Defining the Discipline of Secure Software Assurance: Initial Findings from the National Software Assurance Repository

Dan Shoemaker
Jeff Ingalsbe
Nancy Mead
Rita M. Barrios

February 2011

ABSTRACT: Defect free software is a critical national priority. Yet, we still do not fully understand the shape of the field that underlies the process of producing, sustaining and acquiring secure software. Specifically, there is no common agreement on the knowledge requirements for the field, nor is there even full agreement about the activities that legitimately comprise the process itself. Recognizing this, the Department of Defense, through the National Security Agency, has begun a three-year study to characterize the form and contents of the discipline of software assurance. This type of rigorous study is a necessary first step in formulating an academic study of the field. It is also a prerequisite to formulating the practical steps necessary to achieve a secure software base. The first phase of the project, which has just been completed, created a database containing the known empirical, theoretical, critical/analytic and methodological knowledge elements of the field. This report utilizes that database to characterize the current state of secure software assurance work and suggest future directions.

ESTABLISHING AN IMPORTANT NEW FIELD

It is conventional wisdom that "commonly used software engineering practices permit dangerous errors...which enable hundreds of attack programs to compromise millions of computers every year" [10]. This happens because "commercial software engineering today lacks the…rigorous controls needed to produce high-quality, secure products at acceptable cost" [10, p. 39]. As a result, the hidden cost to the U.S. economy is around $60 billion dollars a year [5]. But the real concern is that the exploitation of a software defect in a basic infrastructure component, like power or communication, could lead to a national tragedy like 9/11 [1].

Given the potential impact of insecure code on our national well-being, it is a matter of extreme national interest to do something about the problem. That is the purpose of the "National Strategy to Secure Cyberspace [1]. Among the three

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

highest priorities in that report is the creation of a National Cyberspace Awareness and Training Program [1, p.37]. This priority recognizes two of the barriers to the improvement of cybersecurity as "a lack of familiarity, knowledge, and understanding of the issues" and "an inability to find sufficient numbers of adequately trained...personnel to create and manage secure systems" [1]. One of the priority's major initiatives is to "foster adequate training and education programs to support the Nation's cybersecurity needs" [1]. The implicit obligation in that mandate is to ensure that the workforce understands and can apply correct secure software assurance practice in their day-to-day activities [3].

Nonetheless, that noble goal has proven to be easier said than done, since half of the defects identified in US-CERT's history have appeared in the past three years. That increase might be expected given the size of the workforce and the overall scope of cyberspace. However, it does not show that we have gotten our arms around the problem yet. What is needed is a better understanding of how to ensure a basic, responsible level of practice for all of the potential participants in the field [6]. Then we need to get that knowledge out to the workforce.

The traditional means of disseminating that kind of critical knowledge is formal education and training programs [3]. The National Strategy formalizes that approach in Action Recommendation 3-7 [1, p. 41]. Accordingly, it would seem logical to begin to start teaching software assurance topics in conventional training and education settings around the Country. The problem is that there is currently no single, authoritative point of reference to "guide the development and integration of education and training content relevant to software assurance" [6, p. xiv.]. The National Strategy recognizes that problem in Action Recommendation 3-6 [1, p.40]. The recommendation of A/R 3-6 is that "a public-private partnership should…provide input into…the development and dissemination of best practices for cybersecurity" [1, p. 41]. That includes secure software assurance.

It is generally agreed that, "secure software assurance involves a range of traditional disciplines, which include software engineering, systems engineering, and project management" [4]. As a result, secure software assurance topics are taught in many different ways in a number of different places. That includes such disconnected fields as "software engineering, systems engineering, information systems security engineering, safety, security, testing, information assurance, law and project management" [4]. It is generally agreed that "software assurance is not a separate profession, so software assurance processes and practices have to be integrated into the body of knowledge of each contributing discipline" [6, 8].

What is not agreed on, however, is the precise relationship between the elements of the body of knowledge for software assurance and the traditional knowledge

elements in each relevant discipline [8]. The ability to characterize that relationship would ensure that the most relevant and correct software assurance content was embedded in each discipline [8]. The practical result would be that; "professionals in several related disciplines, such as software engineering (including its many sub-disciplines) systems engineering and project management could better understand and apply the competencies required to produce, sustain and acquire secure software into their work" [6, p. xiv].

IDENTIFYING AND CATALOGING THE CONTENTS OF THE FIELD OF SOFTWARE ASSURANCE

The logical first step in organizing and presenting the contents of the field of secure software assurance would be to identify, relate and catalogue all relevant knowledge into a single repository. That is the goal of the National Software Assurance Repository (NSAR). This repository is the first fruits of a three year, Department of Defense (DoD) funded project that is being conducted under the auspices of the National Security Agency. The overall aim of the NSAR project is to identify, relate, catalogue and then disseminate, a fully validated collection of software assurance, knowledge elements, for mainstream education and training applications. This extensive classification effort was deemed necessary as an initial condition for going forward because the field is currently so ill defined.

Accordingly, the aim of the NSAR architecture is to ensure a coherent organization of all related ideas within the generally recognized categories of software work, such as acquisition, development, maintenance and operations. These categories are then further sub divided based on the concepts embodied in two internationally accepted standards for lifecycle process definition. These are the ISO/IEC 12207 (IEEE 12207.0) software lifecycle and ISO/IEC 15288 system lifecycle standards. Secure software assurance content that can be established as relevant to a given lifecycle category was catalogued in that category and then associated with all of the other pertinent content within the NSAR database using the relationships established by those standards. For instance, verification, which is a supporting process, is related to development as a primary process through the conceptual framework of the 12207 model. The knowledge contained in those categories is derived from the current state of the practice. This knowledge is kept in a database that attempts to incorporate and relate all of the commonly accepted principles, methodologies and tools for software assurance that could be identified from private and public sector sources. The task of actually maintaining that knowledge and the requisite interrelationships is handled by the NSAR's automated knowledge management system.

The NSAR is maintained at the University of Detroit Mercy, which is a National Security Agency Center of Academic Excellence in Information Assurance Education (CAE/IAE). The repository has attempted to identify and collect all formal academic, industry and governmental publications that contain recommendations regarding best practices for ensuring secure software assurance. The effort is meant to be as evolutionary and inclusive as possible as the literature of the field expands, or new sources are identified. Practices are applicable in all Software lifecycle environments and within all sectors. We made no effort to judge the correctness of the practices we incorporate, nor did we limit the search to a single area or discipline in the field. What was collected can be plainly related to some commonly accepted area of software development, acquisition and sustainment.

The knowledge in the repository was derived from all governmental, industrial and academic sources that could be identified and accessed. It is an indexed compendium of best practices that document the relevant principles and practices for secure software assurance. The knowledge represents as complete a set as possible, of the principles and concepts related to secure development, acquisition and sustainment of software. It supports the creation of an appropriate pedagogy and individualized content for dissemination into higher education. It will be possible to specify real-world policies and procedures for any operational development, acquisition, or sustainment project using this repository.

The repository also categorizes and validates as many of the relevant lifecycle methodologies and tools for secure software assurance as could be identified. As a result, it contains a fully validated set of common methodologies for secure coding as well as a collection of related supporting tools that might be deployed to support those methodologies. It represents a validated inventory of related supporting tools and methods that would allow any organization to increase the assurance and security of software that is purchased from outside vendors, as well as software that is developed and sustained internally.

THE NATIONAL SOFTWARE ASSURANCE REPOSITORY (NSAR) PROJECT METHODOLOGY

The NSAR is the first phase of a multi-phase project that is designed to establish a discipline of secure software assurance. It contains the results of a six-month literature review. That review was conducted by thirteen PhD researchers, who worked under rigorous supervision. The intent was to identify and document any published knowledge that could be related to the assurance of software. That includes knowledge from all of the recognized computing disciplines, such as

computer science, software engineering and information systems, as well as all of the other traditional thinking in computer security, such as the work of Saltzer and Schroeder. It also includes any knowledge from the behavioral and social sciences that could conceivably be related to the assurance of secure software. There were no limitations placed on how far back the search was conducted. So the earliest citations in the NSAR date from the mid 1960s.

The knowledge in the repository was formally related to two commonly accepted models for the software process. These are IEEE/EIA 12207.0 "Standard for Information Technology-Software Life Cycle Processes" and ISO/IEC 15288, "Standard for Systems Lifecycle Processes". In addition, during the data entry process each knowledge element was related to the Department of Homeland Security, National Cyber Security Division (NCSD) Common Body of Knowledge to Develop, Sustain and Acquire Secure Software (CBK). The indexing of the knowledge elements to these three models is relevant to the discussion in the next section. The NSAR currently contains approximately 1,500 individual citations constituting approximately 50 gigabytes of data. This material is fully searchable by keyword and will eventually be made accessible through a sophisticated internet facing knowledge management system that allows for heuristic searches of the entire database.

INITIAL FINDINGS FROM THE NATIONAL SOFTWARE ASSURANCE REPOSITORY (NSAR)

The results presented here were produced as an initial attempt to lay out the general shape of the field. These will be presented in the order that we asked the question. These particular queries were designed to address the early questions that were asked us as our project and its aims became known. We believe that the intent of these questions serves as an introduction to the issues that influence the entire problem space since the queries were aimed at further refining our and other people's understanding of the overall justification and purposes of the project.

HOW WELL DO WE REALLY UNDERSTAND THE FIELD?

The first question was "How well do we actually understand the field?" In its general level of understanding software assurance suffers from the "six blind men and the elephant syndrome". In that old story, six people hold onto various parts of an elephant and make guesses about its form based on the part they are holding onto. Since they can't see the whole beast, whatever part they are touch-

ing becomes the elephant. So, the one holding the tusk thinks it's a spear and the one holding the trunk thinks it's a snake and so on.

In general, this same situation affects software assurance. That is, programmers see the steps that need to be taken to ensure against exploitable defects as a coding problem. Designers see security as a design issue and managers see it as a process problem. The lack of a complete and fully integrated understanding of the form and boundaries of the field and the commensurate steps to be taken is particularly dysfunctional when it comes to attempting to guarantee comprehensive assurance. That is because the security approach has to be comprehensive in order to ensure that all of the bases have been touched. The lack of a common definition of what those bases are brings the integrity of any assurance process into question. Therefore, having a well-defined set of steps within a commonly agreed on area of focus is an important pre-condition for any security process.

In order to better characterize how uniform our understanding of the field is, we did a simple computer based query of all of the citations in the database in order to see how many terms are used to characterize the secure software assurance process. Our assumption was that the term would reflect the way the user understood the process. The assumption was that the presence of a few clear synonyms would represent a more integrated, or coherent understanding of the process. However, if there were a large number of synonyms it would reflect a lack of agreement about the boundaries and components of the field,

The query sought to identify any term or phrase with security, or assurance in it, or associated in a sentence with security, or assurance. What we found was an astonishing seventy-seven synonyms for the general process of ensuring secure software (the raw data for this query is presented in Appendix 1). It should be noted that this is a top-level view and our interpretations of what might actually characterize the software assurance process could be debated. However, the large number of synonyms for the activities that are encompassed within the goal of secure software clearly indicates that we still have a way to go before we can say that we have a consistent and cohesive understanding of the field.

WHOM ARE WE ADDRESSING?

The second question looked at whom we are addressing. Ideally, all of the audiences involved with software assurance would be equally represented in the literature. This would indicate that knowledge and advice was flowing to everybody who might have a stake in producing secure software. It would also indicate where the interest is in that topic. Therefore, we did a query to identify

the target audience for each of the 1,500 articles in our database. What we found is displayed in Table One below.

Table One. Percentage of Articles by Intended Audience

Category	%
Academia	21.3
Academia and Practitioners	20.7
Acquisition Community	0.1
Development Community	13.3
Executive Decision Makers	0.6
Government Defense	1.2
Government Other	0.6
Middle Management	1.6
Practitioners	13.6
Programmers	4.8
Software Assurance Community	13.1
Software Engineers	8.3
Software Project Managers	0.3
Standards Community	0.4

What we found was an overwhelming number of articles written by or addressing the academic community. This is probably not surprising given the fact that academics are much more oriented toward publication than practitioners are. But the lack of interest in engaging in discussion, by people in key roles like development and executive decision-making, could also indicate a commensurate lack of interest in the general topic of secure software. The problem that this table

describes is more one of awareness than education. However, it does indicate that the concept of secure software is not necessarily being discussed among the various constituencies that are part of the field.

In fact, the impression that is gained by working with the large amount of literature that is in the NSAR is that each major constituency is proceeding along as if their understanding of the field is the correct one without reference to the ideas of other constituencies. The reality that most of the discussion of the topic is coming out of academe, whereas most of the actual leadership in the field of software assurance should be in applying these ideas in government and industry, is particularly disturbing. If what is indicated by the apparent lack of discussion is true, then the people who are actually responsible for putting secure software knowledge into day-to-day practice are either not listening to any ideas but their own. Or even worse, they are not thinking about the topic at all.

DISCUSSION OF SECURE SOFTWARE TOPICS BY SOFTWARE LIFECYCLE CATEGORY

Next, it seemed relevant to investigate precisely what lifecycle categories have had the most discussion over the past 40 years. The IEEE/EIA 12207.0 Standard is the commonly accepted definition of the software lifecycle. Its seventeen areas all have some relevance to the production of secure software. Therefore, it seemed reasonable to see which one of these areas has had the most discussion. Obviously, the areas that are discussed the most are the ones that are most likely to benefit from any new ideas about security, while a lack of discussion in an area might indicate that it is not given proper consideration when it comes to implementing secure software assurance practice. The table below indicates the distribution of literature by 12207 processes.

Table Two. Percentage of Articles by 12207 Process Categories

Category	%
Development processes	39.8
Documentation processes	0.6
Improvement Processes	0.3
Infrastructure Processes	10.8

Maintenance Processes	0.3
Management Processes	2.0
Operation Processes	0.2
Problem Resolution Processes	0.8
Quality Assurance Processes	20.3
Supply Processes	5.6
Training Processes	10.3
Validation Processes	0.8
Verification Processes	0.3
Audit Processes	6.4
Configuration Management Processes	0.8

The area that has gotten the most treatment in the discussion of software security is the development primary process. That should be expected since most of secure software assurance knowledge comes down to ensuring that defects are not created in the first place. However, the areas that ensure that any defect that is created is identified, documented and properly addressed are the eight supporting processes of the standard. These are documentation (0.6), configuration management (0.8), software quality assurance (20.3), verification (0.8), validation (0.3), joint review (0.0), audit (6.4) and problem resolution (0.0). Although these eight processes comprise 47 percent of the 12207, categories and arguably these processes constitute the difference between a capable organization and an incapable one the total discussion in these areas amounts to only 29.2 percent.

In that respect, when it comes to security none of these areas have been adequately considered or discussed. Obviously, quality assurance, which accounts for 20% of the total 29% of the discussion has always been an important part of defect prevention. However, the lack of explication of configuration and change management in particular indicates that the field has a long way to go in managing changes to deployed systems and that can be a critical problem since most

operational security is maintained by patching. The lack of discussion of the actual areas that implement and enforce software quality assurance, specifically inspections, reviews and tests is also indicative of the current problem.

It is likely that some of that discussion is done in the articles on secure development. But it is also likely that whatever is done in the way of introducing ideas for security verification and validation for development has been along the lines of current practice. In both cases, these areas create the necessary visibility that assurance rests on and so it is essential that ideas for their execution have their own separate discussion. There are probably a lot of innovative ways to verify and validate using secure practices however, none of these can get the proper explication if they are presented as an adjunct to the development lifecycle.

DISCUSSION OF SECURE SOFTWARE TOPICS BY SYSTEM LIFECYCLE CATEGORY

The newer ISO/IEC 15288 Standard does the same thing for systems as 12207 does for software. It is a higher-level standard in the sense that the software lifecycle categories are subsumed under it. Nonetheless, since it is system oriented the 15288 categories probably characterize the actual context of secure software the best. There is some literature that indicates that it is necessary to bring a contextual approach to the assurance process for software [2,3,4,and 5]. If that is the case then the lifecycle processes itemized in the 15288 are relevant to the discussion. The table below displays the concentration of the literature in the categories of this standard.

Table Three. Percentage of Articles by 15288 Process Categories

Category	%
Implementation Processes	57.4
Information Management Processes	6.6
Acquisition Processes	9.6
Agreement Processes	1
Architectural Design Processes	17.3

Configuration Management Processes	2
Decision Making Processes	2
Enterprise Environment Management Processes	2.5
Enterprise Processes	1.5

The implementation processes is a logical place to attract discussion however its dominance over the rest of the processes, particularly key processes like configuration management and acquisition indicates a lack of focus on the enterprise elements of the problem. The continuing indications of unbalanced focus in the discussion of the problem are most pronounced here and that lack of balance is a source of concern, to the authors at least, since it indicates that the distribution of secure software assurance knowledge is badly limited. One of the things that is crystal clear from the literature is that the field is disaggregate, in that it is very downwardly focused on specific problems rather than on strategic considerations such as the enterprise areas of the 15288 standard.

The concentration that comes through here, on technology rather than big-picture integration, is typical of the discussion in most technical fields. However in the case of a field as strategically important as secure software the lack of consideration of contextual issues would tend to indicate a lack of top-level understanding of how all of the pieces fit together. Obviously, a single-minded concentration on technology is not going to produce a comprehensive and continuously evolving approach that can be accepted by the enterprise as a whole. Nevertheless, if this particular query holds up that is exactly the direction the field is heading. This raises leadership concerns both for people who are responsible for maintaining our national security, as well as for the people who are responsible for devising effective solutions.

CONCLUSIONS

The conclusion that can be drawn from these series of queries is that the impression that the field of secure software assurance is not well defined or adequately studied is correct. The simple fact that there are 77 terms to describe the activity of software assurance itself should make that clear. However, the final table below also tends to substantiate that conclusion. The Department of Homeland Security's Common Body of Knowledge to Produce, Sustain and Acquire Software

(CBK) is intended to define secure software assurance practice. If that is the case then there should be a balanced consideration in the literature of all of the categories of that model. The actual count of references in the literature indicates otherwise.

Table Four. Percentage of Articles by CBK Category

Primary SwA CBK	%
Acquisition of Secure Software	0.0%
Ethics, Law, and Governance	3.9%
Fundamental Concepts and Principles	2.2%
Secure Software Construction	7.3%
Secure Software Design	3.4%
Secure Software Engineering Management	12.4%
Secure Software Processes	26.4%
Secure Software Requirements	2.8%
Secure Software Sustainment	6.7%
Secure Software Tools and Methods	0.6%
Secure Software Verification, Validation, and Evaluation	16.9%
Threats and Hazards	10.7%

As can be seen, two of the three entire sections of the body of knowledge, sustainment and acquisition account for 6.7% of the total literature that is devoted to the topic of security, whereas the practices associated with development account for the balance of the discussion. This is probably to be expected given the bias in the industry toward development but it also indicates that key areas of practice have simply been ignored in the conversation and one-legged stools typically do not stand very well. We obviously have more holes in our knowledge than we

have actual substance and that is not an acceptable condition if the overall purpose is to provide sound advice to practitioners and students alike.

These are very preliminary results aimed at finding out just how mature the body of knowledge is for secure software assurance. The conclusions that can be drawn from this are that there is little agreement about what constitutes the elements of the field and that there is no rhyme or reason to the discussion. One simple conclusion is that we need to work to get common agreement on how to flesh out the areas that have heretofore not gotten adequate consideration.

As an initial step, the authors suggest dedicating ourselves to three practical tasks:

- First, we have to agree on the scope and definitions of the elements of the field. Right now, it is clear that legitimate software assurance activity is defined as whatever anybody wants to define it. That is not rationally consistent with the idea of a well-defined and valid body of knowledge. As such, one conclusion that is painfully clear from the literature is that we need to set our sites on obtaining a complete and commonly accepted definition of what the actual practice of secure software assurance involves.
- Then we need to incorporate all of the knowledge necessary to ensure practice within our established boundaries. In conjunction with getting agreement on the boundaries of the field, we should also better coordinate the focus of our research so that it encompasses practical advice for the entire community, not just the areas that have the greatest "star power".
- Finally, we need to pay attention to how the message is conveyed, since we are not reaching all of the critical constituencies. The unambiguous picture of small groups of people all talking only to themselves and essentially ignoring what is going on in the other parts of the field is clear in the literature and as we have added more citations that impression has only been reinforced. Perhaps this is more of a leadership issue but we are not carrying out a coherent conversation about the problems of the field right now. That situation can change and it has to before we are going to be able to do anything coordinated about solving the problem.

BIBLIOGRAPHY

1. U.S. Department of Homeland Security, *The National Strategy to Secure Cyberspace*. Washington, DC: U.S. Department of Homeland Security, 2003. http://www.dhs.gov/xlibrary/assets/National_Cyberspace_Strategy.pdf.

2. Conklin A. and G. Dietrich, “Secure software engineering: A new paradigm,” in *Proc. 40th Annual. Hawaii Int. Conf. System Sciences (HICSS'07)*, Hawaii, 2007, p. 272.
3. O’Neill, Donald, “Business Considerations and Foundations for Assuring Software Security: Business Case Models for Rational Action”, Carnegie Mellon University, Build Security In. 2008, accessed June 2009.
4. Mead, Nancy R., Dan Shoemaker and Jeffrey Ingalsbe, “Integrating Software Assurance Knowledge into Conventional Curricula” STSC Crosstalk, January, 2008
5. Newman, Michael. Software Errors Cost U.S. Economy $59.5 Billion Annually. Gaithersburg: National Institute of Standards and Technology (NIST), 2002.
6. Redwine, S. T. Editor, “Software assurance: A guide to the common body of knowledge to produce, acquire and sustain secure software, version 1.1,” U.S. Department of Homeland Security, Washington, DC, September, 2006.
7. Saltzer, Jerome and Michael D. Schroeder, “The Protection of Information in Computer Systems”, Communications of the ACM 17, 7 (July 1974)
8. Shoemaker D., A. Drommi, J. Ingalsbe, and N. R. Mead, “A comparison of the software assurance common body of knowledge to common curricular standards, in *Proc. 20th Conf. Software Engineering Education & Training*, Dublin, Ireland, 2007, pp. 149–156.
9. “State of the Art Report, SOAR”, Information Assurance Technology Analysis Center (IATAC), Data and Analysis Center for Software (DACS), July 31, 2007, http://iac.dtic.mil/iatac/download/security.pdf.
10. President’s Information Technology Advisory Committee, “Cybersecurity: A crisis of prioritization,” Exec. Office of the President, Nat. Coordination Office for Information Technology Research and Development, Arlington, VA, Feb. 2005.

APPENDIX ONE

Terms Used to Describe Secure Software Assurance Practice

1. Affecting effective software quality management
2. Application Security
3. Architecting secure distributed software applications
4. Architecture for software Security
5. Assurance for Security protocols
6. Assurance of critical systems
7. Assurance of software
8. Building secure software
9. Building intrusion-tolerant secure software
10. Building secure application software
11. Building secure applications
12. Computer Security
13. Constructing Secure Software
14. Constructing secure systems
15. Creating secure applications
16. Data Security
17. Design of secure architectures
18. Design of secure information systems
19. Developing secure software
20. Developing secure systems
21. Distributed systems Security of software
22. Enterprise software Security
23. Formal product assurance
24. High assurance IT Security
25. High-assurance secure systems development
26. Holistic Security management
27. Holistic Security requirements engineering
28. Improving Security worthiness
29. Information assurance
30. Information Security
31. Information Security governance
32. Integrating Security design into the software development process
33. Life cycle assurance disaster planning

34. Managing software quality through IT Security issues
35. Model-based Security engineering
36. Modeling Security requirements
37. QA programming
38. Quality assurance
39. Quality assurance approach for security
40. Secure application software methods
41. Secure coding
42. Secure interface design
43. Secure programming
44. Secure software architecture
45. Secure software awareness
46. Secure software development
47. Secure software engineering
48. Secure software systems analysis
49. Secure software testing
50. Secure systems development
51. Security architecture
52. Security assurance
53. Security assurance
54. Security in systems
55. Security management
56. Security management for software product lines
57. Security modeling
58. Security principles
59. Security quality requirements engineering
60. Security related design
61. Security requirements engineering
62. Security requirements management
63. Security risk modeling
64. Software artifacts Security
65. Software assurance
66. Software engineering and assurance
67. Software Security
68. Software Security assurance
69. Software Security evaluations
70. Software Security knowledge
71. Software Security metrics
72. Software Security risk management

73. Software systems trustworthiness
74. Systems Security
75. Systems Security engineering
76. Usability assurance
77. Web engineering Security

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM-0001120