Case Study of the NENE Code Project

Richard Kendall (Software Engineering Institute)
Douglass Post (DoD High Performance Computing Modernization Program)
Andrew Mark (DoD High Performance Computing Modernization Program)

January 2007

TECHNICAL NOTE
CMU/SEI-2006-TN-044

SEI Director's Office

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

The Defense Advanced Research Projects Agency (DARPA) High Productivity Computing Systems (HPCS) Program is sponsoring a series of case studies to identify the life cycles, workflows, and technical challenges of computational science and engineering code development that are representative of the program's participants. A secondary goal is to characterize how software development tools are used and what enhancements would increase the productivity of scientific-application programmers. These studies also seek to identify "lessons learned" that can be transferred to the general computational science and engineering community to improve the code development process.

The NENE code is the fifth science-based code project to be analyzed by the Existing Codes subteam of the DARPA HPCS Productivity Team. The NENE code is an application code for analyzing scientific phenomena and predicting the complex behavior and interaction of individual physical systems and individual particles in the systems. The core NENE development team is expert, agile, and of moderate size, consisting of a professor and another permanent staff member, five post docs, and 11 graduate students. NENE is an example of a distributed development project; the core team is anchored at a university, but as many as 250 individual researchers have made contributions from other locations.

1 Introduction

Through the sponsorship of the Defense Advanced Research Projects Agency (DARPA) High Productivity Computing Systems (HPCS) program, the present authors have performed a series of case studies of high-performance code development projects. This is the fifth case study in this series (after Falcon [Falcon 2005], Hawk [Kendall 2005a], Condor [Kendall 2005b], and Eagle [Kendall 2006]). The shared objectives of these studies are:

- identification of critical success factors
- identification of issues that must be addressed by hardware and software vendors to improve the productivity of the code development process
- development of a reference body of case studies for the computational science and engineering community
- documenting lessons learned from the analysis and personal team interviews

It is important in studies of this type to maintain the anonymity of the code project, the host institution, and the sponsoring agency or commercial company. "NENE" is a pseudonym and details that might reveal the identity of this code project have been masked or omitted.

This study followed the methodology described in the previous case studies by our team:

a. Identify the project and sponsors.
b. Negotiate case study with team and sponsors.
c. Complete pre-interview questionnaire process.
d. Analyze the questionnaire and plan on-site interviews.
e. Conduct on-site interview with the team.
f. Analyze the on-site interview and integrate with questionnaire.
g. Conduct follow-up to resolve unanswered questions.
h. Write a report and iterate with code team and sponsor.
i. Publish the report.

2 Code Characteristics

The NENE code development project is now 25 years old and the code is still under very active development. It began as a university research project. Stakeholders, primarily the core team and a large community of collaborators, set the requirements. Approximately 90 percent of the requirements have been driven by the research goals of the stakeholders and sponsors, and 10 percent from the specific needs of other users. NENE had an antecedent which has become a separate application.

NENE has evolved into a suite of modules and codes (tools) that can be combined to calculate the complex behavior and interactions of a set of the individual physical entities and particles. The user can trade off the accuracy of the calculations with time to solution through the selection of the host computer and the choice of physical effects and solution algorithms for solving the problem (Figure 1). Many users download the NENE code and find that the "standard core code" has sufficient capability to solve their problem (Figure 2). However, the "standard" effects and solution methods are sometimes inadequate. In such cases the scientist who wants to use NENE must develop a model for another effect, or develop a new solution algorithm to be added to the existing code to solve the problem of interest. Sometimes this "new capability" is of sufficient general interest that it becomes a candidate for inclusion in the core code. As discussed later, acceptance of new modules into the core NENE code is carefully controlled by the core team. In this sense NENE is a good example of a "feature-driven" code development project.

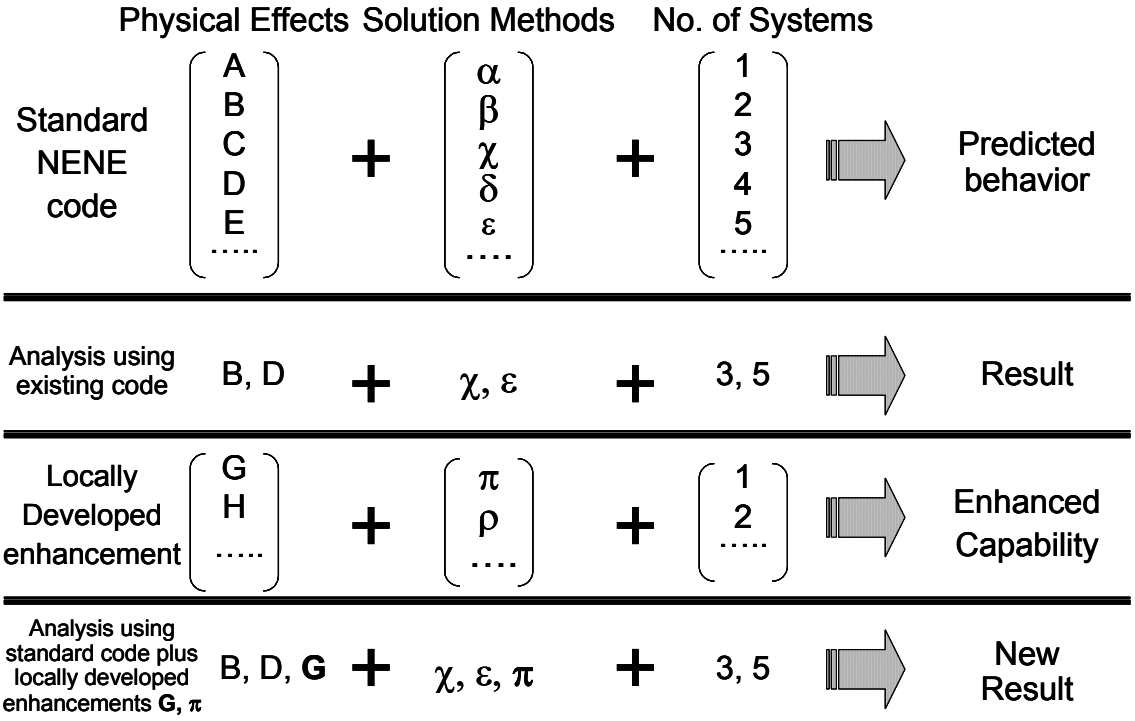

Figure 1: Schematic Illustration of NENE Development

There are now thousands of instantiations of NENE at various institutions and tens of thousands of users worldwide, both academic and non-academic (Figure 2). The code is now distributed via the Web. There is no license fee, however a registration is required to download the code. In the early days it was distributed by tape, later by the email of many separate files, and from the mid-1990s to 2000 via ftp. While, as described later, a strong effort is made to make the code as correct as possible, the code is provided "caveat emptor" to the users, with no guarantee of correctness for solving the problem of interest to the user and not much support beyond the downloaded code and the associated extensive documentation. The quality of the code is attested by the more than 5,000 citations to the seminal paper describing the core code system.

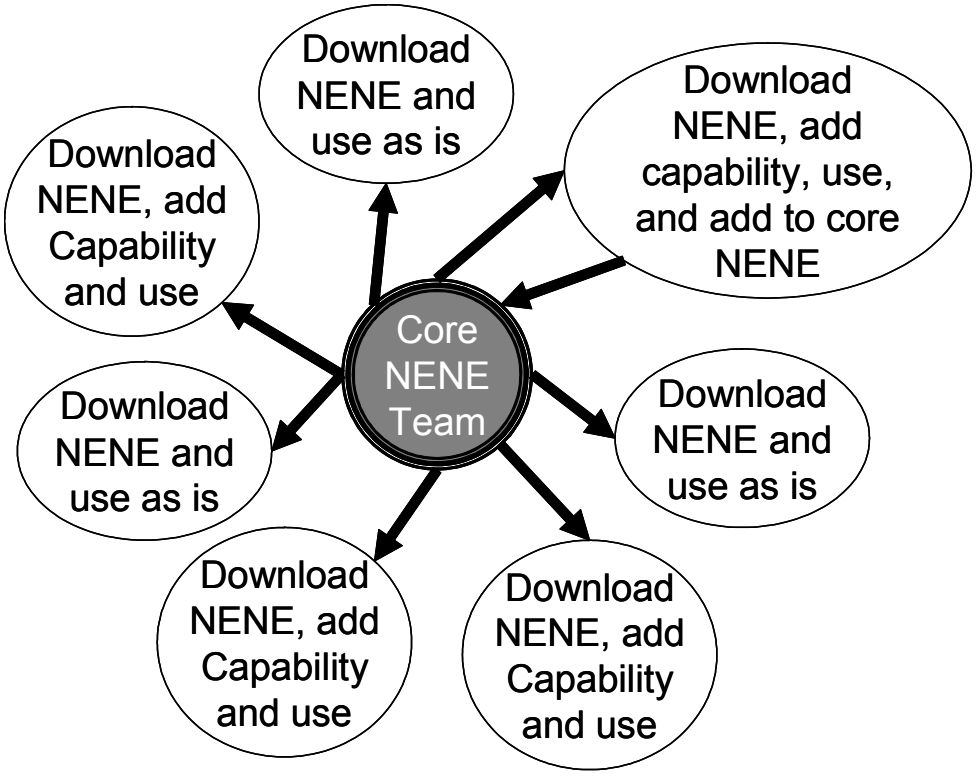

Figure 2: Typical Use Pattern for NENE

The NENE code, with the exception of the parallel program libraries, is written in Fortran 77. The motivation for this choice is reminiscent of several of our previous case studies (such as FALCON and CONDOR), namely portability, ease of development, and maintenance. F77 was the best option available at the inception of NENE and even though Fortran 77 is no longer taught by university computer science departments, its fundamentals can be mastered in a week in contrast to months or longer for C++ or other "modern," higher-level languages. Fortran 77 is easy to learn, intuitive to scientists, and remains highly portable, even though it is typically compiled by F90-era compilers, not F77 compilers. Sticking to Fortran 77 also eliminates linking issues that plague software written in many languages [NENE Co-PI]. Almost all the code development is done by domain scientists with little or no software engineering or computer science assistance.

There is essentially one version of the NENE program library that executes on all commonly used hardware platforms (all Unix environments, desktop Mac OS X, and Windows PCs). There exists

some limited vectorization in the program library for a few platforms, and there is an optimized Windows version.

There are approximately 760,000 lines of F77 code in the program library and approximately 26,000 lines of documentation. The message-passing API, also developed by the NENE team, consists of approximately 10,200 lines of C code. Message passing is handled by conditional compilation directives, and can rely on one or more of the following: TCP/IP, MPI, LAPI, and SHMEM.

Parallelization is a key priority of the team because parallel computing is necessary to get the level of performance needed to obtain the most innovative and important scientific results (see Chapter 5, "Lessons Learned," item *g*). On the other hand, actual demonstrations of parallel performance have been limited. The core difficulty is that domain science involves the strongly coupled interaction of many physical entities. Most solution algorithms require good communication among all parts of the problem. Memory latency and memory hierarchy are thus key issues.

3 Code Project and Team

The NENE core development team started out as one professor, two post docs, and two graduate students. Now it consists of a professor and another permanent staff member, five post docs, and 11 graduate students. Since 1993 a growing body of collaborators, mostly former graduate students and post docs, have made contributions to the NENE code. The contributors now number into the hundreds. The main drivers for the team are

- ease of maintenance
- robustness
- modularity
- portability
- standards compliance

The first two are judged to be the most important. The NENE team co-principal investigator (Co-PI) remarked that all of the drivers are related, but singled out the first as the most important driver of code development (see Chapter 5, "Lessons Learned," item *b*). Standards compliance, which is enforced with the FTNCHEK tool [SourceForge 2006], is important because of the large number of widely distributed contributors to the code. Modularity, with interfaces to the code backbone that minimize connections to the rest of the code, also promotes ease of maintenance. As was the case for the HAWK, CONDOR, and EAGLE projects, the relatively small size of the team limited the degree of formality required to manage the project. Moreover, the team leader is able to play a role in the development of the code, through supervision of graduate students, rather than just managing the project. The sponsors are attracted to the innovative science in the NENE code and the large user base (see Chapter 5, "Lessons Learned," item *d*) and have not required intrusive project management methods or onerous reporting requirements for the principal investigator. Milestones are required by one of the sponsors, but they are only "guidance" as is the usual practice with grants from federal science-funding agencies. While this might be a problem in some fields, it is not a problem with NENE. Grants must be renewed, and renewals do not occur unless there is a demonstrated record of achievement. From the formal software engineering point of view, this project is under-constrained, which is also true of all but one of the other projects our team has studied. However, for the most part, the NENE team sets its own milestones, which are driven by the need to enable students to graduate, and to provide publications for post docs on the project.

Of the 20 or so software development tracking metrics [SEI 2006] that have been cited in the software engineering literature, the NENE team chose to employ

- lines of code
- comment lines
- locality
- code performance
- parallel scaling

- number of users

"Lines of code" is used as a general measure of the growth of the code. Performance indicators like runtime and number of iterations are used to validate changes. (Validation is discussed later.) Number of users is important to funding agencies and industrial users as a measure of impact.

The fact that much of the NENE source code has been developed by a large group of external collaborators represents a risk (see Chapter 5, "Lessons Learned," item *c*) to the coherence and conceptual integrity of the NENE code. This risk has been addressed in part by instituting rigorous, centralized quality control by the code librarian over contributed code. New code must be tested thoroughly by its submitters; it is then submitted to an alpha/beta testing process before final inclusion in the program library. There is only one official supported source for the NENE code. Because most of the external contributions are voluntary, they do not impact the commitments made by the core team to its sponsors.

As the code becomes more complex, the question arises: at what point does the configuration management process itself become a bottleneck? The answer so far has been to keep the backbone of NENE as simple as possible and confine the enhancements, especially those from outside the core team, to the extremities.

The approach to configuration management here differs from that of any other project we have studied. Instead of a reliance on a software solution (like the Concurrent Versions System), the NENE project has taken the approach of assigning the task of program librarian to a Co-PI, an individual with almost encyclopedic knowledge of the code. The program librarian does not merely run regression tests to confirm that updates do no harm, but analyzes every line of code that is submitted for inclusion in the program library. This often involves working side-by-side with the developer. Also, many users select just a few key modules and use them for their calculations (Figures 1 and 2). Any new capability is often the replacement of an existing core module or modules with a new one written by the user.

Long-lived software development projects inevitably face a succession issue—who will replace the PI when he or she retires? A new PI has been recruited and the transition has begun.

The development style that best describes this team's dynamic is "agile" [Agile Alliance 2006] (see Chapter 5, "Lessons Learned," item *a*). This statement should be taken to mean that the NENE development team emphasizes practices over processes. The table below summarizes the development practices used by the NENE team. Although the NENE core team does not use style sheets in any formal sense, the code has a definite style that is communicated through "informal pressure and example" to successive generations of students who work on the code (see Chapter 5, "Lessons Learned," item *e*). Expertise spreads as students graduate and become professors with students of their own. They integrate NENE into their research programs and develop extensions. This is how the NENE code project has grown to have hundreds of collaborators.

Table 1: Development Practices Deployed by the NENE Team

Practice	Description	Degree Followed
Requirements Development	Produce, analyze, and verify customer, project, and product requirements	Development takes place in very small teams reducing the need for formality. Most development is independent of the core team.
Requirements Management	Manage requirements of project and identify inconsistencies with the project plan	Same as above
Project Planning	Establish and maintain plans that define project activities	Same as above
Project Monitoring and Control	Provide an understanding of the project's progress so that appropriate corrective actions can be taken if progress deviates from plan	No formal plans or deadlines; milestones are tied to academic year. External users are tied to their own priorities and milestones.
Configuration Management	Establish and maintain integrity of work products using configuration identification and control	Yes, tight control over program library
Process and Product Quality Assurance	Objectively evaluating adherence to process descriptions and resolving noncompliance	Tight control over contributed capabilities
Organizational Process Definition	Follow an organization-wide process	No, distributed contributors set their own processes; within the core team there is a well-established process.
Organizational Training	Develop the skills and knowledge of people so they can perform their roles effectively	An important output of this project is the training of graduate students.
Risk Management	Identify potential problems before they occur and adequately handle those problems	Long track record of successful risk management. Core code is provided "caveat emptor" to the users.
Peer Reviews	Software artifacts (requirements, design, code) reviewed by peers to improve quality	Code is reviewed by Co-PI before inclusion in program library.
Continuous Integration	Integrate and build system many times a day, each time a task is completed	No
Refactoring	Restructuring of software to remove duplication, improve communication, simplify, or add flexibility	Yes, in areas where new code is being developed
Retrospective	Post-iteration review of the effectiveness of the work performed, methods used, and estimates	No
Tacit Knowledge	Project knowledge is maintained and updated in participants' heads rather than in documents	Yes, a great deal is published, but tacit knowledge is important. There are more than 5,000 papers published based on results obtained from NENE calculations.
Collective	Anyone can change any code, any-	No

Practice	Description	Degree Followed
Ownership	where in the system, at any time	
Test-Driven Development	Module or method tests are written before and during coding	Yes, even the core team is one of users
Feature-Driven Development	Establish overall architecture and feature list, then design-by-feature and build-by-feature	Yes, in the sense that the project is driven by new features

4 Code Life Cycle and Workflow Management

Much of the code in the NENE program library is in maintenance mode, but since NENE is always under continuous enhancement, there is always new development and testing. At any given time a module in the program library may be from one month to 25 years old. It is important to the NENE team that any new capability not be allowed to break an old capability. Literally thousands of users rely on the correctness of the core code. The workflow for this project is typical of the other projects that we have studied, consisting of the following steps:

- question formulation
- formulation of development approach
- code development
- testing (verification and validation)
- production
- analysis
- hypothesis formulation

(This is not the customary language of software engineering—i.e., requirements gathering, specifications formulation, etc.—but is more descriptive of the actual workflow management process employed by the projects we have studied.)

The development path through these steps is iterative [Wikipedia 2006]. There is some refactoring of old modules as capabilities are replaced or enhanced. Typically two major releases of the NENE code occur each year. Approximately six minor releases constitute a major release. Consequently, the NENE code, like the CONDOR code, has seen dozens of releases in its lifetime. One of the most important drivers of the release schedule is that much of the work is done by students who need to graduate and post docs and assistant professors who need to publish.

One of our goals in our case studies has been to document the tools used by the code development teams; for NENE these are summarized in Table 2.

Table 2: NENE Life Cycle Management Tools

Code Development Environment	
Compilers/Interpreters	Fortran, C
Scripts	C shell
Debuggers	Print+FTNCHEK
Performance Monitoring	NetPIPE, History of Runs (numerical results and iteration counts, CPU time and parallel scaling)
Editors	Vi, emacs
Development Environments	UNIX, scripts
Execution Environment	
Element/Grid Generation	Sets of basis functions
Visualization	Desktop-based in-house tools (Mac, Windows, or Linux)
Data Analysis	Desktop-based in-house tools (Mac, Windows, or Linux)
Code Development Process Tools	
Configuration Management	manual
Bug Tracking	No formal tools deployed
Code Documentation	User's manual, code documentation, Web site
Support Libraries	
Computational Mathematics	BLAS
Parallel Programming Libraries	In-house API supporting TCP/IP, MPI, LAPI, SHMEM

The NENE code validation strategy is tied to experiments and data that reside in National Institute of Standards and Technology (NIST) databases. Like the HAWK team, the NENE team has established a very restrictive metric for the level of agreement between serial and parallel runs of the same problem. This is one of the few codes in our experience to do this.

There is a very large regression test library available to confirm the proper execution of the many features of the NENE code.

Within the core team, development is done in batch mode; there is no interactive debugging. The core team uses the tried-and-true approach of reducing the complexity of the problem to trap bugs. Another solution is "to use an old computer (an AXP processor) which generates a core dump on every floating point error—much better than doing a fix-up and trying to run past the original problem," according to the NENE Co-PI. Bug logs are listed at the beginning of code modules. Bugs that are reported by program alumni are taken especially seriously. Some members of this community analyze the error, identify its likely source, and sometimes even propose a fix. The NENE core team often develops a scalar version of the capability first, then parallel versions.

There are now codes in the program library that only exist in parallel versions; they can be run with just one processor, but are not what a serial implementation would have turned out to be. The NENE Co-PI made a comment that has been echoed by all of our case study subjects: “Parallel debugging is difficult” (see Chapter 5, “Lessons Learned,” item *f*).

5 Lessons Learned

The NENE case study has reinforced some lessons learned in previous studies and provided some new ones:

a. Scientific code teams value "agility" over process; rigid software management approaches usually are avoided, but planning and adoption of useful software practices *are* important to the success of scientific software development projects.

b. Maintainability and portability are essential.

c. Risk management is important to the success of technical software development projects—even those with a research orientation.

d. Ultimately customers are critical to the long-term success of these codes.

e. Code teams can function well with a minimum of process so long as there is adequate planning and good communication among team members.

f. Parallel debugging is hard.

g. Parallel performance (in fact, performance in general) is driven by the science, and not the other way around.

h. The focus of a scientific research code is the need to do new scientific research, not achieve impressive performance or demonstrate new computer science. The NENE team follows the words of Voltaire: "The better is the enemy of the good."

References

[Agile Alliance 2006]
Agile Alliance. See http://www.agilealliance.com/. URL valid as of January 2007.

[Kendall 2005a]
Kendall, R. P., Jeff Carver, Andrew Mark, Douglass Post, Susan Squires and Dolores Shaffer, "Case Study of the Hawk Code Project," LA-UR-05-9011, Los Alamos National Laboratory, November, 2005.

[Kendall 2005b]
Kendall, R. P., Andy Mark, Susan Squires, and Douglass Post, "Case Study of the Condor Code Project," LA-UR-05-9291, Los Alamos National Laboratory, December, 2005.

[Kendall 2006]
Kendall, R. P., Jeff Carver, Susan Squires, and Douglass Post, "Case Study of the Eagle Code Project," LA-UR-06-1092, Los Alamos National laboratory, August, 2006.

[Post 2005]
Post, D. E., R. P. Kendall and E. M. Whitney, "Case Study of the Falcon Code Project," 2nd Workshop on HPC Applications," ACM/IEEE International Conference on Software Engineering, St. Louis, MO, May 22, 2005.

[SEI 2006]
Software Engineering Institute, curriculum module.
See http://www.sei.cmu.edu/publications/documents/cms/cm.012.html. URL valid as of January 2007.

[SourceForge 2006]
SourceForge. See http://sourceforge.net/projects/ftnchek/. URL valid as of January 2007.

[Wikipedia 2006]
Wikipedia. See http://en.Wikipedia.org/wiki/Iterative_development. URL valid as of January 2007.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	December 2006	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Case Study of the NENE Code Project	FA8721-05-C-0003

6. AUTHOR(S)

Richard Kendall (Software Engineering Institute); Douglass Post (DoD HPCMPO); & Andrew Mark (DoD HPCMPO)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2006-TN-044

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The Defense Advanced Research Projects Agency (DARPA) High Productivity Computing Systems (HPCS) Program is sponsoring a series of case studies to identify the life cycles, workflows, and technical challenges of computational science and engineering code development that are representative of the program's participants. A secondary goal is to characterize how software development tools are used and what enhancements would increase the productivity of scientific-application programmers. These studies also seek to identify "lessons learned" that can be transferred to the general computational science and engineering community to improve the code development process.

The NENE code is the fifth science-based code project to be analyzed by the Existing Codes subteam of the DARPA HPCS Productivity Team. The NENE code is an application code for analyzing scientific phenomena and predicting the complex behavior and interaction of individual physical systems and individual particles in the systems. The core NENE development team is expert, agile, and of moderate size, consisting of a professor and another permanent staff member, five post docs, and 11 graduate students. NENE is an example of a distributed development project; the core team is anchored at a university, but as many as 250 individual researchers have made contributions from other locations.

14. SUBJECT TERMS	15. NUMBER OF PAGES
High Performance Computing, Verification and Validation, Software Project Management, Case Studies	25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL