APPENDIX D

CLASSIFICATION OF TOPICS ACCORDING TO BLOOM'S TAXONOMY

INTRODUCTION

Bloom's taxonomy[1] is a well-known and widely used classification of cognitive educational goals. In order to help audiences who wish to use the Guide as a tool in defining course material, university curricula, university program accreditation criteria, job descriptions, role descriptions within a software engineering process definition, professional development paths and professional training programs and other needs, Bloom's taxonomy levels for SWEBOK Guide topics are proposed in this appendix for a software engineering graduate with four years of experience. A software engineering graduate with four years of experience is in essence the "target" of the SWEBOK Guide as defined by what is meant by generally accepted knowledge (See Introduction of the SWEBOK Guide).

Since this Appendix only pertains to what can be considered as "generally accepted" knowledge, it is very important to remember that a software engineer must know substantially more than this "category" of knowledge. In addition to "generally accepted" knowledge, a software engineering graduate with four years of knowledge must possess some elements from the Related Disciplines as well as certain elements of specialized knowledge, advanced knowledge and possibly even research knowledge (see Introduction of the SWEBOK Guide).

The following assumptions were made when specifying the proposed taxonomy levels:

- The evaluations are proposed for a "generalist" software engineer and not a software engineer working in a specialized group such as a software configuration management team, for instance. Obviously, such a software engineer would require or would attain much higher taxonomy levels in the specialty area of their group;
- A software engineer with four years of experience is still at the beginning of their career and would be assigned relatively few management duties, or at least not for major endeavors. "Management-related topics" are therefore not given priority in the proposed evaluations. For the same reason, taxonomy levels tend to be lower for "early-life cycle topics" such as those related to software requirements than for more technically-oriented topics such as those within software design, software construction or software testing.
- So the evaluations can be adapted for more senior software engineers or software engineers specializing in certain knowledge areas, no topic is given a taxonomy level higher than Analysis. This is consistent with the approach taken in the Software Engineering Education Body of Knowledge (SEEK) where no topic is assigned a taxonomy level higher than Application[2]. The purpose of SEEK is to define a software engineering education body of knowledge appropriate for guiding the development of undergraduate software engineering curricula. Though distinct notably in terms of scope, SEEK and the SWEBOK Guide are closely related[3].

Bloom's Taxonomy of the Cognitive Domain proposed in 1956 contains six levels. Table 1[4] presents these levels and keywords often associated with each level.

[1] B. Bloom (Eds), Taxonomy of Educational Objectives: The Classification of Educational Goals, Mackay, 1956.

[2] See Joint Task Force on Computing Curricula – IEEE Computer Society Association for Computing Machinery, Computing Curricula – Software Engineering Volume – Publid Draft 1 – Computing Curriculum Software Engineering, 2003. http://sites.computer.org/ccse/

[3] See P Bourque, F. Robert, J.-M. Lavoie, A. Lee, S. Trudel, T. Lethbridge, "Guide to the Software Engineering Body of Knowledge (SWEBOK) and the Software Engineering Education Body of Knowledge (SEEK) – A Preliminary Mapping", in *Proc. Tenth Intern. Workshop Software Technology and Engineering Practice Conference (STEP 2002)*, pp. 8-35, 2002)

[4] Table taken from http://www.nwlink.com/~donclark/hrd/bloom.html

Table 1 Bloom's Taxonomy

Bloom's Taxonomy Level	Associated Keywords
Knowledge: Recall of data.	Defines, describes, identifies, knows, labels, lists, matches, names, outlines, recalls, recognizes, reproduces, selects, states.
Comprehension: Understand the meaning, translation, interpolation, and interpretation of instructions and problems. State a problem in one's own words.	Comprehends, converts, defends, distinguishes, estimates, explains, extends, generalizes, gives examples, infers, interprets, paraphrases, predicts, rewrites, summarizes, translates.
Application: Use a concept in a new situation or unprompted use of an abstraction. Applies what was learned in the classroom into novel situations in the workplace.	Applies, changes, computes, constructs, demonstrates, discovers, manipulates, modifies, operates, predicts, prepares, produces, relates, shows, solves, uses.
Analysis: Separates material or concepts into component parts so that its organizational structure may be understood. Distinguishes between facts and inferences.	Analyzes, breaks down, compares, contrasts, diagrams, deconstructs, differentiates, discriminates, distinguishes, identifies, illustrates, infers, outlines, relates, selects, separates.
Synthesis: Builds a structure or pattern from diverse elements. Put parts together to form a whole, with emphasis on creating a new meaning or structure	Categorizes, combines, compiles, composes, creates, devises, designs, explains, generates, modifies, organizes, plans, rearranges, reconstructs, relates, reorganizes, revises, rewrites, summarizes, tells, writes.
Evaluation: Make judgments about the value of ideas or materials.	Appraises, compares, concludes, contrasts, criticizes, critiques, defends, describes, discriminates, evaluates, explains, interprets, justifies, relates, summarizes, supports.

The breakdown of topics in the tables does not match perfectly tha breakdown in the Knowledge Areas. The evaluation for this Appendix was prepared while some comments were still coming in.

Finally, please bear in mind that the evaluations of this Appendix should definitely only be seen as a proposal to be further developed and validated.

SOFTWARE REQUIREMENTS[5]

Breakdown of Topics	Taxonomy Level
1. Software requirements fundamentals	
Definition of software requirement	C
Product and process requirements	C
Functional and non-functional requirements	C
Emergent properties	C
Quantifiable requirements	C
System requirements and software requirements	C
2. Requirements process	
Process models	C
Process actors	C
Process support and management	C
Process quality and improvement	C
3. Requirements elicitation	
Requirements sources	C
Elicitation techniques	AP
4. Requirements analysis	
Requirements classification	AP
Conceptual modeling	AN
Architectural design and requirements allocation	AN
Requirements negotiation	AP
5. Requirements specification	
System definition document	C
System requirements specification	C
Software requirements specification	AP
6. Requirements validation	
Requirements reviews	AP
Prototyping	AP
Model validation	C
Acceptance tests	AP
7. Practical Considerations	
Iterative nature of requirements process	C
Change management	AP
Requirements attributes	C
Requirements tracing	AP
Measuring requirements	AP

[5] K: Knowledge, C: Comprehension, AP: Application, AN: Analysis, E: Evaluation, S: Synthesis

SOFTWARE DESIGN

Breakdown of Topics	Taxonomy Level
1. Software Design Fundamentals	
General design concepts	C
Context of software design	C
Software design process	C
Enabling techniques	AN
2. Key issues in software design	
Concurrency	AP
Control and handling of events	AP
Distribution of components	AP
Error and exception handling and fault tolerance	AP
Interaction and presentation	AP
Data persistence	AP
3. Software structure and architecture	
Architectural structures and viewpoints	AP
Architectural styles (macroarchictural patterns)	AN
Design patterns (microarchitectural patterns)	AN
Families of programs and frameworks	C
4. Software design quality analysis and evaluation	
Quality attributes	C
Quality analysis and evaluation techniques	AN
Measures	C
5. Software design notations	
Structural descriptions (static)	AP
Behavioral descriptions (dynamic)	AP
6. Software design strategies and methods	
General strategies	AN
Function-oriented (structured) design	AP
Object-oriented design	AN
Data-structure centered design	C
Component-based design (CBD)	C
Other methods	C

SOFTWARE CONSTRUCTION

Breakdown of Topics	Taxonomy Level
1. Software construction fundamentals	
Minimizing complexity	AN
Anticipating change	AN
Constructing for verification	AN
Standards in construction	AP
2. Managing construction	
Construction methods	C
Construction planning	AP
Construction measurement	AP
3. Practical considerations	
Construction design	AN
Construction languages	AP
Coding	AN
Construction testing	AP
Construction quality	AN
Integration	AP

SOFTWARE TESTING

Breakdown of Topics	Taxonomy Level
1. Software testing fundamentals	
Testing-related terminology	C
Key issues	AP
Relationships of testing to other activities	C
2. Test levels	
The target of the tests	AP
Objectives of testing	AP
3. Test techniques	
Based on tester's intuition and experience	AP
Specification-based	AP
Code-based	AP
Fault-based	AP
Usage-based	AP
Based on nature of application	AP
Selecting and combining techniques	AP
4. Test related measures	
Evaluation of the program under test	AN
Evaluation of the tests performed	AN
5. Test process	
Management concerns	C
Test activities	AP

SOFTWARE MAINTENANCE

	Taxonomy Level
1. Software maintenance fundamentals	
Definitions and terminology	C
Nature of maintenance	C
Need for maintenance	C
Majority of maintenance costs	C
Evolution of software	C
Categories of maintenance	AP
2. Key issues in software maintenance	
Technical	
Limited Understanding	C
Testing	AP
Impact Analysis	AN
Maintainability	AN
Management issues	
Alignment with organizational issues	C
Staffing	C
Process issues	C
Organizational	C
Maintenance cost estimation	
Cost estimation	AP
Parametric models	C
Experience	AP
Software maintenance measurement	AP
3. Maintenance process	
Maintenance process models	C
Maintenance activities	
Unique Activities	AP
Supporting Activities	AP
4. Techniques for maintenance	
Program comprehension	AN
Re-engineering	C
Reverse engineering	C

SOFTWARE CONFIGURATION MANAGEMENT

Breakdown of Topics	Taxonomy Level
1. Management of the SCM Process	
Organizational context for SCM	C
Constraints and guidance for SCM	C
Planning for SCM	
SCM organization and responsibilities	AP
SCM resources and schedules	AP
Tool selection and implementation	AP
Vendor/Subcontractor control	C
Interface control	C
Software configuration management plan	C
Surveillance of software configuration management	
SCM measures and measurement	AP
In-Process audits of SCM	C
2. Software Configuration Identification	
Identifying items to be controlled	
Software configuration	AP
Software configuration items	AP
Software configuration item relationships	AP
Software versions	AP
Baseline	AP
Acquiring software configuration items	AP
Software library	C
3. Software Configuration Control	
Requesting, evaluating and approving software changes	
Software configuration control board	AP
Software change request process	AP
Implementing software changes	AP
Deviations & waivers	C
4. Software Configuration Status Accounting	
Software configuration status information	C
Software configuration status reporting	AP
5. Software Configuration Auditing	
Software functional configuration audit	C
Software physical configuration audit	C
In-Process audits of a software baseline	C
6. Software Release Management and Delivery	
Software building	AP
Software release management	C

SOFTWARE ENGINEERING MANAGEMENT

	Taxonomy Level
1. Initiation and scope definition	
Determination and negotiation of requirements	AP
Feasibility analysis	AP
Process for requirements review/revision	C
2. Software project planning	
Process planning	C
Determine deliverables	AP
Effort, schedule and cost estimation	AP
Resource allocation	AP
Risk management	AP
Quality management	AP
Plan management	C
3. Software project enactment	
Implementation of plans	AP
Supplier contract management	C
Implementation of measurement process	AP
Monitor process	AN
Control process	AP
Reporting	AP
4. Review and evaluation	
Determining satisfaction of requirements	AP
Reviewing and evaluating performance	AP
5. Closure	
Determining closure	AP
Closure activities	AP
6. Software Engineering Measurement	
Establish and sustain measurement commitment	C
Plan the measurement process	C
Perform the measurement process	C
Evaluate measurement	C

SOFTWARE ENGINEERING PROCESS

	Taxonomy Level
1. Process implementation and change	
Process infrastructure	
Software engineering process group	C
Experience factory	C
Activities	AP
Models for process implementation and change	K
Practical considerations	C
2. Process definition	
Life cycle models	AP
Software life cycle processes	C
Notations for process definitions	C
Process adaptation	C
Automation	C
3. Process assessment	
Process assessment models	C
Process assessment methods	C
4. Product and process measurement	
Software process measurement	AP
Software product measurement	AP
Size measurement	AP
Structure measurement	AP
Quality measurement	AP
Quality of measurement results	AN
Software information models	
Model building	AP
Model implementation	AP
Measurement techniques	
Analytic techniques	AP
Benchmarking techniques	C

SOFTWARE ENGINEERING TOOLS AND METHODS

Breakdown of Topics	Taxonomy Level
1. Software tools	
Software requirements tools	AP
Software design tools	AP
Software construction tools	AP
Software testing tools	AP
Software maintenance tools	AP
Software engineering process tools	AP
Software quality tools	AP
Software configuration management tools	AP
Software engineering management tools	AP
Miscellaneous tool issues	AP
2. Software engineering methods	
Heuristic methods	AP
Formal methods and notations	C
Prototyping methods	AP
Miscellaneous method issues	C

SOFTWARE QUALITY

	Taxonomy Level
1. Software quality fundamentals	
Software engineering culture and ethics	AN
Value and costs of quality	AN
Quality models and characteristics	
Software process quality	AN
Software product quality	AN
Quality improvement	AP
2. Software quality management processes	
Software quality assurance	AP
Verification and validation	AP
Reviews and audits	
Inspections	AP
Peer reviews	AP
Walkthroughs	AP
Testing	AP
Audits	C
3. Practical considerations	
Application quality requirements	
Criticality of systems	C
Dependability	C
Integrity levels of software	C
Defect characterization	AP
Software quality management techniques	
Static techniques	AP
People-intensive techniques	AP
Analytic techniques	AP
Dynamic techniques	AP
Software quality measurement	AP