A Software Engineering Body of Knowledge Version 1.0

Thomas B. Hilburn
Iraj Hirmanpour
Soheil Khajenoori
Richard Turner
Abir Qasem

April 1999

TECHNICAL REPORT
CMU/SEI-99-TR-004
ESC-TR-99-004

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

A Software Engineering Body of Knowledge Version 1.0

CMU/SEI-99-TR-004
ESC-TR-99-004

Thomas B. Hilburn
Embry-Riddle Aeronautical University
Iraj Hirmanpour
Embry-Riddle Aeronautical University
Soheil Khajenoori
Embry-Riddle Aeronautical University
Richard Turner
Federal Aviation Administration
Abir Qasem
Embry-Riddle Aeronautical University

April 1999

Software Engineering Process Management

This work is sponsored by the Federal Aviation Administration. 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 F19628-95-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 52.227-7013.

This document is available through Asset Source for Software Engineering Technology (ASSET): 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 or toll-free in the U.S. 1-800-547-8306 / FAX: (304) 284-9001 World Wide Web: http://www.asset.com / e-mail: sei@asset.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218 / Phone: (703) 767-8274 or toll-free in the U.S.: 1-800-225-3842.

Table of Contents

List of Figures

Acknowledgments

Work on this report was done under contract in association with the U.S. Federal Aviation Administration (FAA).

The authors would like to thank the following individuals for reviewing this report:

Barry Boehm (University of Southern California)

John Brackett (Boston University)

Robert Cannon (SEI)

Andrew Kornecki (Embry-Riddle Aeronautical University)

Nancy Mead (SEI)

Marsha Pomeroy-Huff (SEI)

Barbara Saragovitz (FAA)

Mary Lou Schaallman (Cambria Consulting)

Massood Towhidnejad (Embry-Riddle Aeronautical University)

We would also like to acknowledge Karen Forte of Embry-Riddle Aeronautical University and Arnold Smith of TRW, who provided administrative support for the FAA contract work.

For further information about the topics discussed in this report, please contact

Thomas B. Hilburn

Department of Computing & Mathematics

Embry-Riddle Aeronautical University

Daytona Beach, Fl 32114

1-904-226-6889

hilburn@db.erau.edu

Foreword

One of the characteristics of a mature profession is a body of knowledge. When the Federal Aviation Administration initiated a project to improve the software engineering competencies of its technical and management staff, they were unable to find such a body of knowledge. The result was a decision to develop a software engineering body of knowledge to use in defining competencies and establishing a curriculum. This report presents that body of knowledge. It can be of assistance to industries that increasingly need to assess and improve the software engineering capabilities of their employees. In addition, academic institutions may find the information useful as they define curricula for software engineering.

The Software Engineering Coordination Committee (SWECC), sponsored jointly by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronic Engineers (IEEE) Computer Society, is tasked with developing an international standard body of knowledge for software engineering. The work described in this report will contribute to the SWECC's task and represents the ongoing development of software engineering and its maturation as a discipline.

Robert L. Cannon
Software Engineering Institute
April 1999

Abstract

Software engineering, both as a discipline and as a profession, is at a pivotal point in its evolution. Although software has become critical in the development of most new human-created systems, the concepts, principles, and methods for engineering software are still neither well defined nor uniformly agreed upon. The lack of consensus regarding software engineering practice and the requisite competencies creates confusion and has serious consequences for the evaluation, acquisition, and application of software engineering knowledge. This report presents an effort to organize and catalogue a body of knowledge for software engineering and to provide a systematic, concise, and complete description of the software engineering discipline. This body of knowledge can assist organizations in defining and improving the software engineering competencies of their workforces; it can help educational institutions in defining software engineering curricula; it can provide a basis for classifying academic and industrial research and development efforts; and it can improve the understanding and practice of software engineering.

1 Introduction

"Knowledge is of two kinds: we know a subject ourselves, or we know where we can find information upon it."

-Samuel Johnson

Software is playing an increasingly important and central role in all aspects of daily life—in government, banking and finance, education, transportation, entertainment, medicine, agriculture, and law. The number, size, and application domains of programs being developed have grown dramatically; as a result, billions of dollars are being spent on software development, and the livelihood and lives of millions directly depend on the effectiveness of this development. Unfortunately, there are severe problems in the cost, timeliness, and quality of many software products; even more serious is the effect that quality problems can have on the safety-critical elements of software that directly affect the health and welfare of humans.

It has been almost 30 years since the first organized, formal discussion of software engineering as a discipline took place at the 1968 North Atlantic Treaty Organization (NATO) Conference on Software Engineering. The term "software engineering" is now widely accepted by industry, government, and academia: many thousands of computing professionals go by the title "software engineer"; numerous publications, groups and organizations, and professional conferences use the term "software engineering" in their names; and there are many educational courses and programs on software engineering. However, in spite of this widespread acceptance, software engineering does not have a well-defined and universally accepted professional content. Some would even dispute that it is engineering. A more generous attitude might be that the discipline is new relative to the more traditional engineering fields and, thus, software engineering simply is not yet fully mature. In spite of this immaturity (or maybe because of it), there is a crucial need to define the knowledge and competencies required by software professionals.

In recent years, there have been a number of studies and commentaries on the state of the software engineering profession [Ford 96, Gibbs 94]. All seem to agree that the profession is not mature. In their study of the software engineering profession, Ford and Gibbs [Ford 96] designate eight infrastructure components that can be used to evaluate a mature profession: initial professional education, accreditation, skills development, certification, licensing, professional development, a code of ethics, and a professional society. In 1996, their evaluation of the software engineering profession was low for all components except "professional development." Although there has been progress in recent years, a significant problem in advancing the state of the software engineering "infrastructure components" is the lack of a clear and comprehensive understanding of the nature and content of the software engineering

profession. This document attempts to address this issue by providing a hierarchical description and decomposition of a body of knowledge for software engineering (SWE-BOK).

1.1 Sources and Influences

A number of key activities and sources have influenced the direction, framework, and content of this work. Two publications that had an early influence on the structure and style of this SWE-BOK were *A Guide to the Project Management Body of Knowledge* [PMI 94], published by the Project Management Institute, *and The Federal Aviation Administration Integrated Capability Maturity Model®* [Ibrahim 97]. Also, the organization and content of two widely-recognized computing curriculum models have been of major benefit to this project: *IS'97: Model Curriculum for Undergraduate Degree Programs in Information Systems* [Davis 97] and *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force* [Tucker 91].

In 1993, the IEEE Computer Society (IEEE-CS) and the Association of Computing Machinery (ACM) established the IEEE-CS/ACM Joint Steering Committee for the Establishment of Software Engineering as a Profession. (The steering committee has recently been replaced by the Software Engineering Coordinating Committee.) In the past year, the activities of the IEEE-CS and ACM in this area have accelerated. In 1997, the steering committee appointed a task force on the software engineering body of knowledge. The task force recently published a strategy [Dupuis 98] for developing a comprehensive body of knowledge for software engineering; this is a multi-year project that is expected to be completed sometime in 2001. This strategy, along with conversations with persons working on the IEEE-CS/ACM projects, has been extremely useful in validating the content, scope, and conceptual organization of this SWE-BOK as presented in this report.

1.2 Overview

This report presents an effort to organize and catalogue a body of knowledge for software engineering and to provide a systematic, concise, and complete description of the software engineering discipline. The material includes information about the methodology and sources, as well as the anticipated uses of this body of knowledge. While the material presented here has been reviewed by a number of highly qualified experts, comments and suggestions for future revision are welcomed. Contact information is provided in the Acknowledgements section of this document.

1.3 Organization of This Report

This report is organized in four sections. Section 1 provides background and introductory information. Section 2 describes the development of this body of knowledge in terms of sources and methodologies. It also introduces the architecture selected for structuring this information. Section 3 discusses various uses for this SWE-BOK in the areas of academic, industrial, and professional practice. A body of knowledge is described in Section 4.

® Capability Maturity Model is registered in the U.S. Patent and Trademark Office.

The purpose of this report is to provide a single-source reference to the collection of knowledge that constitutes the software engineering discipline. Since software engineering is a rather young and somewhat immature discipline, there is a special need to capture and categorize the subdisciplines that collectively make up the field. Although new technologies will emerge, and new methods and techniques will surface, the fundamental building blocks of software engineering are known today. The SWE-BOK classification scheme that we propose is intended to identify those elements that are invariant, while allowing for new knowledge to be incorporated into its structure.

2 Developing a Body of Knowledge

2.1 Background

This SWE-BOK resulted from work sponsored by the U.S. Federal Aviation Administration (FAA) as part of a project to improve its software acquisition, development, and maintenance processes. This SWE-BOK provides the basis for establishing the skills and knowledge needed for FAA personnel to perform their software-intensive system acquisition, development, and maintenance roles.

There are several collateral factors that were considered in the development of this SWE-BOK. One was the role of system engineering in software engineering. Since software is typically developed to be part of more complex technological systems (that include computing and non-computing hardware, people, procedures, and processes), its body of knowledge cannot easily be isolated from the other components of a total system. For the purposes of this project and in order to reduce the complexity of our efforts, this SWE-BOK does not directly address the association and relationship between software engineering and system engineering.

In addition, this SWE-BOK does not include areas of knowledge that may be supportive of software engineering, but that do not play a direct role in software development. For example, the following discipline areas were not included in this SWE-BOK: continuous mathematics, the natural sciences, traditional engineering science, psychology, economics, and business administration.

2.2 Knowledge Architecture

The SWE-BOK knowledge architecture in this report provides a hierarchical description and decomposition of a body of knowledge for software engineering. For the purposes of this work, the term “knowledge” is used to describe the whole spectrum of content for the discipline: information, terminology, artifacts, data, roles, methods, models, procedures, techniques, practices, processes, and literature. Figure 1 shows the three levels of abstraction (knowledge categories, knowledge areas, and knowledge units) and the relationships that were used in modeling this SWE-BOK.

Three levels of abstraction were chosen to achieve a balance between simplicity and clarity, and the appropriate depth and detail of knowledge description. This architectural model was influenced by a number of other models for organizing and structuring computing knowledge. The operational definitions of terms used in our classification scheme are described in Section 2.3.

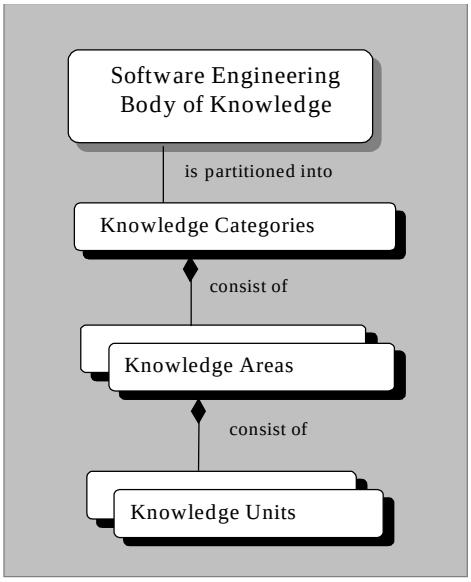

Figure 1: Levels of Abstraction in This SWE-BOK Architecture

2.3 Operational Definitions

Knowledge: A term used to describe the whole spectrum of content for the discipline: information, terminology, artifacts, data, roles, methods, models, procedures, techniques, practices, processes, and literature.

Body of knowledge (BOK): A hierarchical description of software engineering knowledge that organizes and structures the knowledge into three levels of hierarchy: knowledge categories, knowledge areas, and knowledge units.

Knowledge category (KC): A subdiscipline of software engineering that is generally recognized as representing a significant part of this body of software engineering knowledge. Knowledge categories are high-level structural elements used for organizing, classifying, and describing software engineering knowledge. A knowledge category is composed of knowledge areas.

Knowledge area (KA): A subdivision of a knowledge category that represents software engineering knowledge that is logically cohesive and related to the knowledge category through inheritance or aggregation. A knowledge area is composed of a set of knowledge units.

Knowledge unit (KU): A subdivision of a knowledge area that represents a basic component of software engineering knowledge that has a crisp and explicit description. For the purposes of this activity, the knowledge unit is atomic; that is, it will not be subdivided into simpler or more basic elements.

3 Using This Body of Knowledge

This SWE-BOK can be used for a variety of purposes: it can help to solve problems and address issues in industrial, academic, and professional settings. Most importantly, it can serve as a general model for understanding and describing the software engineering profession.

3.1 Industrial Use

Although software engineering has become a major industrial and commercial activity in the last 20 years, the nature and content of software engineering is still unclear in many of the organizations involved in software development and acquisition. This SWE-BOK can help such organizations to examine, categorize, and organize their software activities and can lead to better definitions of software roles and software processes.

Software organizations that want to assess the knowledge of their engineers could use this SWE-BOK in designing a competency evaluation system. Such a competency system could use this SWE-BOK to help identify and judge what kind of software engineering knowledge is required to accomplish the tasks associated with individual software-related roles. Such an assessment might be coupled with the use of this SWE-BOK to design training programs and to develop an overall effort to improve an organization's software processes.

3.2 Academic Use

If the practice of software engineering is to advance, it is critical that the initial preparation for software engineers be improved. Currently, most entry-level software engineers receive their initial preparation in academic computing programs (computer science, computer engineering, and information systems) or in non-computing engineering programs, such as electrical engineering. There are an increasing number of academic institutions that are interested in developing software engineering degree programs. In some cases, the programs will be created as entirely new programs, but in most cases, they will represent the conversion of an existing program in computing to one that emphasizes software engineering. Unfortunately, many computing faculty members have little preparation or experience in software engineering. This SWE-BOK can provide faculty with basic information about software engineering knowledge, and this will support the development of education curricula in software engineering.

In addition, this SWE-BOK supports the development of more general guidelines for the development and accreditation of software engineering programs by groups such as the Accreditation Board for Engineering and Technology (ABET) and the Computing Sciences Accreditation Board (CSAB). This SWE-BOK can also serve as a framework that can be used to conduct research into software engineering methods, techniques, and practices. Research organizations could use this SWE-BOK to help organize, classify, and guide inquiry into soft-

ware engineering. Finally, this SWE-BOK can also be used as a framework for designing software engineering curricula.

3.3 Professional Use

This SWE-BOK provides the means for studying and addressing a number of issues related to the software engineering profession. First, a body of knowledge could be used to define the profession itself or, at least, to delineate the knowledge associated with the profession. Such a definition is essential to the maturation of a discipline and is a necessary step in determining the professional standards and procedures required for the effective practice of software engineering. The definition and the professional standards could then be used to develop criteria and assessment instruments for the certification and licensing of software engineers. In addition, this SWE-BOK could be used by individual engineers to assess their own knowledge about the software engineering profession and to provide a framework that they can use to plan their professional development.

4 A Software Engineering Body of Knowledge

The following sections provide descriptions of each of the knowledge categories, and the corresponding knowledge areas and knowledge units of which they are composed. This body of knowledge has varying levels of decomposition and exposition, depending upon the current state of knowledge in a particular component.

There are several decisions about the organization and content of this SWE-BOK that require comment:

- The knowledge areas in the Software Domains category are not partitioned into knowledge units. We felt that this level of detail was sufficient for the purpose of this project, since much of the knowledge for each Software Domain area is already included in the Software Product Engineering category.
- There were a number of software engineering subjects that were considered for incorporation in this SWE-BOK, but were not included because we felt that they were covered sufficiently in other knowledge areas or knowledge units. For instance, formal methods was not included in this SWE-BOK, but knowledge about it was covered in the Mathematical Foundations area in the Computing Fundamentals category, and the Verification and Validation knowledge unit in the Software Quality Management knowledge area.
- Initial versions of this document included a great deal of information about specific software engineering techniques and tools. After review and analysis, we concluded that this type of knowledge would be difficult to keep current and complete, and was not needed for the objectives of this project.
- Although software testing is typically considered to be part of verification and validation, we chose to present the two in separate components (software testing in the Software Product Engineering category and verification and validation in the Software Quality Management knowledge area). This is in deference to the prominent role that software testing currently plays in the product development life cycle.

Each of the following sections, numbered 1 through 4, contains a complete description of the software engineering knowledge categories.

- The Computing Fundamentals knowledge category is concerned with knowledge, concepts, theory, principles, methods, skills, and applications of computing that form the foundation for the development of software and the discipline of software engineering.
- The Software Product Engineering knowledge category is concerned with a well-defined and integrated set of activities to produce correct, consistent software products effectively and efficiently. Software product engineering includes the technical activities of producing a software product, such as requirements engineering, design, coding, and test.

- The Software Management knowledge category deals with the concepts, methods, and techniques for managing software products and projects. Software management includes activities concerned with project management, risk management, software quality, and configuration management.
- The Software Domains knowledge category concerns knowledge about specific domains that involve computing and software engineering application or utilization. This category includes the following software domains: artificial intelligence, database systems, human-computer interaction, numerical and symbolic computing, computer simulation, and software acquisition.

Each category description is followed by descriptions of the knowledge areas in that category. Area descriptions are followed by descriptions of the knowledge units belonging to each of the knowledge areas.

In the references section of this document, we identify the principal literature sources for our research efforts. Within the various parts of this SWE-BOK, we have cited references that are appropriate for a particular knowledge category, knowledge area, or knowledge unit. However, there are four references that provide comprehensive information about software engineering knowledge [Dorfman 97, Marciniak 94, Pressman 97, Sommerville 95]. These references were used as sources for describing knowledge in all of the knowledge categories and in almost all of the knowledge areas and knowledge units. Consequently, for the sake of brevity, we have not included individual citations for these four references within the body of this SWE-BOK.

Computing Fundamentals

Category Number	1
Category Name	**Computing Fundamentals**
Category Description	This category is concerned with knowledge, concepts, theory, principles, methods, skills, and applications of computing that form the foundation for the development of software and the discipline of software engineering.
Knowledge Areas	4.6 Algorithms and Data Structures 1.2 Computer Architecture 1.3 Mathematical Foundations 1.4 Operating Systems 1.5 Programming Languages
References	[Aho 92], [Brookshear 94], [Denning 89], [Lethbridge 98], [Tannebaum 90], [Tucker 91], [Woodcock 88]

Description of the Computing Fundamentals Knowledge Areas

KA Number	1.1
KA Name	**Algorithms and Data Structures**
KA Description	This area is concerned with basic data structures, abstract data types, recursive algorithms, algorithm analysis, sorting and searching, complexity and computability issues, problem-solving strategies, and parallel and distributed algorithms.
KA Number	1.2
KA Name	**Computer Architecture**
KA Description	This area is concerned with methods of organizing efficient, reliable computing systems. It includes digital logic and digital systems, machine-level representation of data, assembly-level machine organization, processor architecture, memory system organization, interfacing, communications and networks, and alternative architectures.
KA Number	1.3
KA Name	**Mathematical Foundations**
KA Description	This area is concerned with the mathematical foundations of computing. This area includes mathematical logic, proof systems, discrete mathematical structures, formal languages, combinatorics, and probability and statistics.
KA Number	1.4
KA Name	**Operating Systems**
KA Description	This area is concerned with the history, evolution, philosophy, design, implementation, and use of computer operating systems. This area includes elements such as tasking and processes, process coordination and synchronization, scheduling and dispatch, physical and virtual memory organization, device management, file systems and naming, security and protection, and distributed and real-time systems.
KA Number	1.5
KA Name	**Programming Languages**
KA Description	This area is concerned with the history, theory, design, implementation, and use of programming languages. This area includes elements such as programming paradigms and related languages; grammars, automata, and semantics; language translation systems; subprograms and modules; and issues related to data types, program control, assignment and expressions, and run-time performance.

The Algorithms and Data Structures Knowledge Area

KA Number	1.1
KA Name	**Algorithms and Data Structures**
KA Description	This area is concerned with basic data structures, abstract data types, recursive algorithms, algorithm analysis, sorting and searching, complexity and computability issues, problem-solving strategies, and parallel and distributed algorithms.
Knowledge Units	1.1.1 Basic Data Structures 1.1.2 Design of Algorithms 1.1.3 Analysis of Algorithms
References	[Horowitz 83], [Sedgewick 88], [Tucker 91], [Weiss 94]

Description of the Algorithms and Data Structures Knowledge Units

KU Number	1.1.1
KU Name	**Basic Data Structures**
KU Description	This unit is concerned with abstract data types, information hiding, modularity, and implementations using various data structures. Knowledge about basic data structures includes the definition, implementation, and applications of lists, arrays, tables, stacks, queues, trees, and graphs. This unit also includes issues related to static and dynamic implementations and the tradeoffs between different implementation strategies.
KU Number	1.1.2
KU Name	**Design of Algorithms**
KU Description	This unit is concerned with understanding and using problem-solving strategies such as greedy algorithms, divide-and-conquer algorithms, and backtracking algorithms. This unit also includes knowledge about the design and use of recursive algorithms, about algorithms related to solving problems using the basic data structures, about sorting and searching algorithms, and about parallel and distributed algorithms.
KU Number	1.1.3
KU Name	**Analysis of Algorithms**
KU Description	This unit is concerned with time and space complexity and its use in analyzing algorithms. Knowledge in this unit includes using algorithm and data structure analysis to make time-space tradeoffs in selecting algorithm/data structures and using experimental methods to corroborate theoretical complexity analysis. This unit also includes knowledge about complexity classes (including P and NP) and about models of computable functions, undecidable problems, and recursive functions.

The Computer Architecture Knowledge Area

KA Number	1.2
KA Name	**Computer Architecture**
KA Description	This area is concerned with methods of organizing efficient, reliable computing systems. It includes digital logic and digital systems, machine-level representation of data, assembly-level machine organization, processor architecture, memory system organization, interfacing, communications and networks, and alternative architectures.
Knowledge Units	1.2.1 Digital Systems 1.2.2 Computer System Organization 1.2.3 Alternative Architectures 1.2.4 Communications and Networks
References	[Floyd 97], [Forouzan 98], [Hayes 88], [Heuring 97], [Mano 93], [Tannebaum 90], [Tucker 91]

Description of the Computer Architecture Knowledge Units

KU Number	1.2.1
KU Name	**Digital Systems**
KU Description	This unit is concerned with the theory, analysis, and design of combinational and sequential digital circuits. This unit also includes knowledge about the following topics: Boolean algebra, combinational logic circuits, digital multiplexers, circuit minimization techniques, flip-flop storage elements, shift registers, counting devices, sequential logic circuits, and integrated circuits.
KU Number	1.2.2
KU Name	**Computer System Organization**
KU Description	This unit is concerned with the Von Neumann computer architecture that includes processors, memory, input/output (I/O), and transfer of information. This unit also includes knowledge on the following topics: machine language organization, assembly language organization, processor design, microprogramming, control unit, arithmetic logic unit, bus architecture, memory organization, I/O and interrupt interface, and peripheral devices.
KU Number	1.2.3
KU Name	**Alternative Architectures**
KU Description	This unit is concerned with the concepts, design, and evolution of non-Von Neumann computer architectures. This unit also includes knowledge about the following topics: pipelining, reduced instruction set computer (RISC) and complex instruction set computer (CISC) architectures, multiprocessors and multicomputers, parallel programming, data flow architecture, interconnection networks, and neural networks.
KU Number	1.2.4
KU Name	**Communications and Networks**
KU Description	This unit is concerned with concepts, techniques, and applications in telecommunications. This unit also includes knowledge about the following topics: network architectures, protocols and standards, transmission techniques and devices, speed and quality tradeoffs, and security and encoding algorithms.

The Mathematical Foundations Knowledge Area

KA Number	1.3
KA Name	**Mathematical Foundations**
KA Description	This area is concerned with the mathematical foundations of computing. This area includes mathematical logic, proof systems, discrete mathematical structures, formal languages, combinatorics, and probability and statistics.
Knowledge Units	1.3.1 Mathematical Logic and Proof Systems 1.3.2 Discrete Mathematical Structures 1.3.3 Formal Systems 1.3.4 Combinatorics 1.3.5 Probability and Statistics
References	[Aho 92], [Grimaldi 94], [Guttman 71], [Tucker 91], [Woodcock 88]

Description of the Mathematical Foundations Knowledge Units

KU Number	1.3.1
KU Name	**Mathematical Logic and Proof Systems**
KU Description	This unit is concerned with propositional logic, predicate logic, temporal logic, formal and rigorous proof techniques, mathematical induction, and automatic theorem proving.
KU Number	1.3.2
KU Name	**Discrete Mathematical Structures**
KU Description	This unit is concerned with the mathematical structure, operations, properties, and applications of sets, relations, trees, and graphs. This unit also includes knowledge about algebraic systems and their applications: Boolean algebra and switching functions, rings and modular arithmetic, and groups and coding theory.
KU Number	1.3.3
KU Name	**Formal Systems**
KU Description	This unit is concerned with formal models of computation such as state machines and automata, regular expressions, Turing machines and recursive functions, and the corresponding elements of formal languages. This unit includes knowledge about the mathematical basis of computability, the formal theory used in compiler design, and model-based and algebraic-based formal specification languages.
KU Number	1.3.4
KU Name	**Combinatorics**
KU Description	This unit is concerned with the rules of counting, permutations, combinations, the principle of inclusion and exclusion, generating functions, recurrence relations, and combinatorial algorithms.
KU Number	1.3.5
KU Name	**Probability and Statistics**
KU Description	This unit is concerned with the elements of probability, descriptive statistics, discrete distributions, probability distributions, estimation of population parameters, hypothesis testing, confidence intervals, regression analysis, analysis of variance, experimental design, and statistical control.

The Operating Systems Knowledge Area

KA Number	1.4
KA Name	**Operating Systems**
KA Description	This area is concerned with the history, evolution, philosophy, design, implementation and use of computer operating systems. This area includes elements such as tasking and processes, process coordination and synchronization, scheduling and dispatch, physical and virtual memory organization, device management, file systems and naming, security and protection, and distributed and real-time systems.
Knowledge Units	1.4.1 Operating Systems Fundamentals 1.4.2 Process Management 1.4.3 Memory Management 1.4.4 Security and Protection 1.4.5 Distributed and Real-time Systems
References	[Silberschatz 94], [Tanenbaum 87], [Tucker 91]

Description of the Operating Systems Knowledge Units

KU Number	1.4.1
KU Name	**Operating System Fundamentals**
KU Description	This unit is concerned with general operating system objectives, components, features, functions, and structuring. This unit includes information about the history, evolution, and philosophy of operating systems.
KU Number	1.4.2
KU Name	**Process Management**
KU Description	This unit is concerned with the process concept, and how processes are represented and controlled. This unit includes knowledge about models of process creation and activation, coordination and synchronization of concurrent processes, and scheduling strategies and algorithms.
KU Number	1.4.3
KU Name	**Memory Management**
KU Description	This unit is concerned with memory management strategies and algorithms, the implementation and use of virtual memory, the organization and management of file systems, and input/output management and disk scheduling.
KU Number	1.4.4
KU Name	**Security and Protection**
KU Description	This unit is concerned with techniques for dealing with threats to the security of computer system information and resources from unauthorized access, malicious destruction, or accidental events. This unit also includes knowledge about mechanisms for protecting the access of programs, processes, and users to the resources of a computer system.
KU Number	1.4.5
KU Name	**Distributed and Real-Time Systems**
KU Description	This unit is concerned with operating system issues for distributed and real-time computing systems. These includes knowledge about representation of time, process synchronization and communication in distributed and real-time systems, distributed file systems, remote services, control of shared resources, and client/server systems.

The Programming Languages Knowledge Area

KA Number	1.5
KA Name	**Programming Languages**
KA Description	This area is concerned with the history, theory, design, implementation, and use of programming languages. This area includes elements such as programming paradigms and related languages; grammars, automata, and semantics; language translation systems; subprograms and modules; and issues concerning data types, program control, assignment and expressions, and run-time processes.
Knowledge Units	1.5.1 Theory of Programming Languages 1.5.2 Programming Paradigms 1.5.3 Programming Language Design and Implementation
References	[Appleby 91], [Sebesta 89], [Trembley 85], [Tucker 91], [Wilson 93]

Description of the Programming Languages Knowledge Units

KU Number	1.5.1
KU Name	**Theory of Programming Languages**
KU Description	This unit is concerned with the formal and theoretical elements of programming languages, including fundamentals of formal languages, finite state automata and regular expressions, context-free grammars and push-down automata, and programming language semantics (informal, axiomatic, denotational, and operational).
KU Number	1.5.2
KU Name	**Programming Paradigms**
KU Description	This unit is concerned with programming paradigms and languages for these paradigms, including the history of programming languages, the procedural language paradigm, the object-oriented language paradigm, the functional language paradigm and the logic language paradigm. This unit includes knowledge about writing programs in the various paradigms, and comparing and contrasting the advantages and disadvantages of the various paradigms.
KU Number	1.5.3
KU Name	**Programming Language Design and Implementation**
KU Description	This unit is concerned with programming language design and implementation issues, including the use of virtual machines in language understanding; representation of data types; sequence control; data control, sharing, and type checking; run-time management; and language translation systems. This unit includes design and implementation of compiler and run-time systems for high-level languages, and the interaction between language design, compiler design, and run-time organization.

Software Product Engineeering

Category Number	2
Category Name	**Software Product Engineering**
Category Description	This category is concerned with a well-defined and integrated set of activities to produce correct, consistent software products effectively and efficiently. Software Product Engineering includes the technical activities of producing a software product, such as requirements engineering, design, coding, and test. These engineering activities involve documenting software work products and maintaining traceability and consistency between them. This category includes knowledge about the controlled transition between the stages of the software life cycle and the activities needed to deliver high-quality software products to the customer.
Knowledge Areas	2.1 Software Requirements Engineering 2.2 Software Design 2.3 Software Coding 2.4 Software Testing 2.5 Software Operation and Maintenance
References	[Beizer 90], [Budgen 94] , [Davis 93], [IEEE 94], [Pigoski 97]

Description of the Software Product Engineering Knowledge Areas

KA Number	2.1
KA Name	**Software Requirements Engineering**
KA Description	This area is concerned with establishing a common understanding of the requirements to be addressed by the software product. It consists of a set of transformations that attempt to understand the exact needs of a software-intensive system and convert the statement of the needs into a complete and unambiguous description of the requirements, documented according to a specified standard. This area includes knowledge of the requirements activities of elicitation, analysis, and specification.
KA Number	2.2
KA Name	**Software Design**
KA Description	This area is concerned with the transformation of the statement of requirements into a description of how these requirements are to be implemented. Software design consists of activities such as architectural design, abstract specification and interface design, component design, data structure design, and algorithm design. Software design uses a variety of techniques and forms of representation, each providing a capability for capturing and expressing a different view of the system.
KA Number	2.3
KA Name	**Software Coding**
KA Description	This area is concerned with knowledge about the construction of the software components that are identified and described in the design documents. This area includes knowledge about translation of a design into an implementation language, program coding styles, and the development and use of program documentation.
KA Number	2.4
KA Name	**Software Testing**
KA Description	This area is concerned with establishing that a correct solution to a problem, embodied in the statement of the requirements, has been developed. Testing is a multi-stage process that consists of activities for validating the software product, from the most primitive elements up to the fully integrated system. This area includes activities such as unit testing, performance testing, integration testing, system testing, and acceptance testing.

KA Number	2.5
KA Name	**Software Operation and Maintenance**
KA Description	This area includes concepts, methods, processes and techniques that support the ability of a software system to change, evolve, and survive. It begins with initial development and configuration of the system; proceeds through its installation, day-to-day operation, and maintenance; and may eventually include re-implementing the system to increase its maintainability or because of major changes in system requirements. Knowledge in this category supports an understanding of how software systems evolve, the study and analysis of maintenance costs, the development and use of processes that are needed for effective and efficient maintenance, and strategies for dealing with legacy systems.

The Software Requirements Engineering Knowledge Area

KA Number	2.1
KA Name	**Software Requirements Engineering**
KA Description	This area is concerned with establishing a common understanding of the requirements to be addressed by the software product. It consists of a set of transformations that attempt to understand the exact needs of a software-intensive system and convert the statement of needs into a complete and unambiguous description of the requirements, documented according to a specified standard. This area includes information about the requirements activities of elicitation, analysis, and specification.
Knowledge Units	2.1.1 Requirements Elicitation 2.1.2 Requirements Analysis 2.1.3 Requirements Specification
References	[Davis 93], [Faulk 96], [IEEE 94], [Jackson 95], [Loucopoulos 95], [Pfleeger 98]

Description of the Software Requirements Engineering Knowledge Units

KU Number	2.1.1
KU Name	**Requirements Elicitation**
KU Description	This unit provides knowledge that supports the systematic development of a complete understanding of the problem domain. This unit also includes knowledge about methods and techniques for uncovering, discovering, and communicating functional and non-functional requirements and constraints; it provides a foundation for decomposing a problem into intellectually manageable pieces by using objects, functions, and states.
KU Number	2.1.2
KU Name	**Requirements Analysis**
KU Description	This unit provides knowledge about the modeling of software requirements in the information, functional, and behavioral domains of a problem. This unit includes a tradeoff analysis of performance requirements and the constraints on a system, along with all of the perceived primary and derived requirements of a system, which highlight the effect on development cost and schedule. The unit includes knowledge about various requirements modeling methods (e.g., structured analysis, object-oriented analysis), the use of prototyping to examine and assess requirements, and domain analysis techniques.
KU Number	2.1.3
KU Name	**Requirements Specification**
KU Description	This unit is concerned with the representation of software requirements that result from requirements elicitation and requirements analysis. This unit includes knowledge about the principles of defining system services and constraints, about the use of specification standards, and about the application of specification methods that involve structured natural language, graphical and symbolic notation, design description languages, and formal specification languages.

The Software Design Knowledge Area

KA Number	2.2
KA Name	**Software Design**
KA Description	This area is concerned with the transformation of the statement of requirements into a description of how these requirements are to be implemented. Software design consists of activities such as architectural design, abstract specification, interface design, component design, data structure design, tasking design, and algorithm design. Software design uses a variety of techniques and forms of representation, each providing a capability for capturing and expressing a different view of the system.
Knowledge Units	2.2.1 Architectural Design 2.2.2 Abstract Specification 2.2.3 Interface Design 2.2.4 Data Structure Design 2.2.5 Algorithm Design
References	[Budgen 94], [DeMarco 79], [Dixon 96], [GE 86], [Garland 96], [Gomaa 93], [Hatley 87], [IEEE 94], [Page-Jones 80], [Pfleeger 98], [Ward 85], [Yourdon 89]

Description of the Software Design Knowledge Units

KU Number	2.2.1
KU Name	**Architectural Design**
KU Description	This unit is concerned with knowledge about developing a modular structure and representing the control relationships between modules. The unit includes knowledge about identifying and documenting the subsystems making up the overall system, and the relationships between and among the subsystems. It also includes knowledge about design methods and techniques for functional design, object-oriented design, real-time system design, and client-server system design.
KU Number	2.2.2
KU Name	**Abstract Specification**
KU Description	This unit is concerned with knowledge about producing an abstract specification of the services provided by each software module and the constraints under which it must operate. The unit includes specification notation and techniques for object-oriented designs, structured designs, real-time system design, and client-server system design. This unit also includes knowledge about partitioning the services provided by a subsystem across the components in that subsystem and about how to optimize component independence. This includes knowledge about module coupling and cohesion design concepts, and how to recognize and measure the degree of component independence in a design.
KU Number	2.2.3
KU Name	**Interface Design**
KU Description	This unit is concerned with knowledge about the design and documentation of the interface between software subsystems and between the software system and the user. This unit includes knowledge about interface design principles, task analysis and interface modeling, implementation tools, information presentation, design evaluation, and user documentation.
KU Number	2.2.4
KU Name	**Data Structure Design**
KU Description	This unit is concerned with knowledge about the specification and design of the data structures to be used in the software implementation. This unit includes knowledge about how to translate the data objects defined in the analysis and design models into data structures that reside within the software system. This unit also includes knowledge about data dictionaries, data-flow diagram, and entity relationship diagrams.

KU Number	2.2.5
KU Name	**Algorithm Design**
KU Description	This unit is concerned with knowledge about the detailed specification and design of the algorithms used to implement software functionality and services. This unit includes knowledge about the representation of procedural detail in an appropriate notation or language such as flow charts, box diagrams, decision tables, and program design languages.

The Software Coding Knowledge Area

KA Number	2.3
KA Name	**Software Coding**
KA Description	This area is concerned with knowledge about the construction of the software components that are identified and described in the design documents. This area includes knowledge about translation of a design into an implementation language, program coding styles, and the development and use of program documentation.
Knowledge Units	2.3.1 Code Implementation 2.3.2 Code Reuse 2.3.3 Coding Standards and Documentation
References	[Booch 87], [Deimel 90], [Dijkstra 76], [Humphrey 95], [Pfleeger 98], [Wilde 90]

Description of the Software Coding Knowledge Units

KU Number	2.3.1
KU Name	**Code Implementation**
KU Description	This unit is concerned with knowledge about how to translate a software design into an implementation programming language. This unit includes knowledge about modular and incremental programming, structured programming, and knowledge of various programming paradigms (assembly, procedural, object-oriented, functional, and logic). It also includes knowledge about how to use source code development tools and programming language translation tools.
KU Number	2.3.2
KU Name	**Code Reuse**
KU Description	This unit is concerned with knowledge about developing code by reuse of existing components and about developing reusable code. This unit also includes knowledge about reusable libraries, the inheritance mechanism, module referencing, and software portability issues and techniques.
KU Number	2.3.3
KU Name	**Coding Standards and Documentation**
KU Description	This unit is concerned with knowledge about the use of standards for style and documentation in the construction of software. This unit includes knowledge about how to develop internal and external program documentation.

The Software Testing Knowledge Area

KA Number	2.4
KA Name	**Software Testing**
KA Description	This area is concerned with verifying that a correct solution to the problem, embodied in the statement of the requirements, has been developed. Testing is a multi-stage process that consists of activities for validating the software product, from the most primitive elements up to the fully integrated system. This area includes activities such as unit testing, integration testing, system testing, performance testing, and acceptance testing.
Knowledge Units	2.4.1 Unit Testing 2.4.2 Integration Testing 2.4.3 System Testing 2.4.4 Performance Testing 2.4.5 Acceptance Testing 2.4.6 Installation Testing 2.4.7 Test documentation
References	[Beizer 84], [Beizer 90], [IEEE 94], [Pfleeger 98]

Description of the Software Testing Knowledge Units

KU Number	2.4.1
KU Name	**Unit Testing**
KU Description	This unit is concerned with knowledge about testing a program unit, typically developed by a single individual, to determine that it is free of data, logic, or standards errors. This unit includes knowledge of dynamic analysis (equivalent partitioning, boundary value analysis, cause-effect graphing, logic-based testing, random testing, and syntax testing) and static analysis (complete path testing, decision testing, condition testing, and data-flow testing).
KU Number	2.4.2
KU Name	**Integration Testing**
KU Description	This unit is concerned with knowledge about validating that software components, which have been unit tested separately, interact correctly when they are put together to perform a higher order function. This unit also includes knowledge about dependency checking for calls, data, and processes, and about interface checking in terms of range, type compatibility, representation, number and order of parameters, and method of transfer.
KU Number	2.4.3
KU Name	**System Testing**
KU Description	This unit is concerned with knowledge about validating the specified functional requirements of a system. This unit includes knowledge about techniques to design and enact an independent testing process of all of the system's functions described in the software requirements specification.
KU Number	2.4.4
KU Name	**Performance Testing**
KU Description	This unit is concerned with knowledge about validating the performance requirements of a system. This unit includes knowledge about techniques to instrument performance measures like logging, event counts, event duration, and sampling. It also includes knowledge about methods for tuning a system for optimum saturation, load, and throughput threshold.
KU Number	2.4.5
KU Name	**Acceptance Testing**
KU Description	This unit is concerned with knowledge about validating the functional and non- functional requirements of a purchased or acquired system. This unit includes knowledge about techniques for using the contract, the statement of work, the software requirements specification, and the request for proposal to ensure that the delivered system meets all of the requirements (as perceived by the purchasing or acquiring organization).

KU Number	2.4.6
KU Name	**Installation Testing**
KU Description	This unit is concerned with knowledge about validating that a system will operate under all configuration possibilities. This unit includes knowledge techniques to perform configuration command checking in terms of rotation, and permutation of all physical, logical, and functional entities of a system.
KU Number	2.4.7
KU Name	**Test Documentation**
KU Description	This unit is concerned with knowledge about test plan preparation, test design specification, test case specification, test procedures specification, test item transmittal reports, test log specification, test incident reports, and test summary reports.

The Software Operation and Maintenance Knowledge Area

KA Number	2.5
KA Name	**Software Operation and Maintenance**
KA Description	This area includes concepts, methods, processes, and techniques that support the ability of a software system to change, evolve, and survive. It begins with initial development and configuration of the system; proceeds through its installation, day-to-day operation, and maintenance; and may eventually include re-implementing the system to increase its maintainability to address major changes in system requirements. Knowledge in this category supports understanding of how software systems evolve, the study and analysis of maintenance costs, the development and use of processes that are needed for effective and efficient maintenance, and strategies for dealing with legacy systems.
Knowledge Units	2.5.1 Software Installation and Operation 2.5.2 Software Maintenance Operations 2.5.3 Software Maintenance Process 2.5.4 Software Maintenance Management 2.5.5 Software Reengineering
References	[Arnold 93], [Arthur 88], [IEEE 94], [Lano 94], [Lehman 85], [Pigoski 97]

Description of the Software Operation and Maintenance Knowledge Units

KU Number	2.5.1
KU Name	**Software Installation and Operation**
KU Description	This unit is concerned with the methods and techniques for installing a software product and the continuing effective operation of that product. This unit includes provision for a smooth, orderly transition of a system from the developer organization to the user organization, and the documentation and training necessary for proper system operation.
KU Number	2.5.2
KU Name	**Software Maintenance Operations**
KU Description	This unit is concerned with the following types of maintenance: corrective maintenance, adaptive maintenance, perfective maintenance, and preventive maintenance. Corrective maintenance entails the identification and removal of faults in the software. Adaptive maintenance is concerned with changing software so that it can operate in some new environment, such as on a different hardware platform or for use with a different operating system. Perfective maintenance involves implementing new functional or non-functional system requirements generated by software customers as their organization or business changes. Preventive maintenance concerns changing software to make it more maintainable.
KU Number	2.5.3
KU Name	**Software Maintenance Process**
KU Description	This unit is concerned with the process used in performing software maintenance. Such a process would include phases similar to those in a process for developing a new software product. A maintenance process starts with a change request and a preliminary problem analysis. Next, a managerial and technical analysis is undertaken to investigate and determine the cost of alternative solutions. Then, the chosen solution is implemented and tested. Finally, the change is released to the customer.
KU Number	2.5.4
KU Name	**Software Maintenance Management**
KU Description	This unit is concerned with the organizational, economic, and management issues involved in the maintenance of software within an organization. This unit includes knowledge about different organizational maintenance models, maintenance cost analysis and estimation, techniques for management and execution of maintenance operations, and the examination and understanding of program evolution dynamics.

KU Number	2.5.5
KU Name	**Software Reengineering**
KU Description	This unit is concerned with the restructuring or reconstruction of a software system to improve its quality, understandability, and maintainability. Software reengineering efforts are often focused on legacy systems. Software reengineering includes activities such as inventory analysis, document restructuring, reverse engineering and code restructuring, data restructuring, and forward reengineering.

Software Management

Category Number	3
Category Name	**Software Management**
Category Description	This category deals with the concepts, methods, and techniques for managing software products and projects. Software management includes activities concerned with project management, risk management, software quality, and configuration management.
Knowledge Areas	3.1 Software Project Management 3.2 Software Risk Management 3.3 Software Quality Management 3.4 Software Configuration Management 3.5 Software Process Management 3.6 Software Acquisition
References	[Boehm 81], [Boehm 91], [Humphrey 89], [Thayer 88]

Description of the Software Management Knowledge Areas

KA Number	3.1
KA Name	**Software Process Management**
KA Description	This area is concerned with defining project objectives, assessing project needs and resources, developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan for performing the work.
KA Number	3.2
KA Name	**Software Risk Management**
KA Description	This area is concerned with the concepts, methods, and techniques for managing risks that threaten a plan for developing a software product. Risk management includes such activities as risk identification, risk analysis, monitoring risks, risk mitigation, and risk planning.
KA Number	3.3
KA Name	**Software Quality Management**
KA Description	This area is concerned with the concepts, methods, techniques, procedures, and standards for producing high-quality software products. This area includes knowledge about quality planning and control, verification and validation activities, measurement of product and process attributes, and software dependability and reliability.
KA Number	3.4
KA Name	**Software Configuration Management**
KA Description	This area deals with the discipline of identifying the configuration of a system at discrete points in time for systematically controlling changes to this configuration and maintaining the integrity and traceability of this configuration throughout the life of the software system.
KA Number	3.5
KA Name	**Software Process Management**
KA Description	This area is concerned with the management of the technical aspects of the software development process. This area includes knowledge about the following software process elements: activities, methods, practice, and transformations that people use to develop and maintain software and associated products. This also includes ensuring that the processes within an organization are performing as expected; that is, defined processes are being followed and improvements to the processes are being made so as to meet organizational objectives. This area also includes knowledge about establishing processes that are used and can act as a foundation for systematic improvement based on the organization's needs.

KA Number	3.6
KA Name	**Software Acquisition**
KA Description	This area is concerned with knowledge about acquiring a custom software system by a contracting agency from software developers independent of the agency. This area includes knowledge about acquisition activities such as procurement, contracting, performance evaluation, and providing for future support of the software system.

The Software Project Management Knowledge Area

KA Number	3.1
KA Name	**Software Project Management**
KA Description	This area is concerned with defining project objectives, assessing project needs and resources, developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan for performing the work.
Knowledge Units	3.1.1 Project Planning 3.1.2 Project Organization 3.1.3 Project Forecasting 3.1.4 Project Scheduling 3.1.5 Project Control
References	[Paulk 93], [Thayer 88]

Description of the Software Project Management Knowledge Units

KU Number	3.1.1
KU Name	**Project Planning**
KU Description	This unit includes knowledge about preparing a plan for a software project. This unit includes knowledge about how to determine the project scope, select the objectives and goals of the project, and decide on the strategies, policies, programs, and procedures for achieving the project objectives.
KU Number	3.1.2
KU Name	**Project Organization**
KU Description	This unit includes knowledge about how to determine and itemize the activities required to achieve the objectives of a software development project. This unit includes arranging these activities into logical clusters and assigning these logical clusters to a project team, as well as delegating responsibility and authority to the team members.
KU Number	3.1.3
KU Name	**Project Forecasting**
KU Description	This unit includes knowledge about how to anticipate future events (such as availability of personnel, predicted inflation rate, and availability of new computer hardware) and how to judge the effect that these events will have on a software engineering project. This unit includes knowledge about how to make an informed prediction of the effort, cost, time, and quality that will be needed for developing, changing, and maintaining a software system.
KU Number	3.1.4
KU Name	**Project Scheduling**
KU Description	This unit includes knowledge about how to develop a schedule for the completion of a software project. This unit includes determining project tasks, allocating resources for completing the tasks, determining task ordering and dependencies, scheduling start and completion times for each project task, and establishing project milestone dates.
KU Number	3.1.5
KU Name	**Project Control**
KU Description	This unit includes knowledge about how to ensure that actual operations go according to plan. This unit includes knowledge about measuring performance against goals and plans, determining when deviations exist, and initiating actions to correct deviations.

The Software Risk Management Knowledge Area

KA Number	3.2
KA Name	**Software Risk Management**
KA Description	This area is concerned with the concepts, methods, and techniques for managing risks that threaten a plan for developing a software product. Risk management includes such activities as risk identification, risk analysis, monitoring risks, risk mitigation, and risk planning.
Knowledge Units	3.2.1 Risk Analysis 3.2.2 Risk Management Planning 3.2.3 Risk Monitoring
References	[Boehm 91], [Hall 98], [Karolak 96]

Description of the Software Risk Management Knowledge Units

KU Number	3.2.1
KU Name	**Risk Analysis**
KU Description	This unit includes knowledge about how to identify sources of risk, how to classify risks as to their causes and symptoms, how to assess the loss probability and loss magnitude for each risk, and how to prioritize risk items using their probability of occurrence and the severity of their impact. A software engineer who is knowledgeable in risk identification will be able to produce lists of the product-specific risk items likely to compromise a project's success.
KU Number	3.2.2
KU Name	**Risk Management Planning**
KU Description	This unit includes knowledge about how to develop a risk management plan. Such a plan lays out the activities necessary to bring each risk item under control. This unit requires knowledge about risk resolution strategies and how to make plans to mitigate or avoid the consequences when a risk item occurs.
KU Number	3.2.3
KU Name	**Risk Monitoring**
KU Description	This unit includes knowledge about how to track the projects' progress toward the resolution of risk items, and how and when to take appropriate corrective action.

The Software Quality Management Knowledge Area

KA Number	3.3
KA Name	**Software Quality Management**
KA Description	This area is concerned with the concepts, methods, techniques, procedures, and standards for producing high-quality software products in an efficient and cost-effective manner. This area includes knowledge about quality planning and control, verification and validation activities, measurement of product and process attributes, and software dependability and reliability.
Knowledge Units	3.3.1 Software Quality Assurance 3.3.2 Verification and Validation 3.3.3 Software Metrics 3.3.4 Dependable Systems
References	[Brooks 95], [Fagan 76], [Gilb 93], [Gillies 92], [Grady 92], [IEEE 94], [Ince 94], [Kan 95], [Levenson 97], [Musa 87], [Paulk 93], [Sheppard 92], [Woodcock 88]

Description of the Software Quality Management Knowledge Units

KU Number	3.3.1
KU Name	**Software Quality Assurance**
KU Description	This unit is concerned with software management functions and activities intended to ensure that a software product conforms to its explicitly stated functional and performance requirements. It includes knowledge about organization of quality assurance units, quality planning, oversight, record keeping, analysis, auditing, and reporting. This unit also includes knowledge about quality assurance techniques such as Pareto analysis, trend analysis, statistical quality control, and regression testing.
KU Number	3.3.2
KU Name	**Verification and Validation**
KU Description	This unit is concerned with knowledge about verification and validation (V&V) concepts, methods, activities, and deliverables associated with each phase in the software life cycle. This unit includes knowledge about V&V planning and organization; personal reviews, walkthroughs, and inspections; traceability analysis; formal verification techniques; cleanroom techniques; and software testing.
KU Number	3.3.3
KU Name	**Software Metrics**
KU Description	This unit is concerned with the formulation of software measures and metrics, the collection of data required by the formulated metrics, the computation of metrics, and the analysis, interpretation, and feedback of metrics results. The unit involves knowledge about product metrics (for requirements, design, and code), resource metrics (for human and technical resources), and process metrics (for effort and schedule measures).
KU Number	3.3.4
KU Name	**Dependable Systems**
KU Description	This unit is concerned with knowledge about the development of software systems that must be dependable (i.e., systems that have critical non-functional requirements for reliability, safety, and security). This unit includes knowledge about the specification of reliable software, reliability metrics, statistical testing, fault tolerance and avoidance, defensive programming and exception handling, and safety specification and assurance.

The Software Configuration Management Knowledge Area

KA Number	3.4
KA Name	**Software Configuration Management**
KA Description	This area deals with the discipline of identifying the configuration of a system at discrete points in time for the purposes of systematically controlling changes to this configuration and maintaining the integrity and traceability of this configuration throughout the life of the software system.
Knowledge Units	3.4.1 Software Configuration Identification 3.4.2 Software Configuration Control 3.4.3 Software Configuration Audit 3.4.4 Software Configuration Status Accounting
References	[Bersoff 80], [Buckley 94], [Paulk 93]

Description of the Software Configuration Management Knowledge Units

KU Number	3.4.1
KU Name	**Software Configuration Identification**
KU Description	This unit is concerned with knowledge about defining a system's baseline components and identifying updates to a particular baseline. It also includes knowledge about the incremental establishment and maintenance of a basis for control and status accounting for configuration items throughout the life cycle of a software system.
KU Number	3.4.2
KU Name	**Software Configuration Control**
KU Description	This unit is concerned with knowledge about initiation, evaluation, coordination, approval or disapproval, and implementation of changes to configuration items throughout the life cycle of a software system.
KU Number	3.4.3
KU Name	**Software Configuration Audit**
KU Description	This unit is concerned with knowledge about verifying that all required configuration items have been produced, that the current version agrees with the requirements specification, that the technical documentation completely and accurately describes the configuration items, and that all change requests have been resolved.
KU Number	3.4.4
KU Name	**Software Configuration Status Accounting**
KU Description	This unit is concerned with knowledge about recording and reporting the information that is needed to change a configuration effectively, including a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of all approved changes.

The Software Process Management Knowledge Area

KA Number	3.5
KA Name	**Software Process Management**
KA Description	This area is concerned with the management of the technical aspects of the software development process. This includes knowledge about software process elements: activities, methods, practice, and transformations that people use to develop and maintain software and associated products. This also includes ensuring that the processes within an organization are performing as expected; that is, defined processes are being followed and improvements to the processes are being made so as to meet organizational objectives. This area also includes knowledge about establishing processes that are used and can act as a foundation for systematic improvement based on the organization's needs.
Knowledge Units	3.5.1 Quantitative Software Process Management 3.5.2 Software Process Improvement 3.5.3 Software Process Assessment 3.5.4 Software Process Automation 3.5.5 Software Process Engineering
References	[Christie 94], [Hanrahan 95], [Humphrey 89], [Humphrey 92], [Paulk 93], [Paulk 95], [SPC 92]

Description of the Software Process Management Knowledge Units

KU Number	3.5.1
KU Name	**Quantitative Software Process Management**
KU Description	This unit is concerned with quantitative control of process performance for a software project. This includes knowledge about establishing software process performance goals, measuring the process performance, analyzing process measurements, and making adjustments to maintain process performance within acceptable limits.
KU Number	3.5.2
KU Name	**Software Process Improvement**
KU Description	This unit is concerned with knowledge about the use of artifacts, lessons, data, and general experience from software projects to improve software products and processes. This unit includes knowledge about how to evolve a process from lower levels to higher levels of process maturity and how to plan, develop, and implement changes to the software process.
KU Number	3.5.3
KU Name	**Software Process Assessment**
KU Description	This unit is concerned with knowledge about acquiring an understanding of a software development organization's state of software practice, identifying the key areas for improvement, and initiating actions to make these improvements. This unit includes knowledge about how to use an assessment as a diagnostic tool to aid organizational development.
KU Number	3.5.4
KU Name	**Software Process Automation**
KU Description	This unit is concerned with knowledge about how to integrate people in a software development organization with the development process and the tools supporting that development. This unit includes knowledge about how to provide or use computer-based real-time support and guidance for the enactment of software development processes.
KU Number	3.5.5
KU Name	**Software Process Engineering**
KU Description	This unit deals with methodologies, tools, and techniques for the design and implementation of software processes. This unit includes knowledge about representing the important characteristics of a process as a coherent, integrated set of well-defined software engineering and management processes for organizations, teams, and individuals.

The Software Acquisition Knowledge Area

KA Number	3.6
KA Name	**Software Acquisition**
KA Description	This area is concerned with knowledge about acquiring a custom software system by a contracting agency from software developers independent of the agency. This area includes knowledge about acquisition activities such as procurement, contracting, performance evaluation, and providing for future support of the software system.
Knowledge Units	3.6.1 Procurement Process 3.6.2 Acquisition Planning 3.6.3 Performance Management
References	[Humphrey 89], [Marciniak 90]

Description of the Software Acquisition Knowledge Units

KU Number	3.6.1
KU Name	**Procurement Management**
KU Description	This unit is concerned with knowledge about the process for competitive procurement of software systems. This unit includes knowledge about preparation and distribution of a solicitation package, proposal evaluation and source selection, and contract negotiations and finalization.
KU Number	3.6.2
KU Name	**Acquisition Planning**
KU Description	This unit is concerned with knowledge about developing a life-cycle plan for acquisition and use of a software system. This unit includes knowledge about project organization and communication, project budget and schedule, acquisition and development standards, subcontractor management, and software development planning.
KU Number	3.6.3
KU Name	**Performance Management**
KU Description	This unit is concerned with knowledge about assessing the developer's performance in developing the system being acquired. This unit includes knowledge about management reviews, quality assurance, test and evaluation, and metrics and performance indicators.

Software Domains

Category Number	4
Category Name	**Software Domains**
Category Description	This category is concerned with knowledge about specific domains that involve computing and software engineering application or utilization. This category includes the following software domains: artificial intelligence, database systems, human-computer interaction, numerical and symbolic computing, computer simulation, and software acquisition.
Knowledge Areas	4.1 Artificial Intelligence 4.2 Database Systems 4.3 Human-Computer Interaction 4.4 Numerical and Symbolic Computing 4.5 Computer Simulation 4.6 Real-Time Systems
References	[CACM 95], [Gomaa 93], [Hill 90], [Krishna 92], [Lethbridge 98], [Levi 90], [Maron 87], [Pooch 93], [Proctor 94], [Rob 97], [Russell 95], [Trembley 85], [Tucker 91]

Description of the Software Domains Knowledge Areas

KA Number	4.1
KA Name	**Artificial Intelligence**
KA Description	This area is concerned with basic models of behavior and the building of virtual and actual machines to simulate animal and human behavior. This area also includes knowledge about the history and applications of artificial intelligence; problems, state spaces, and search strategies; logical and probabilistic reasoning systems; knowledge engineering; robotics; learning theory; neural networks; and natural language processing.
KA Number	4.2
KA Name	**Database Systems**
KA Description	This area is concerned with file systems, database systems, and database models. This area also includes knowledge about the history and evolution of file and database systems, the relational database model and the Structured Query Language, entity-relationship modeling, data normalization, transaction management and concurrency control, distributed and client/server database systems, and object-oriented database systems.
KA Number	4.3
KA Name	**Human-Computer Interaction**
KA Description	This area is concerned with user interfaces, computer graphics, and hypertext/ hypermedia. This area also includes knowledge about input/output devices; the use and construction of interfaces; graphical devices, models, and algorithms used in graphical systems, graphics packages, and graphics applications; and hypertext/hypermedia concepts, environments, applications, and design.
KA Number	4.4
KA Name	**Numerical and Symbolic Computing**
KA Description	This area is concerned with methods for efficiently and accurately using computers to solve equations for mathematical models. This area also includes knowledge about computer representation of numerical systems; classification, analysis, and control of numeric errors; iterative approximation methods; numerical algorithms used in science and engineering; and the development of mathematical software packages.

KA Number	4.5
KA Name	**Computer Simulation**
KA Description	This area is concerned with the basic aspects of modeling and simulation. It includes knowledge about statistical models, queuing theory, random variable generation, discrete system simulation, simulation languages, graphic output with animation, and validation of simulation models.
KA Number	4.6
KA Name	**Real-Time Systems**
KA Description	This area is concerned with knowledge about the development of real-time software systems. It includes knowledge about requirements, design, implementation, and basic properties of real-time application software. This area also includes knowledge about concurrent programming, process synchronization and scheduling, resource management, software reliability, real-time programming languages, and real-time operating systems.

5 Summary

This SWE-BOK is a structured model that describes the knowledge and skills that constitute the software engineering discipline. Materials from the ACM and IEEE-CS, the Project Management Institute, the FAA, and others [ACM 89, ACM 98, Davis 97, Dupuis 98, Ibrahim 97, PMI 94, Tucker 91] have provided ideas, concepts, and perspectives. The resulting three-tiered architecture, and the material that it organizes, corresponds with other similar approaches and will be easy to use and adapt to other related projects and activities.

The use of this body of knowledge in a wide variety of contexts will determine the extent to which the breadth and depth of the discipline has been captured in a useful way. To that end, and to support similar efforts aimed at maturing software engineering as a discipline, we anticipate that this report will not only elicit comments and suggestions for enhancement and extension, but will also encourage the broader use of this SWE-BOK throughout the software community.

References

[ACM 89] ACM Task Force on the Core of Computer Science. “Computing as a Discipline.” *Communications of the ACM 32*, 1 (January 1989): 1-5.

[ACM 98] Association for Computing Machinery. “The Full Computing Classification System, 1998 Version.” *Computing Reviews 38,*1 (January 1998): 8-62.

[Aho 92] Aho, V. A. and Ullman, J. D. *Foundations of Computer Science.* New York, NY: Computer Science Press, 1992.

[Appleby 91] Appleby, Doris. *Programming Languages: Paradigm and Practice*. New York, NY: McGraw-Hill, 1991.

[Arnold 93] Arnold, R. S. *Software Reengineering*. Los Alamitos, CA: IEEE Computer Society Press, 1993.

[Arthur 88] Arthur, L. J. *Software Evolution*. New York, NY: John Wiley, 1988.

[Beizer 84] Beizer, B. *Software System Testing and Quality Assurance*. New York, NY: Van Norstrand Reinhold Company, 1984.

[Beizer 90] Beizer, B. *Software Testing Techniques*. New York, NY: Van Norstrand Reinhold Company, 1990.

[Bersoff 80] Bersoff, E. H.; Henderson, V. H.; and Siegel, S. G. *Software Configuration Management*. Englewood Cliffs, NJ: Prentice-Hall, 1980.

[Boehm 81] Boehm, B. W. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[Boehm 91] Boehm, B. W. “Software Risk Management: Principles and Practices.” *IEEE Software 8*, 1 (January 1991): 32-41.

[Booch 87] Booch, G. *Software Components with Ada: Structures, Tools and Subsystems*. Menlo Park, CA: Benjamin/Cummings, 1987.

[Brooks 95] Brooks, F. P. *The Mythical Man-Month, Essays on Software Engineering, Anniversary Edition*. Reading, MA: Addison-Wesley, 1995.

[Brookshear 94] Brookshear, J. Glenn. *Computer Science: An Overview, 4th edition*. Menlo Park, CA: Benjamin/Cummings, 1994.

[Buckley 94] Buckley, F. J. *Implementing Configuration Management: Hardware, Software, and Firmware*. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[Budgen 94] Budgen, D. *Software Design*. Reading, MA: Addison-Wesley, 1994.

[CACM 95] Special Issue on "Designing Hypermedia Applications." *Communications of the ACM 38*, 8 (August 1995).

[Cheney 85] Cheney, W. and Kincaid, D. *Numerical Mathematics and Computing*. Pacific Grove, CA: Brooks/Cole, 1985.

[Christie 94] Christie, A. *A Practical Guide to the Technology and Adoption of Software Process Automation*, (CMU/SEI-94-TR-007, ADA 280916). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1994.

[Davis 93] Davis, A. *Software Requirements: Objects, Functions & States*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[Davis 97] Davis, G. B. et al. *IS'97: Model Curriculum for Undergraduate Degree Programs in Information Systems*. Available WWW <URL: http://www.acm.org/education/curricula.html#IS97> (1997).

[Deimel 90] Deimel, L. and Naveda, F. *Reading Computer Programs: Instructor's Guide and Exercises* (CMU/SEI-90-EM-3, ADA 228026). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

[DeMarco 79] DeMarco, Tom. *Structured Analysis and System Specification*. Englewood Cliffs, NJ: Prentice-Hall/Yourdon Press, 1979.

[Denning 89] Denning, Peter J. et al. "Computing as a Discipline." *Communications of the ACM 32*, 1 (January 1989): 9-23.

[Dijkstra 76] Dijkstra, E. *A Discipline of Programming*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

[Dixon 96] Dixon R. *Client/Server and Open Systems*. New York, NY: John Wiley, 1996.

[Dorfman 97] Dorfman, M. and Thayer, R., eds. *Software Engineering*. Los Alamitos, CA: Computer Society Press, 1997.

[Dupuis 98] Dupuis, R. et al. *A Guide to the Software Engineering Body of Knowledge, A Straw Man Version*. Los Alamitos, CA: IEEE Computer Society Press, 1998.

[Fagan 76] Fagan, M. E. "Design and Code Inspections to Reduce Errors in Programs." *IBM Systems Journal 15,* 3 (3rd Quarter 1976): 219-248.

[Faulk 96] Faulk, S. *Software Requirements: A Tutorial, Software Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[Floyd 97] Floyd, T. *Digital Fundamentals, 6th edition*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[Ford 96] Ford, Gary and Gibbs, Norman E. *A Mature Profession of Software Engineering* (CMU/SEI-96-TR-004, ADA 307889). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996.

[Forouzan 98] Forouzan, B. *Introduction to Communications and Networking.* New York, NY: McGraw-Hill, 1998.

[Garland 96] Garland, D. and Shaw, M. *Software Architecture: Perspectives on an Emerging Discipline*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[GE 86] General Electric Company. *Software Engineering Handbook.* New York, NY: McGraw-Hill, 1986.

[Gibbs 94] Gibbs, W. "Software's Chronic Crisis." *Scientific American 271,* 3 (September 1994): 86-95.

[Gilb 93] Gilb, T. and Graham, D. *Software Inspections*. Reading, MA: Addison-Wesley, 1993.

[Gillies 92] Gillies, A. C. *Software Quality: Theory and Management*. London: Chapman & Hall, 1992.

[Gomaa 93] Gomaa, H. *Software Design Methods for Concurrent and Real-Time Systems*. Reading, MA: Addison-Wesley, 1993.

[Grady 92] Grady, R. B. and Caswell, D. L. *Practical Software Metrics for Project Management and Process Improvement.* Englewood Cliffs, NJ: Prentice-Hall, 1992.

[Grimaldi 94] Grimaldi, R. *Discrete and Combinatorial Mathematics, 3rd edition*. Reading, MA: Addison-Wesley, 1994.

[Guttman 71] Guttman, I.; Wilkes, S.; and Hunter, J. *Introductory Engineering Statistics, 2nd edition*. New York, NY: John Wiley, 1971.

[Hall 98] Hall, E. M. *Managing Risk*. Reading, MA: Addison-Wesley, 1998.

[Hanrahan 95] Hanrahan, Robert P. *The IDEF Process Modeling Methodology*. Available WWW <URL: http://www.stsc.hill.af.mil/crosstalk/ > (1995).

[Hartmanis 92] Hartmanis, Juris and Lin, Herbert, eds. *Computing the Future*. Washington, D.C.: National Academy Press, 1992.

[Hatley 87] Hatley, Derek and Pirbhai, Imtiaz. *Strategies for Real-Time System Specification.* New York, NY: Dorset House, 1987.

[Hayes 88] Hayes, J. *Computer Architecture and Organization, 2nd edition*. New York, NY: McGraw-Hill, 1988.

[Heuring 97] Heuring, V. and Jordan, H. *Computer Systems Design and Architecture*. Reading, MA: Addison-Wesley, 1997.

[Hilburn 98] Hilburn, T.; Bagert, D.; Mengel, S.; and Oexmann, D. "Software Engineering Across Computing Curricula," *Proceedings of Sixth Annual Conference on the Teaching of Computing.* Dublin, August 1998.

[Hill 90] Hill, F. *Computer Graphics*. New York, NY: Macmillan, 1990.

[Horowitz 83] Horowitz, E. and Sahni, S. *Fundamentals of Data Structures*. New York, NY: Computer Science Press, 1983.

[Humphrey 89] Humphrey, W. S. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.

[Humphrey 92] Humphrey, W. S. *Introduction to Software Process Improvement* (CMU/SEI-92-TR-7, ADA 253326). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992.

[Humphrey 95] Humphrey, W. S. *A Discipline for Software Engineering.* Reading, MA: Addison-Wesley, 1995.

[Ibrahim 97] Ibrahim, L. et al. *The Federal Aviation Administration Integrated Capability Maturity Model, Version 1.0*. Washington, D.C.: Federal Aviation Administration, November 1997.

[IEEE 94] IEEE. *Software Engineering Standards, 1994 Edition*. Los Alamitos, CA: IEEE Computer Society Press, 1994.

[Ince 94] Ince, D. *ISO 9001 and Software Quality Assurance*. New York, NY: McGraw-Hill, 1994.

[Jackson 95] Jackson, M. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Reading, MA: Addison-Wesley, 1995.

[Kan 95] Kan, S. H. *Metrics and Models in Software Quality Engineering*. Reading, MA: Addison-Wesley, 1995.

[Karolak 96] Karolak, D. W. *Software Engineering Risk Management*. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[Krishna 92] Krishna, M. K. *Real-Time Systems: Abstractions, Languages and Design Methodologies*. Los Alamitos, CA: IEEE Computer Society Press, 1992.

[Lano 94] Lano, K. and Haughton, H. *Reverse Engineering and Software Maintenance.* New York, NY: McGraw-Hill, 1994.

[Lehman 85] Lehman, M. and Belady, L. *Program Evolution: Processes of Software Change*. San Diego, CA: Academic Press, 1985.

[Lethbridge 98] Lethbridge, T.C. “A Survey of the Relevance of Computer Science and Software Engineering Education.” *Proceedings of 11th Conference on Software Engineering Education & Training*. Los Alamitos, CA: IEEE Computer Society Press, 1998.

[Leveson 97] Leveson, N.G. *Safeware: System Safety And Computers*. Reading, MA: Addison-Wesley, 1997.

[Levi 90] Levi, S. and Agrawala, A. *Real-Time System Design*. New York, NY: McGraw-Hill, 1990.

[Loucopoulos 95] Loucopouslos, P. and Karakostas, V. *Systems Requirements Engineering*. New York, NY: McGraw-Hill, 1995.

[Mano 93] Mano, M., *Computer System Architecture, 3rd edition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[Marciniak 90] Marciniak, J. J., and Reifer, D. J. *Software Acquisition Management*. New York, NY: John Wiley, 1990.

[Marciniak 94] Marciniak, J. J., ed. *Encyclopedia of Software Engineering*. New York, NY: John Wiley, 1994.

[Maron 87] Maron, M. *Numerical Analysis, A Practical Approach.* New York, NY: Macmillan, 1987.

[Musa 87] Musa, J. D.; Iannino, A.; and Okumoto, K. *Software Reliability: Measurement, Prediction, Applications*. New York, NY: McGraw-Hill, 1987.

[Page-Jones 80] Page-Jones, Meilir. *The Practical Guide to Structured Systems Design.* Englewood Cliffs, NJ: Prentice-Hall/Yourdon Press, 1980.

[Paulk 93] Paulk M. et al. *Capability Maturity Model, Version 1.1*, (CMU/SEI-93-TR-24, ADA 263403). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1993.

[Paulk 95] Paulk M. et al. *The Capability Maturity Model Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley, 1995.

[Pfleeger 98] Pfleeger, S. *Software Engineering Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[Pigoski 97] Pigoski, T. M. *Practical Software Maintenance*. New York, NY: John Wiley, 1997.

[Pooch 93] Pooch, U. and Wall, J. *Discrete Event Simulation*. Boca Raton, FL: CRC Press, 1993.

[Pressman 97] Pressman, Roger S. *Software Engineering: A Practitioner's Approach, 4th Edition.* New York, NY: McGraw-Hill, 1997.

[Proctor 94] Proctor, R. and Zandt, T. *Human Factors*. Boston, MA: Allyn and Bacon, 1994.

[PMI 94] PMI Standards Committee. *A Guide to the Project Management Body of Knowledge (PMBOK)*. Available WWW <URL: http://www.pmi.org/publictn/pmboktoc.htm > (1994).

[Rob 97] Rob, P. and Coronel, C. *Database Systems, 3rd edition*. Boston, MA: Boyd & Fraser, 1997.

[Russell 95] Russell, S. and Norvig, P. *Artificial Intelligence, A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[Sebesta 89] Sebesta, Robert W. *Concepts of Programming Languages*. Menlo Park, CA: Benjamin/Cummings, 1989.

[Sedgewick 88] Sedgewick, R. *Algorithms, 2nd edition*. Reading, MA: Addison-Wesley, 1988.

[Sheppard 92] Sheppard, M. *Software Engineering Metrics*. New York, NY: McGraw-Hill, 1992.

[Silberschatz 94] Silberschatz, A. and Galvin, P. *Operating System Concepts*. Reading, MA: Addison-Wesley, 1994.

[Sommerville 95] Sommerville, I. *Software Engineering, 5th edition*. Reading, MA: Addison-Wesley, 1995.

[SPC 92] SPC. *Process Definition and Modeling Guidebook* (SPC-92041-CMC). Boston, MA: International Thomson Computer Press, 1992.

[Tannebaum 90] Tannebaum, Andrew S. *Structured Computer Organization, 3rd edition*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[Thayer 88] Thayer, R. H. *Software Engineering Project Management: A Top-Down View. Tutorial: Software Engineering Project Management*. Los Alamitos, CA: IEEE Computer Society Press, 1988.

[Trembley 85] Trembley, J. and Sorenson, P. *The Theory and Practice of Compiler Writing*. New York, NY: McGraw-Hill, 1985.

[Tucker 91] Tucker, Allen B., ed. *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[Ward 85] Ward, Paul and Mellor, Stephen. *Structured Development for Real-Time Systems Volume 1, 2, and 3*. Englewood Cliffs, NJ: Prentice-Hall/Yourdon Press, 1985.

[Weiss 94] Weiss, M. *Data Structures and Algorithm Analysis in C++*. Reading, MA: Addison-Wesley, 1994.

[Wilde 90] Wilde, N. *Understanding Program Dependencies* (CMU/SEI-CM-26, ADA 235700). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

[Wilson 93] Wilson, Leslie B. and Clark, Robert G. *Comparative Programming Languages*. Reading, MA: Addison-Wesley, 1993.

[Woodcock 88] Woodcock, J. and Loomes, M. *Software Engineering Mathematics*. Reading, MA: Addison-Wesley, 1988.

[Yourdon 89] Yourdon, Edward. *Modern Structured Analysis*. Englewood Cliffs, NJ: Prentice-Hall/Yourdon Press, 1989.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (LEAVE BLANK)	**2.** REPORT DATE **April 1999**	**3.** REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE **A Software Engineering Body of Knowledge Version 1.0**		**5.** FUNDING NUMBERS C — F19628-95-C-0003
6. AUTHOR(S) **Thomas B. Hilburn, Iraj Hirmanpour, Soheil Khajenoori, Richard Turner, Abir Qasem**		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		**8.** PERFORMING ORGANIZATION REPORT NUMBER **CMU/SEI-99-TR-004**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/DIB 5 Eglin Street Hanscom AFB, MA 01731-2116		**10.** SPONSORING/MONITORING AGENCY REPORT NUMBER **ESC-TR-99-004**
11. SUPPLEMENTARY NOTES		
12.A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		**12.B** DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

Software engineering, both as a discipline and as a profession, is at a pivotal point in its evolution. Although software has become critical in the development of most new human-created systems, the concepts, principles, and methods for engineering software are still neither well defined nor uniformly agreed upon. The lack of consensus regarding software engineering practice and the requisite competencies creates confusion and has serious consequences for the evaluation, acquisition, and application of software engineering knowledge. This report presents an effort to organize and catalogue a body of knowledge for software engineering and to provide a systematic, concise, and complete description of the software engineering discipline. This body of knowledge can assist organizations in defining and improving the software engineering competencies of their workforces; it can help educational institutions in defining software engineering curricula; it can provide a basis for classifying academic and industrial research and development efforts; and it can improve the understanding and practice of software engineering.

14. SUBJECT TERMS **body of knowledge, Federal Aviation Administration (FAA), knowledge area,**			**15.** NUMBER OF PAGES **55**
knowledge units, software engineering, software engineering capabilities, software engineering curriculum			**16.** PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18.** SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	**19.** SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	**20.** LIMITATION OF ABSTRACT UL