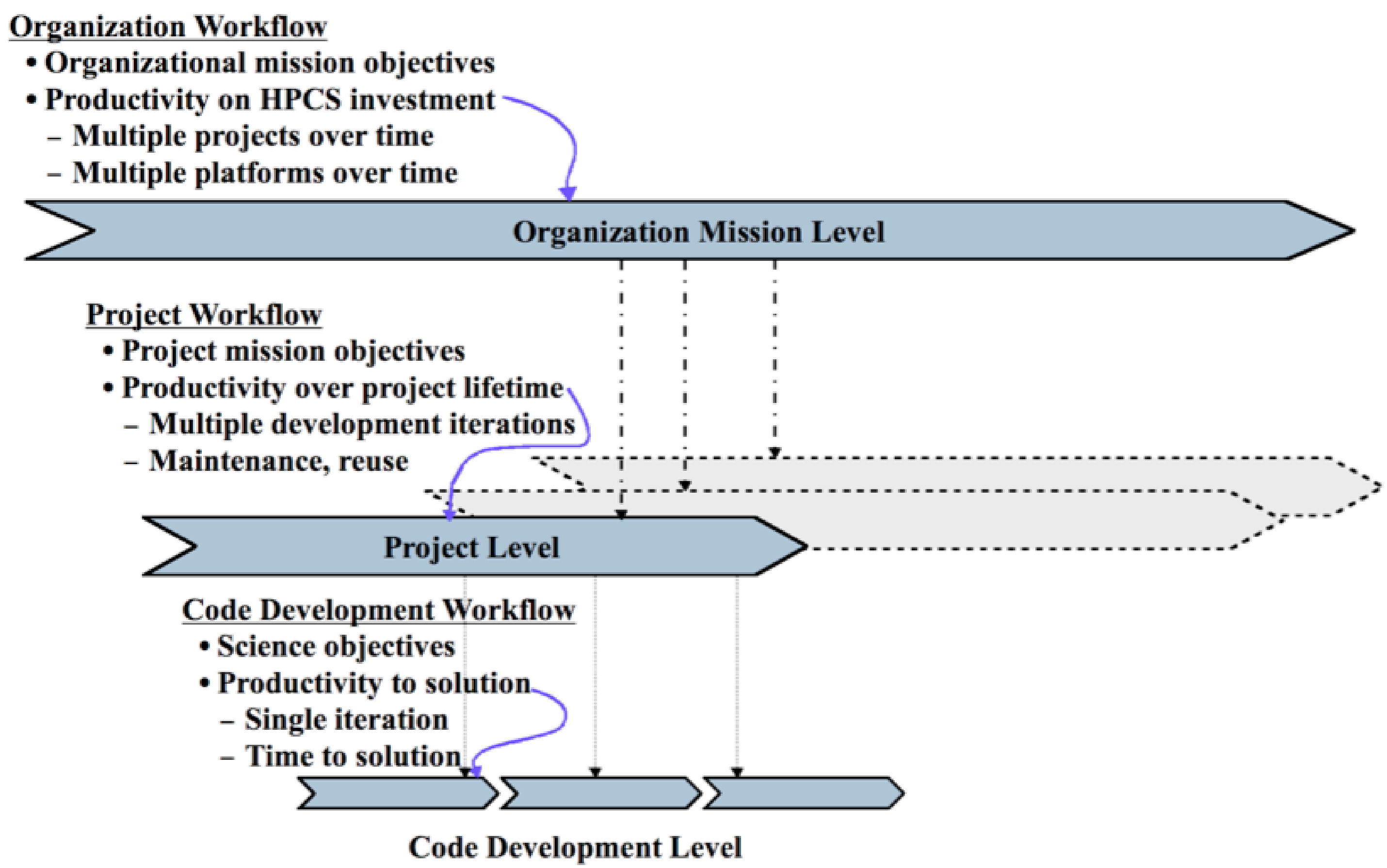


Figure 3: Software engineering perspective on workflows

Resulting Gridlock

- Currently stuck at local optima
- Bottlenecks are inherent in the approach
  - i.e., multi-disciplinary experts hand-crafting code
  - Current efforts seek to optimize this approach
- Cannot be resolved by doing more of the same
  - There aren't enough experts
  - We cannot produce them fast enough (even if we wanted to)
  - The need is growing
- *Productivity gridlock*: resulting inability to start solving productivity problems, even as overall productivity declines