SWEBOK Guide v4 Knowledge Area	SWEBOK Guide v4 section	%	# of Items (+/-2)
1. Software Requirements		**8%**	**5**
1. acceptance criteria	4.3		
2. techniques for identifying conflict(s) in requirements	3.1, 3.4		
3. definition of a well-formed requirement	3.1		
4. elicitation techniques (e.g., interviews, brainstorming, user story mapping)	2.0-2.2		
5. prototyping techniques	5.3		
6. requirement categories	1.2-1.10		
7. requirement sources (e.g., stakeholders, regulations, standards)	2.1		
8. requirement specification types (e.g., user stories, use case diagrams)	4.0-4.4		
9. requirements change control	6.0, 6.2		
10. requirements prioritization	7.2		
11. requirements quantification techniques (e.g., story points)	7.5		
12. requirements scrubbing	6.0-6.1		
13. requirements stability and volatility	7.4		
14. requirements tracing	7.3		
15. matching scope to project resources	6.0, 6.3		
16. simulation techniques	5.2		
17. requirements analysis (e.g., ambiguity, testability, binding)	3.0-3.3		
18. acceptance test and behavior driven development	4.3		
19. validation criteria (e.g., correctness, completeness, clarity, verifiability)	3.1, 5.0-5.3		
2. Software Architecture		**0%**	**0**
not on this exam			
3. Software Design		**10%**	**7**
1. accessibility considerations	1.3, 6.2		
2. design of APIs (application programming interfaces)	1.2, 5.6		
3. design measures (e.g., complexity)	6.4		
4. design methods (e.g., user-centered, component-based, event driven, functional)	5.5, 5.6		
5. design patterns (e.g., creational, structural, behavioral)	4.4		
6. design principles (e.g., abstraction, coupling and cohesion, separation of concerns, modularization)	1.4		
7. design characteristics (e.g., concurrency, persistence, distribution)	3.0-3.8		
8. design representations (e.g., UML, ERD, data flow diagrams)	4.0-4.6		
9. design strategies (e.g., top-down, iterative)	5.0-5.11		
10. Design Thinking process	1.1		
11. detailed design	2.2		
12. high-level design	2.1		
13. object-oriented design	5.4		
4. Software Construction		**12%**	**8**
1. code characteristics (e.g., modularity, reusability)	1.0-1.5		
2. code review practices	3.6		
3. complexity metrics (e.g., cyclomatic complexity, cyclic dependency, LCOM)	1.1		
4. construction standards	4.15		
5. construction techniques (e.g., assertions, defensive programming, concurrency primitives)	4.4, 4.10		
6. cross-platform development	3.8, 4.13		
7. debugging methods, tools, and techniques	See: [McConnell04] c23		
8. graphical user interface development	5.2		
9. performance analysis and tuning (e.g., construction profiling tools and code optimization)	4.14		
10. programming languages strengths and weaknesses	3.2, 4.2, 4.3, 4.6		
11. refactoring techniques	See: [McConnell04] c24		
12. software construction tools (e.g., development environments, debuggers, build pipeline)	5.1 - 5.4		
13. software integration approaches (e.g., incremental, continuous, big bang)	3.7		
14. software licensing types and implications	2.4		
15. static analysis	3.6, Quality: 3.4.1		
16. types of programming languages	3.2		
17. unit test-driven development	4.16		
18. organizational coding conventions	1.5		
5. Software Testing		**9%**	**6**
1. testing for accessibility	2.2.12		
2. defect-management processes	Maintenance: 2.1.2-2.1.3, 2.3.2, 3.2 Configuration: 3.0-3.3		
3. testing various characteristics of non-functional requirements	1.2.12, 2.2.7-2.2.9		
4. reusable test asset identification and management	5.1.9		
5. risk-based testing	1.2.3, 5.1.2		
6. software completion assessment (including criteria)	4.0-4.2, 5.1.8		
7. software test documentation	5.1.4, 5.2.5		
8. software test-related measures (e.g., test coverage)	4.0-4.2		

SWEBOK Guide v4 Knowledge Area	SWEBOK Guide v4 section	%	# of Items (+/-2)
9. software testing fundamentals	1.0 -1.3		
10. test levels (e.g., unit test, integration test)	2.0-2.2		
11. test techniques (e.g., boundary value, control flow, exploratory)	3.0-3.8		
12. testing tools and environments	8.0-8.2		
13. test-automation techniques	8.0-8.2		
6. Software Engineering Operations		**4%**	**2**
1. tools for continuous integration (CI) and continuous delivery (CD) deployment [CI/CD]	6.0, 6.2		
2. containerization, orchestration, and virtualization technologies and tools	6.1		
3. operations debugging methods and tools	6.4		
4. DevOps principles (e.g., collaboration, automation, continuous improvement)	1.1-1.2, 1.4		
5. installation and deployment technologies and techniques	1.3, 3.2, 3.4		
6. installation tools	6.2		
7. key performance indicators (e.g., response time, throughput, availability)	2.3-2.5		
8. observability stack (logs, metrics, traces)	4.2		
9. redundancy and fail-over mechanisms	3.3		
10. reliability techniques (e.g., error tracing, recovery, root cause)	1.6, 3.5, 4.1		
11. service discovery	Construction: 4.12		
12. service-level indicators (e.g., DevOps Research and Assessment [DORA] metrics)	2.4		
7. Software Maintenance		**6%**	**4**
1. tools for reverse-engineering design from code (e.g., code comprehension, design recovery, cross-referencer).	5		
2. importance of requirements and design documentation (including design rationales) for facilitating maintenance	2.1.1		
3. software-maintenance techniques (e.g., reengineering)	4.0-4.5		
4. software-maintenance categories (e.g., corrective, preventive, adaptive, perfective)	1.6		
5. impact analysis (e.g., code dependencies, system components, interface dependencies, testing implications, documentation)	2.1.3		
6. impact-analysis techniques (e.g., traceability matrix, code reviews, static analysis, slicing)	2.1.3		
8. Software Configuration Management		**4%**	**2**
1. configuration management processes that are relevant to the selected software lifecycle paradigm (e.g., iterative, predictive)	2, 3, 4, 6		
2. software configuration management (SCM) processes and tools (e.g., version control, dependency management, config items, release processes)	7		
3. software library management	2.6		
9. Software Engineering Management		**3%**	**2**
1. feasibility analysis (e.g., schedule, resources, technical difficulty, capability within cost)	1.2		
2. product-validation criteria [e.g., acceptance tests met, satisfaction criteria (definition of "done")]	5.0-5.2		
3. lean software development concepts (e.g., lead and cycle times, managing work-in-progress [WIP], flow metrics, Kanban)	Models & Methods: 4.4		
4. processing monitoring (e.g., burn down, backlog grooming) using application lifecycle management tools	3.4		
5. retrospective practices (e.g., at sprint end, at release, at project end)	4.2		
6. types of technical deliverables and their respective purposes and scopes	2.2, 3.2, Maintenance: 2.1.1		
7. short-term effort and scheduling estimation (e.g., iteration [sprint] level)	2.3		
10. Software Engineering Process		**4%**	**3**
1. empirical methods to support process assessment and improvement (e.g., feedback loops)	3.0-3.4		
2. empirical methods to support product assessment for process improvement (e.g., feedback loops)	2.11		
3. software engineering processes (e.g., build automation, deployment automation, continuous integration, continuous deployment)	1.2		
4. purposes, strengths, and weaknesses of various software-lifecycle paradigms	2.1-2.5, 2.8-2.9		
5. software-lifecycle paradigm standards	2.8, 3.3		
11. Software Engineering Models and Methods		**5%**	**3**
1. model-analysis criteria (e.g., completeness, consistency, correctness, traceability)	3.0-3.5		
2. strengths and weaknesses of various software-engineering models	2.0-2.2		
3. types of software-engineering methods (e.g., heuristic, formal, prototyping, agile)	4.0-4.4		
4. types of software-engineering models [e.g., information (database schema), behavioral (sequence diagram), structural (class diagram)]	2.0-2.2		
12. Software Quality		**6%**	**4**
1. value and costs of software quality	1.2		
2. defect classifications	2.4.1		
3. early detection techniques	1.2, 1.4, 3.4.5		
4. preventative and corrective actions	2.4		
6. quality assurance versus quality control	3.1, 3.4.4		
7. quality control techniques (e.g., verification and validation)	3.4		
8. quality process improvement (e.g., Kaizen, PDCA)	1.3, 2.1		

SWEBOK Guide v4 Knowledge Area	SWEBOK Guide v4 section	%	# of Items (+/-2)
9. quality requirements prioritization	1.1, 1.4, 3.3		
10. review and inspection techniques	3.4.5		
11. root-cause analysis	2.4, Engineering Foundations: 9		
12. software quality standards (e.g., ISO/IEC 25010)	1.3		
13. software quality tools	4		
14. technical debt (e.g., architecture, design, code)	Architecture: 2.4, Maintenance: 2.1.4, 2.3.1		
13. Software Security		**6%**	**4**
1. access-control principles and techniques	3.2, 4.2		
2. business continuity and disaster recovery concepts	Operations: 2.5		
3. credential-management principles and techniques	3.2		
4. data classification and protection	1.2, 2.2, Operations 2.6		
5. information security and cybersecurity fundamentals (e.g., security principles, security policies, security procedures)	1.3, 3.2		
6. network security fundamentals	Not in SWEBOK Guide		
7. secure coding	1.1, 4.4		
8. security considerations in design (e.g., cryptographic standards)	4.2, 4.3		
9. security incident categories	4.6		
10. security incident response	4.6, Operations 4.1		
11. vulnerability types and management (e.g., OWASP [Open Worldwide Application Security Project], Top 10, CERT [computer emergency response/readiness team] recommendations and standards)	4.6		
12. security considerations in requirements	4.1		
13. security tools (e.g., security scanning)	5.0-5.2		
14. supply-chain security management (e.g., control internal and external supply chain components, open source)	Not in SWEBOK Guide		
15. security considerations in testing	4.5		
16. threat modeling	4.2		
14. Software Engineering Professional Practice		**3%**	**2**
1. trade compliance regulations (e.g., trade, legal, international, export, safety, intellectual property) <<<broaded to engr professionalism?>>>	1.7.0-1.7.10		
2. import and export software classification (e.g., Export Administration Regulations assignation of EAR99)	Not in SWEBOK Guide		
15. Software Engineering Economics		**5%**	**3**
1. software engineering economics fundamentals	1.0-1.6		
2. engineering decision-making process	2.0-2.7		
3. estimation techniques (e.g., expert judgment, analogy, decomposition, and statistical)	8.0-8.5		
4. for-profit decision making	3.0-3.5		
5. multiple attribute decision making	6.0-6.2		
6. non-profit decision making (e.g., benefits/cost analysis, cost-effectiveness)	4.0-4.2		
7. present economy decision making	5.0 -5.2		
16. Computing Foundations		**7%**	**4**
1. computer networks and communications	7.0-7.9		
2. data structures and algorithms	3.0-3.9		
3. database management	6.0-6.7		
4. operating systems	5.0-5.5		
5. user and developer human factors (e.g., usability, accessibility, code readability)	8.0-8.2		
17. Mathematical Foundations		**4%**	**3**
1. basic logic principles (e.g., propositional, predicate)	1.0-1.2		
2. finite-state machines	5		
3. graph and tree manipulation	4.0-4.2		
4. numerical precision, accuracy, and error	10		
5. proof techniques	2.0-2.4		
18. Engineering Foundations		**4%**	**3**
1. measurement theory	7.0-7.3		
2. modeling, simulation, and prototyping	6.0-6.3		
3. statistical analysis	5.0-5.2		
Last Updated - March 21, 2025 - 80 Total Items - 15 Unscored 65 Scored			