A new DoD initiative: the Computational Research and Engineering Acquisition Tools and Environments (CREATE) program

D E Post for the CREATE Team: S. Arevalo, C. Atwood, P. Bell, T. D. Blacker, S. Dey, D. Fisher, D. A. Fisher, P. Genalis, J. Gorski, A. Harris, K. Hill, M. Hurwitz, R. P. Kendall, R. L. Meakin, S. Morton, E. T. Moyer, D. E. Post, R. Strawn, D. van Veldhuizen, L. G. Votta, S. Wynn, and G. Zelinski.

DoD High Performance Computing Modernization Program, Lorton, VA 22079, USA

E-mail: post@hpcmo.hpc.mil

Abstract. In FY2008, the U.S. Department of Defense (DoD) initiated the Computational Research and Engineering Acquisition Tools and Environments (CREATE) program, a $360M program with a two-year planning phase and a ten-year execution phase. CREATE will develop and deploy three computational engineering tool sets for DoD acquisition programs to use to design aircraft, ships and radio-frequency antennas. The planning and execution of CREATE are based on the “lessons learned” from case studies of large-scale computational science and engineering projects. The case studies stress the importance of a stable, close-knit development team; a focus on customer needs and requirements; verification and validation; flexible and agile planning, management, and development processes; risk management; realistic schedules and resource levels; balanced short- and long-term goals and deliverables; and stable, long-term support by the program sponsor. Since it began in FY2008, the CREATE program has built a team and project structure, developed requirements and begun validating them, identified candidate products, established initial connections with the acquisition programs, begun detailed project planning and development, and generated the initial collaboration infrastructure necessary for success by its multi-institutional, multidisciplinary teams.

1. A new paradigm for solving many of the major problems facing society

Since the invention of the electronic digital computer in the early 1940s, computing power has been increasing exponentially by a factor of ~ 1.7 each year from ~ 1 floating-point operations per second (Flops) to ~ $10^{15}$ Flops today (figure 1). This has led to enormous gains in our ability to develop, assess, and use information. Computers have revolutionized our ability to conduct national and international business and trade and to coordinate the diverse activities of society, both private and public. This revolution is also transforming the fundamental processes of science and engineering.

While computational science and engineering (CSE) has been making significant contributions to society, the future holds the promise of a major paradigm shift in problem solving from traditional methods toward CSE. Within 10 years, supercomputers will be able to perform $10^{18}$ calculations per second. With this computing power, we will be able to employ sufficient resolution to accurately model complex systems, deploy highly accurate computational mathematical algorithms, include all of the effects we know to be important in a system, model a complete system (e.g., an entire airplane or ship) in multiple dimensions, and be able to run enough problems to carry out useful parameter surveys.
While high-speed, powerful computers are the enabling, transformational technology, more than computers are needed (table 1). Sponsors use the information for decisions and designs and provide the resources and oversight. Code users apply software to solve problems. Code developers write and support the use of the software. Verification and validation ensure the accuracy of the calculations. Computers are needed to perform the calculations. Successful CSE requires all five elements.

Table 1. Essential elements of Computational Science and Engineering

| Sponsors                  | Code users and designers |
|---------------------------|--------------------------|
|                           | Codes (development and user support) |
|                           | Verification and Validation |
|                           | Computers                 |

2. Challenges facing computational science and engineering

The establishment of a new technology usually takes several generations and many mistakes. The design methodology for steel and concrete suspension bridges took almost 200 years to mature. While computational science and engineering has been making significant contributions to society, the future holds the promise of a major paradigm shift in problem solving from traditional methods toward CSE. Within 10 years, supercomputers will be able to perform $10^{18}$ calculations per second. With this computing power, we will be able to employ sufficient resolution to accurately model complex systems, deploy highly accurate computational mathematical algorithms, include all of the effects we know to be important in a system, model a complete system (e.g., an entire airplane or ship) in multiple dimensions, and be able to run enough problems to carry out useful parameter surveys [1].

Case studies of bridge-design errors led to the identification of “lessons learned.” Their incorporation into the design of future bridges was essential for this technology to mature. CSE will be similar. Unfortunately, there have been only a few case studies of CSE projects [2, 3]. However, these have provided an important number of “lessons learned,” and it is crucial to incorporate as many of them into future CSE projects as possible. Petroski points out that one learns the most from unsuccessful projects [1]. While there have been many failures in CSE software development [4], there have been almost no case studies of unsuccessful CSE projects, so special focus is needed on what little data exist for such projects.

Based on our case studies, we present many “lessons learned” in this paper [4, 5]. These lessons may seem obvious – and they are. However, our case studies indicate that implementing them in real projects is not common. Few projects follow most of the lessons, and many follow only a few. Our case studies trace the failure of many projects to achieve their goals within schedule and budget, or management decisions to abandon the projects, to violations of the practices and procedures captured in the “lessons learned.” This has resulted in a tragic...
waste of the creative energy and efforts of many talented people and financial resources, and the loss of the potential benefits of those projects. Based on the case studies (e.g., [5]), we group the challenges for CSE into three categories:

1. Scientific and Engineering - How to develop software that captures the scientific and engineering aspects that must be calculated and interpreted accurately
2. Project Management - Organizing and executing the development of the complex CSE software that meets the needs of the users
3. Programming and Computing - Developing and supporting software that can execute on tomorrow’s complex and rapidly evolving computer hardware

2.1 Scientific and engineering challenges

The scientific and engineering challenges of developing CSE codes reflect the complexity of the scientific or engineering subject matter. The codes must be able to accurately calculate the trade-off of many distinct physical phenomena over many orders of magnitude of time and distance scales. Since CSE codes represent models of nature, we must verify that the code has no important defects (verification) and establish that the model correctly includes all of the important effects (validation) [6, 7]. Incorrect and inaccurate calculations lead to wrong decisions and bad designs. Verification includes intensive software testing. Validation involves comparison of the code’s predictions with experimental data. Existing verification and validation (V&V) methods are inadequate for the present generation of CSE codes and research is needed to develop better V&V methods. Accurate numerical solution algorithms that can efficiently use many processors (up to $10^9$) will be necessary, but few such algorithms exist. Calculations of complex, multimaterial systems require multidimensional numerical representations of the objects. Methods for rapidly generating the required accurate geometries and numerical grids and meshes generally do not exist and need to be developed. Generating an accurate mesh for a large ship can take a team of analysts a year, far too long to benefit design decisions. Improved techniques for rapid analysis and visualization of the huge datasets generated by large-scale codes are required to ensure that computational results will provide useful information for making decisions or developing designs.

2.2 Project management challenges

The development and support of complex scientific and engineering software are a complicated process. The software must meet the needs of the customers, namely, the designer or user community. It must provide accurate solutions for the problems that the users need to solve, and it must be convenient to use. The users’ requirements are often difficult to determine and constantly change over the life cycle of the project.
code. These issues are especially challenging if the users or customers are not the code developers as is the case for codes for design engineers. Many “scientific” codes are developed and run by the same groups who then publish the results of their investigations. These groups understand the limitations and range of validity of their codes. In contrast, many distinct groups are involved in the steps in Figure 3. Following development and V&V, the code will be used to develop a design for a system that will be built, tested and operated by distinct groups. Few of these disparate groups have a detailed knowledge of the strengths and weaknesses of the code. The quality of the code must therefore be very high since the quality of the final product depends on the quality of the code and the skill of the design engineer.

Development and initial deployment of large-scale CSE codes generally take large, multidisciplinary teams (~20–30 professional staff) on the order of 10 years (figure 2) or more [5]. The required funding is around $100M over a ten-year period. The development of large-scale code requires more organization and structure than the development of smaller codes. The team leaders must possess many skills, including subject matter, computational, managerial, financial, leadership, and team-building skills. Multi-institutional teams require close collaboration among the team members from the participating institutions. Each institution must provide adequate support for their portion of a distributed project. The development of large-scale codes is a complex, iterative process (figure 3). Large-scale code development is risky. As many as one-half or more large-scale CSE code projects fail to meet their initial goals in terms of performance, cost and schedule. Up to one-third (or more) are eventually abandoned after significant investment [5, 7].

2.2 Programming challenges

Computers are becoming more complex as a result of massive parallelism and the use of heterogeneous processors. Computers in 2020 will likely have as many as $10^9$ processors and execution speeds of $10^{18}$ Flops. Exploiting even a small portion of this power will require overcoming the architectural complexity of $10^9$ or more processors connected through complicated networks. Codes will need to be able to achieve reasonably efficient performance (~ 5% or more of peak speed). It will be necessary to port codes to each generation of computers every three to four years, and different computer architectures every 10 years or so, as well as port codes to a variety of existing platforms. Code development tools for tomorrow’s computers are inadequate and immature, when they exist at all.

3. Lessons learned

Case studies of many large-scale CSE code projects have identified the requirements for successful development and deployment of such codes (table 2).
Table 2. Essential elements of successful CSE code development projects [5]

1. Sponsor support for the life of the project
2. Requirements based on the evolving needs of the customers over the life of the project, i.e. strong customer focus
3. Verification and validation
4. Software project management designed for large-scale CSE codes
   • Realistic goals and schedules with adequate resources
   • Balance of long-term and short-term goals
   • Sufficient project orientation to organize and coordinate the work, but enough flexibility to solve unanticipated problems and meet new requirements
   • Multidisciplinary team with strong technical skills
   • Good team dynamics: trust, respect, cooperation and commitment
   • Collaboration methods and tools for multi-institutional code development teams
5. Software engineering designed for large-scale CSE codes
   • Risk identification and mitigation
   • Effective software engineering methods and tools that balance the need for structured development with the required degree of flexibility and agility
   • A flexible code architecture and infrastructure

The evidence indicates that it takes on the order of $100M for 10 years to build a large-scale CSE code [5]. To get this level of support, one has to first find someone who can and will pay $100M to solve their problem and then make a convincing case that you can develop a CSE tool that can solve their problem (table 3). It is not sufficient to have a good proposal to solve a problem one thinks is important. The idea must solve a problem that someone with $100M thinks is important. This is not easy. Not many groups have resources at that level, nor are they willing to wait up to 10 years for a code. In addition, CSE is not yet a widely accepted problem solving paradigm.

Table 3. Four elements for marketing a code development program

1. Identify a problem that needs to be solved, that CSE can solve, and that a sponsor will pay to have solved.
2. Establish that solving the problem can provide enough benefit to the sponsor that he will pay for the solution.
3. Establish that your group has credibility for solving the problem.
4. Present a compelling vision for solving the problem.

4. An Example: A new Department of Defense Initiative, The Computational Research and Engineering Acquisition Tools and Environments (CREATE) program

In 2008, the Department of Defense (DoD) initiated the CREATE program, a new $360M program to develop and deploy three computational engineering tool sets for the design of airplanes, ships and radio-frequency (RF) antennas. CREATE is a multiservice DoD program with support from the Office of the Secretary of Defense (~70%) and 30% matching “in-kind” resources from the armed services (Navy, Air Force and Army). CREATE is a 12-year program with a 2-year planning phase and a 10-year development phase. The program was conceived and planned based on the “lessons learned” described in the prior sections, and the concept for CREATE was developed in 2005-2006. The proposal was developed and submitted during 2006-2007 as part of the DoD budget process (Program Objective Memorandum, POM) for the 2008 federal fiscal year. CREATE was approved in FY07 and started in FY08.

The CREATE proposal was built around four points: problem, benefit, credibility, and vision (table 3). We began by identifying DoD acquisition programs as organizations with severe problems that computational tools could help solve. Many major DoD acquisition programs are years behind schedule,
billions of dollars over budget, and fail to meet performance goals [8]. The acquisition process is rigid and slow, and it is not able to respond rapidly to changing threats [9]. The benefit of an improved, flexible, and agile acquisition process would be timely delivery of weapons systems that met performance requirements at the projected cost and the ability of acquisition programs to respond to rapidly changing requirements. The U.S. would save billions of dollars, weapons systems would be available for operations at the scheduled time, and the systems would work as specified.

Part of acquisition program shortcomings are due to the steady decline in the size of the DoD engineering workforce (a factor of 5 to 10 over the past 35 years) [9]. Another challenge is the increased need to accelerate the introduction of new design concepts and technologies into new weapons systems. Historical experience is not a good approach for new designs based on concepts and technologies outside of the existing design experience. The present acquisition and design paradigm is based on an iterative, empirically based “design, build, test, fail, redesign, etc.” approach. The CREATE proposal was based on the development and deployment of physics-based computational engineering tools to replace design methods based on historical experience. The CREATE tools would thus enhance the productivity of the acquisition workforce by enabling acquisition engineering to produce better designs more quickly. Using evidence and lessons-learned from our case studies, the credentials of our team and successful examples of DoD CSE applications, we established our credibility to build the proposed computational engineering tools within our estimated schedule and budget. We illustrated how use of CSE tools had helped other government agencies and industry solve problems similar to those facing DoD acquisition programs and achieve the desired benefits.

The physics-based CREATE tools will be able to produce optimized designs early in the design process. Design engineers could use the tools to identify design defects and correct them early in the design process before major schedule and budget commitments were made. Testing would be reduced by getting the design right the first time. Tests would be used to validate a mature design rather than identify the problems in an immature design. Experimental prototypes can be replaced with computational prototypes that can be constructed much more quickly and cheaply. Industry is beginning to employ this approach. One business used CSE to reduce their time to market from three years to less than 9 months, and credits their company’s survival to the adoption of CSE.

4.1 Key elements of the CREATE program

Once the CREATE proposal was funded, we then began to plan the program in detail. We had embedded the lessons learned listed in table 2 into the initial program proposal. We were able to build on them so that our plans were consistent with what we had promised in the proposal. In the proposal process, we first ensured that our DoD sponsors understood that CREATE was a long-term program and that it would take $10M/year for at least 10 years to fully develop each suite of codes. Second, we had established connections to the prospective CREATE customers and users, the DoD acquisition programs. The proposal included endorsements of CREATE and promises of resource commitments of about 30% of the total CREATE budget from the acquisition programs. Third, we emphasized the importance of the elements in table 2 to our DoD sponsors so that they would understand and support CREATE during the early phases of the program prior to delivery of our major products. It was especially important for our DoD sponsors to understand our challenges and provide the institutional support as well as financial support to meet those challenges. These include resolving infrastructure issues and providing access to the personnel required to establish and execute a successful multi-institutional program.

We began to carefully build a strong leadership team for the CREATE program and the three major projects—aircraft, ships, and RF antennas. We solicited and received nominations for the leadership position from the offices of assistant secretaries of the Navy, Army, and Air Force as part of an open, advertised national search process. The position descriptions and selection criteria emphasized the need for a balance of subject matter expertise in ship, aircraft, or RF antenna design issues with expertise and experience in project management and leadership experience in large-scale computational science
4.2 Requirements development and validation

To ensure sponsor support and customer focus, we examined the acquisition programs to identify gaps in their capabilities to design and deliver weapons systems involving ships, airplanes, and RF antennas. We then identified the capability gaps that would benefit from the application of computational engineering tools. Then we developed concepts and ultimately requirements for candidate tools to fill those gaps. Finally, based on the potential impact on high-priority programs, the maturity of the computational technologies and physics-based algorithms, and the budgeted resources and schedule, we selected a set of products to develop from the list of the candidate tools. We documented the gaps and requirements in an “Initial Capability Document” (ICD) patterned after the DoD Acquisition Process (Joint Capabilities Integration Development System, JCIDS) [10]. Mapping our process into the general DoD process helped us meet the expectations of the DoD community. We are using “Boards of Directors” composed of senior staff (generally at the flag rank, Senior Executive Service level or equivalent) for the targeted acquisition programs to validate the gaps and requirements described in the ICD. This approach ensures that the tools we plan to develop will meet the needs of the acquisition programs. The board members will vet our ICDs through their organizations before endorsing them, thus ensuring that we have correctly identified the gaps and requirements. Since the gaps and requirements evolve and change, we intend to revalidate them at least once a year to keep them current.

Studies of Systems of Systems in the DoD, of which large, complex codes are an example, show that a leading cause of failure of large-scale systems is that many do not update their requirements rapidly enough to fulfill the evolving needs of their customers [11]. The resultant system is thus not useful to the customers when it is finally delivered. Revalidating our requirements annually will help keep the projects on track. CREATE has formed small teams who are charged with “shadowing” the acquisition programs. These “Shadow-Operations” (Shadow-Ops) teams will identify opportunities for computational engineering tools to address problems in acquisition programs. The Shadow-Ops teams will then apply computational tools to solve acquisition engineering problems and share the results with the programs. This approach enables the Shadow-Ops teams to track the evolving requirements of the acquisition programs, helps establish the value of computational engineering analyses to the acquisition programs and helps facilitate the transition of computational engineering tools to the programs. The initial results of this approach indicate that it will be highly successful.

Verification and validation (V&V) are essential. The CREATE program is establishing strong connections to the DoD Test and Evaluation (T&E) community to ensure access to validation data. This community has been a strong supporter of the CREATE program, because CREATE tools will help the T&E community fulfill its testing role with fewer tests and greater efficiency and effectiveness. The goal is to establish a design paradigm in which designs will be based not on historical experience requiring a long series of exploratory tests, but on one in which designs will be developed using physics-based computational tools. Physics-based tools are much more effective for new concepts and technologies than reliance on historical knowledge. Such designs can be validated with a much smaller set of targeted tests. This has been a highly successful strategy for several businesses. A strategy that emphasizes the specification of tests as part of the development process is also being developed.

4.3 Code development strategy

As part of our planning process, we assessed the requirements of the acquisition programs and prioritized them to help identify the most useful tools that could be developed for ~$10M/year for 10 years. To minimize risks, we emphasized building on existing expertise and knowledge, and focused on codes that integrated four to five different physical effects. We started developing detailed project plans based on our initial gaps and requirements analysis. Those plans will evolve as the requirements change.
We have developed a strategy to achieve a good balance between long- and short-term deliverables. Without establishing value in the short-term, we will not be able to defend our long-term budget. However, if we do not keep a long-term focus, we probably will not be able to develop codes that can exploit 10^6 cores and 10^{18} Flops to achieve the goal of a paradigm shift from reliance on historical data and experience for design to the use of physics-based tools for new concepts. Our strategy is to begin by increasing the capability, parallel scalability, robustness and usability of existing computational engineering tools, and support their transition to the acquisition programs (figure 4). We will then use the “lessons learned” from this experience to design and build new tools designed to exploit the next generation of supercomputers in 2020 and transition them to the users when they are ready. The Shadow-Ops program described above will we help realize this strategy. It will also help keep the code developers closely tied to the user community and the user community’s current requirements.

We are developing a flexible and agile project management process that is designed to ensure the achievement of project goals and tasks while retaining the flexibility and agility to respond to events that occur during the development process (e.g., figure 3).

We have emphasized the formation of a good team with the range of experience and skills necessary to carry out the code development projects and transition the products to the customers. The DoD High Performance Computing Modernization Program (HPCMP) Institutes and Portfolios have helped to create a community of trained and skilled computational scientists who have formed the core of much of the CREATE leadership and code development teams. We have also emphasized subject matter expertise, and technical leadership and team building skills, rather than strong project management skills and background.

![Figure 4. CREATE code development strategy.](image)

Code development requires strong teams that have good team dynamics. Software is developed by teams. We have found that the level of sponsor support and the quality and cohesiveness of the team are three of the most reliable predictors of project success [12].

No single DoD site has sufficient staff to execute a CREATE project. We have therefore emphasized the development of effective methods for multi-institutional teams to work together effectively and efficiently. We have established a central server repository for code development files and documents. We are procuring and supporting collaboration tools, including several different types of high-quality video conferencing desktop tools that are essential for good communication and building trust and a sense of shared commitment and community within a highly distributed project. Computer security issues that inhibit communication among DoD facilities have made this challenging.

Large-scale code development has large risks [7]. Historically, half or more of such projects fail to meet their initial goals within the estimated budget and schedule. We have assessed the general risks for CSE [13]. The most critical risk is the potential loss of sponsor support [12]. For CREATE, we have therefore focused on providing short-term and long-term value to our customers and ensuring that our customers keep our sponsors informed of this value. Another major risk is the stability and competence of
the development team. We concentrated on building a solid, competent leadership team because it is the first step in building a solid project team. Risk assessment and the development of mitigation plans will be a priority for FY09.

Developers of engineering and scientific codes have found that imposition of rigid processes, for example the Capability Maturity Model (CMM, CMMI) [14, 15], is not useful [5]. Instead, we have concentrated on identifying the software engineering practices and tools that have proven useful for large-scale engineering projects [16, 17]. We are providing training in these practices and are establishing a central set of servers to provide those tools to the teams. This approach allows us to capture the best aspects of the process approaches like CMM but retain the flexibility and agility we need.

The CREATE computational engineering tool suite must be usable, modular, extensible and maintainable. We are emphasizing ease of use through extensive use of Graphical User Interfaces (GUI) for the Application Program Interface. Easy and convenient generation of geometries and meshes is a priority. Extensibility and maintainability are necessary to provide the new functionality that the users will require as their needs change and computer platforms evolve. Modularity and interoperability are essential to achieve these aspects. The products developed by each sub-team must be usable by the other subteams. For instance, many of the RF antenna modules must be usable by the ship and aircraft design tools. We have been developing a strategy based on a lightweight infrastructure that provides wrappers for each major set of modules (computational fluid mechanics, structural mechanics, control algorithm, etc.) to achieve a high degree of modularity. Each module has well-defined interfaces so that it can be used for multiple purposes. Lightweight means that the interfaces and wrappers are designed to handle our specific classes of problems, rather than broad classes of problems. Our “lightweight” infrastructures are “shallow” in the sense that the interfaces do not intrude too deeply into the code modules. The emphasis is on code-interoperability rather than tight integration. If a new need arises, the relevant wrappers are extended to accommodate the new interface and connectivity requirements. If necessary, enhanced functionality can be added to the module core. We have found that heavyweight frameworks that emphasize generic “plug and play” aspects are not as useful. Such frameworks attempt to address many different types of problems. This forces them to incorporate choices and assumptions that often result in too much rigidity to find technical solutions for different problems. Rigid frameworks have proven to be a major cause of failure of some large-scale CSE projects [4]. The lightweight infrastructure approach also allows us to deliver products quickly while retaining a long-term focus. Wrappers and GUIs to link existing major code modules (e.g., computational fluid dynamics or structural mechanics modules) can be built and delivered quickly to the user community within a year or two. We will build on this experience to design and develop new major modules and solvers (e.g., fluid dynamics, structural mechanics, etc.) that can take advantage of the next generation of computer with 10^5 to 10^6 processors, but will take 5 to 10 years to design and build.

We have been able to exploit these lightweight infrastructures to simultaneously provide short-term deliverables with immediate value while keeping a focus on meeting our long-term objective to provide codes that can exploit the next generation of computers (figure 4). The political reality is that if we cannot provide near-term value to the DoD, our long-term funding will be at risk in the highly competitive resource environment in the DoD. We are beginning to link several existing legacy codes with a user-friendly, light-weight framework to provide the ability to study integrated, multi-physics effects such as the interaction of aerodynamic air flow and the structural mechanics of an airframe. This is being accomplished in one to two years, much shorter than the five to ten years that experience indicates is required to develop a new computational fluid dynamics or structural mechanics module. This allows us to provide design engineers the capability to study the design implications of these interactions early in our project, and us the opportunity to learn about the strengths and weaknesses of existing codes and the real needs of the design community. We can then use this knowledge to design and develop new codes that can exploit the power of the next generation of computers.
5. Summary
We built the CREATE program on the “lessons learned” from the past 50 years of experience with large-scale CSE software. While some of these lessons are undocumented, many have been described in papers and reports (e.g., [18]). This experience has been essential for us to get the necessary resources and to plan and execute the CREATE program. CSE is a new, emerging paradigm. As noted earlier, many of these lessons seem obvious but we have found that very few projects incorporate them, resulting in the waste of people’s time, the country’s resources, and loss of the potential benefits of the project. It is crucial that everyone launching CSE projects examine and learn from the past experiences of others who have tried and succeeded, and especially those who have not succeeded. “Those who cannot learn from history are doomed to repeat it” [19]. Every new technological paradigm builds on the mistakes of prior efforts. We urge our CSE colleagues to share and document their experiences, both the good and especially the bad, so that our field can move forward, and society can have the full benefit of CSE tools.

Acknowledgments
The CREATE team is grateful for the support and advice from our colleagues including John Blair, Luise Couchman, Larry Davis, John Grosh, Cray Henry, Jeremy Kepner, Ed Kraft, Scott Littlefield, Robert Lucas, Andrew Mark, Paul Nielsen, Patrick Purtell, Cliff Rhoades, Al Shaffer, André van Tilborg, and John Young.

References
[1] Petroski H 1994 Design Paradigms: Case Histories of Error and Judgment in Engineering (New York: Cambridge University Press)
[2] Post D E, Kendall R and Whitney E M 2005 Case study of the Falcon code project In Johnson P (Ed.) 2nd Workshop on HPC applications, ACM/IEEE International Conference on Software Engineering (St. Louis, MO: Association of Computing Machinery)
[3] Hatton L 1997 The T experiments: errors in scientific software IEEE Computational Science and Engineering 4:27-38
[4] Post D E and Kendall R P 2004 Lessons learned from ASCI International J. High Performance Computing Applications 18:399-416
[5] Post D E, Kendall R P and Lucas R F 2006 The opportunities, challenges, and risks of high performance computing in computational science and engineering In Zelkowitz M V (Ed.) Advances in Computers (New York: Elsevier)
[6] Oberkampf W and Trucano T 2002 Verification and validation in computational fluid mechanics Progress in Aerospace Studies 38:209-272
[7] Post D E and Votta L G 2005 Computational science demands a new paradigm Physics Today 58:35-41
[8] GAO 2007 General Accounting Office: Defense Acquisitions, Assessments of Selected Major Weapon Programs 167, GAO-07-406SP.
[9] Kaminski P G, et al. 2008 Pre-Milestone A and Early-Phase Systems Engineering (National Academy of Sciences)
[10] Garrett G A and Rendon R G 2006 U.S. Military Program Management, Lessons Learned and Best Practices (West Sussex: Kogan Page Publishers) CJCSI 3170.01C
[11] Fisher D A 2006 An emergent perspective on interoperability in systems of systems 66 CMU/SEI-2006-TR-003
[12] Demarco T and Lister T 2003 Waltzing with Bears, Managing Risk on Software Projects (New York: Dorset House Publishing)
[13] Kendall R P, et al. 2007 A proposed taxonomy for software development risks for high-performance computing (HPC) scientific/engineering applications 39 CMU/SEI-2006-TN-039
[14] Chrissis M B, Konrad M abd Shrum S 2003 CMMI: Guidelines for Process Integration and Product Improvement (Addison-Wesley Professional)
[15] Humphrey W 1989 *Managing the Software Process* (Addison-Wesley Professional)

16] Wilson G 2006 Software carpentry, getting scientists to write better code by making them more productive *Computing in Science and Engrg.* **8**:66-69

[17] Post D E and Kendall R P 2007 Large-scale computational and engineering code development and production workflows *Use of High Performance Computing in Meteorology* Mozdzynski G (ed.) (London: World Scientific)

[18] Post D E and Kendall R P 2006 Large-scale computational scientific and engineering project development and production workflows *CTWatch Quarterly* 2

[19] Santayana G 1905 *Reason in Common Sense* (New York: Charles Scribner's Sons) 504