NCEES
advancing licensure for engineers and surveyors

NCEES Principles and Practice of Engineering Examination
Software Engineering Exam Specifications

Effective Beginning with the April 2013 Examinations

- The exam is an 8-hour open-book exam. It contains 40 multiple-choice questions in the 4-hour morning session, and 40 multiple-choice questions in the 4-hour afternoon session. Examinee works all questions.
- The exam uses both the International System of units (SI) and the US Customary System (USCS).
- The exam is developed with questions that will require a variety of approaches and methodologies, including design, analysis, and application.
- The knowledge areas specified as examples of kinds of knowledge are not exclusive or exhaustive categories.

		Approximate Number of Questions
I.	**Requirements**	**14**
A.	Software requirements fundamentals (e.g., concept of operations, types of requirements, product and process requirements, functional and nonfunctional requirements, quantifiable requirements, system requirements, software requirements, derived requirements, constraints, service level)	
B.	Requirements elicitation (e.g., requirements sources, elicitation techniques, requirements representation)	
C.	Requirements specification (e.g., System Definition Document, System/Subsystems Specification, Software Requirements Specification, Interface Requirements Specification)	
D.	Requirements analysis (e.g., requirements classification, conceptual modeling, architectural design and requirements, requirements allocation, requirements negotiation, formal methods, feasibility analysis)	
E.	Requirements verification and validation (e.g., requirements reviews, prototyping, model validation, simulation)	
F.	Requirements management (e.g., iterative nature of the requirements process, change management, requirement attributes, requirements traceability, measuring requirements, software requirements tools)	
II.	**Design**	**11**
A.	Critical design issues (e.g., context of software design, software design process, concurrency, control and handling of events, allocation of components, error and exception handling and fault tolerance, interaction and presentation, data persistence, user interfaces)	
B.	Software design strategies and methods (e.g., data structure-oriented design, object-oriented design, function-oriented design, COTS integration, design for maintainability)	
C.	Design modeling (e.g., information, behavioral, structure)	

D. Software structure and architecture (e.g., architectural structures and viewpoints, architectural styles, design patterns, families of programs and frameworks, hardware issues in software architecture)
E. Software design representation (e.g., UML, E-R diagrams, SASD, data flow diagram)
F. Analysis (e.g., analyzing for correctness, analyzing for well-formedness, analyzing for completeness, analyzing for quality, traceability, interaction analysis, prioritization and tradeoff analysis)
G. Externally furnished software components (e.g., make-buy-reuse decisions, open source, COTS, customer-furnished software)
H. Domain-specific technology (e.g., hardware-software interface, hardware-software trade-off)

III. Construction **9**

A. Software construction fundamentals (e.g., minimizing complexity, anticipating change, constructing for verification, standards in construction, patterns)
B. Construction methods (e.g., construction planning, construction measurement, test-driven development)
C. Construction techniques (e.g., profiling and performance analysis, error and exception handling, reuse, timing and synchronization analysis, tool selection and use, coding standards)

IV. Testing **10**

A. Test levels (e.g., unit testing, integration testing, system testing, acceptance testing, usability testing)
B. Test process (e.g., planning, metrics collection and analysis, reporting)
C. Software testing strategies (e.g., black box methods, white box methods, error guessing, performance, test coverage metrics, validation, regression testing)
D. Verification techniques (e.g., test, demonstration, analysis, inspection, technical review)
E. Tools and techniques (e.g., debugging, scripting, libraries, functional execution, performance execution, expected/actual results analysis and reporting)

V. Maintenance **6**

A. Maintenance processes and activities (e.g., program comprehension, re-engineering, re-factoring, migration, disaster recovery techniques, service provisioning)
B. Problem management (e.g., user-reported incidents, maintainer-reported incidents, root cause analysis, defect tracking and resolution, impact analysis)
C. Pre-emptive maintenance metrics (e.g., complexity, error occurrence rates, failure rates, performance monitoring)

VI. Configuration Management **6**

A. Configuration identification (e.g., identifying items to be controlled, control methods, software library)
B. Configuration control (e.g., requesting, evaluating and approving software changes, approving releases)
C. Configuration status accounting (e.g., configuration status information, configuration status reporting)
D. Release management and delivery (e.g., software building, software release management, segregation of duties)

VII. Engineering Processes 6

A. Process definition (e.g., software life cycle models [Agile, Spiral, Waterfall, etc.], software life cycle processes, process tailoring, process assessment, process improvement)

B. Software engineering standards (e.g., process, coding, development, testing, reliability, architecture)

C. Estimation (e.g., software size and complexity, effort)

D. Measurement (e.g., metrics definition, collection, and analysis, goal question metric)

E. Risk management (e.g., addressing uncertainty, opportunities, decisions under risk, decisions under uncertainty)

VIII. Quality Assurance 6

A. Software quality fundamentals (e.g., organizational role, value and cost of quality, models and quality characteristics, software quality improvement)

B. Software quality management processes and systems (e.g., product assurance, process assurance, quality analysis and evaluation

C. Software quality techniques (e.g., reviews, audits, software quality requirements, defect characterization, software quality measurement, software quality tools)

IX. Safety, Security, and Privacy 12

A. Basic concepts (e.g., security versus privacy, intellectual property, confidentiality, integrity, availability, Common Criteria, component criticality)

B. Secure architecture and design (e.g., secure communications, disaster recovery, encryption, patterns and anti-patterns, infrastructure and environment planning)

C. Secure coding (e.g., secure subsets, encryption and keying, numerical precision, accuracy and errors)

D. Human–computer interface design (e.g., use of shape and color, response time, system navigation, consistency, error messages)

E. Safety issues (e.g., hazard analysis, failure analysis, fault-tolerant design, fail-safe design, fault-recovery)

F. Identity, authentication, and authorization (e.g., biometrics, password strength)

G. Threat analysis and remediation (e.g., spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege, assurance cases, security audits)

H. Security testing (e.g., penetration testing, intrusion detection, fuzz testing, fault injection)