Klassifikation softwaretechnischer Themengebiete

Udo Kelter

17.10.2012

Zusammenfassung dieses Lehrmoduls

Dieses Lehrmodul stellt zwei Klassifikationen softwaretechnischer Themengebiete vor: den IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK®), der das zu erwartende softwaretechnische Wissen eines "typischen Software-Ingenieurs" sehr detailliert beschreibt, und Anteile des Computing Classification Systems der ACM, das vor allem zur Klassifizierung von Literatur dient. Ferner wird analysiert, wie sich die inzwischen vorgeschriebenen Akkreditierungsverfahren auf die softwaretechnischen Anteile in Informatikstudiengängen auswirken.

Vorausgesetzte Lehrmodule: keine

Stoffumfang in Vorlesungsdoppelstunden: 0.6

Inhaltsverzeichnis

1 Klassifikation softwaretechnischer Themengebiete nach dem IEEE Guide to the Software Engineering Body of Knowledge

Innerhalb der IEEE Computer Society hat das "IEEE Computer Society Professional Practices Committee" in einem mehrjährigen Prozeß den Versuch unternommen, die Themengebiete zu beschreiben,

1. in denen allgemein anerkanntes und als wesentlich erachtetes Wissen über softwaretechnische Fragestellungen vorliegt und
2. in denen ein "typischer Software-Ingenieur" nach einem Bachelor-Studium und ca. "4 Jahren Erfahrung" über Kompetenzen verfügen muß.

Ergebnis ist der *IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK®)* [GSB04]. Der Sinn dieser Beschreibung von Wissensgebieten wird insb. darin gesehen, ein Kompetenzprofil zu definieren, das professionelle Softwaretechniker gegenüber weniger gründlich ausgebildeten Berufstätigen in der IT-Branche auszeichnet.

Ein Hintergrund hiervon sind seit langer Zeit laufende Überlegungen, für die Entwicklung sicherheitskritischer Software eine Art Approbation zu verlangen, also eine staatliche Genehmigung zur Berufsausübung, die nur nach einer erfolgreichen akademischen Ausbildung und einer anschließenden erfolgreichen beruflichen Praxis verliehen wird. Motiviert ist dies durch die jährlichen immensen Schäden, angefangen von finanziellen Schäden in Milliardenhöhe bis hin zu Menschenleben, die durch fehlerhafte Software verursacht werden.

Einige der im SWEBOK definierten Kompetenzen wird man eher in der beruflichen Praxis erwerben können als in einem universitären Kontext.

1.1 Hauptgebiete

Es werden folgende Hauptgebiete (*"knowledge areas"; KAs*) unterschieden:

1. Software Requirements
2. Software Design
3. Software Construction
4. Software Testing
5. Software Maintenance
6. Software Configuration Management
7. Software Engineering Management
8. Software Engineering Process
9. Software Engineering Tools And Methods
10. Software Quality

Jedem Hauptgebiet ist ein Kapitel gewidmet, in dem die Unterbereiche beschrieben werden; ein lokaler Anhang enthält jeweils Referenzen auf Lehrbücher und andere Quellen, in denen typische konkrete Stoffinhalte im Detail beschrieben sind[1]. Konkret zu beherrschende Sprachen oder Stoffinhalte sind i.d.R. nicht vorgeschrieben, i.d.R. sind eher Problembereiche beschrieben, in denen *irgendeine* konkrete Technik beherrscht werden sollte. Im Anhang dieses Lehrmoduls ist das detaillierte Inhaltsverzeichnis der Version des Guide von 2004 angegeben.

Die Wissensgebiete sind nicht disjunkt, sondern überschneiden sich vielfach, weil bestimmte Wissenskomplexe aus verschiedenen Perspektiven klassifiziert werden.

1.2 Ergänzende Kompetenzen und Überschneidungen mit anderen Wissensgebieten

Kompetenzen auf den o.g. softwaretechnischen Themengebieten sind nur ein (großer) Teil des berufsqualifizierenden Wissens und müssen ergänzt werden um Kompetenzen in den typischen Informatik-"Hauptfächern" wie z.B. Programmierung und Datenstrukturen, Datenbanksysteme, Betriebssysteme, Rechnernetze usw., ferner ggf. in einem

[1] Zitat: "*The Guide should not be confused with the Body of Knowledge itself, which already exists in the published literature. The purpose of the Guide is to describe what portion of the Body of Knowledge is generally accepted, to organize that portion, and to provide a topical access to it.*"

Anwendungsgebiet. Insofern wird der Begriff "Software Engineering" leider doppeldeutig benutzt:

1. im weiten Sinne eines *Berufsbilds* als Menge berufsqualifizierender Fähigkeiten, wenn von *Recognized Profession / Characteristics of a Profession* die Rede ist;
2. im engeren Sinne eines *Hauptfachs* bzw. einer Fächergruppe in Informatik-Studiengängen.

Mit allen vorstehend genannten Informatik-Hauptfächern bestehen signifikante Überschneidungen. Die benachbarten Informatik-Hauptfächer werden im SWEBOK nicht behandelt, ebenso wird nicht versucht, eine klare Grenze zu ziehen (was auch schwierig wäre).

Systembedingt kann das SWEBOK auch eine Fähigkeit von Informatikern, die aus Sicht von Arbeitgebern extrem wichtig ist, nicht beschreiben: die Fähigkeit, sich in neue Technologien rasch und möglichst selbständig einzuarbeiten.

1.3 Kompetenzstufen

Die Beschreibungen der Wissensgebiete in den Hauptkapiteln des Guide enthalten keine Angaben zum Vertiefungsgrad, in dem das jeweilige Wissen beherrscht werden muß. Hierzu finden sich erst in Appendix D Angaben, die zu jedem Unterbereich pauschal eine Kompetenzstufe angeben. Zur Beschreibung der Kompetenzstufen wird eine aus [Bl56] stammende Taxonomie verwendet, die folgende Stufen benutzt:

A Wissen (*Knowledge*): (unreflektiertes) Faktenwissen

B Verstehen (*Comprehension*): Verstehen von Begriffen oder Verfahren, so daß sie mit eigenen Worten dargestellt oder übersetzt werden können

C Anwendung (*Application*): die Fähigkeit, Konzepte und Verfahren in konkreten Anwendungssituationen anwenden zu können

D Analyse (*Analysis*): Fähigkeit, das Wissen auf seine Hintergründe und Komponenten hin zu analysieren und zu organisieren und zwischen grundlegendem und ableitbarem Wissen zu unterscheiden

E Synthese (*Synthesis*): Fähigkeit, einzelne Wissensbestandteile neu zu arrangieren und zu kombinieren, um neues Wissen zu erzeugen

F Bewertung (*Evaluation*): Fähigkeit, Wissen zu vergleichen, einzuordnen und zu bewerten

In dem Inhaltsverzeichnis des SWEBOK, das im Anhang beigefügt ist, sind die vorstehend definierten Kennbuchstaben links neben den Wissensgebieten angeordnet. In rund der Hälfte der Fälle wird Stufe C verlangt, bei ca. 35 % Stufe B, und 15 % Stufe D. Die angegebenen Kompetenzstufen werden als Mindestanforderungen angesehen.

Kein Gegenstand des SWEBOK sind Beschreibungen von graduellen Unterschieden, also was den Unterschied zwischen einem "sehr guten" und einem "befriedigenden" Kompetenzgrad ausmacht. Ebenfalls nicht adressiert werden Produktivitätsmaße. Zwei Entwickler können beide im Prinzip imstande sein, mittelgroße Java-Programme zu entwickeln, sind also qualitativ vergleichbar. Dennoch braucht der eine ggf. zehnmal soviel Zeit für die gleiche Aufgabe wie der andere, ist also wesentlich unproduktiver.

Wegen der eher vagen Beschreibung der Wissensgebiete und der relativ groben Stufung der Kompetenzstufen bleibt im Endeffekt ein ganz erheblicher Beurteilungsspielraum, ob jemand den hier verlangten Wissensstand erfüllt (im Sinne einer binären Note).

Nicht vorhanden sind Angaben zur Priorität oder zur Praxisrelevanz der einzelnen Gebiete (die durchaus ungleichmäßig erscheint) und Einschätzungen des Schwierigkeitsgrads oder Lernaufwands.

1.4 Curriculare Umsetzung

Der Guide läßt es völlig offen, wie das beschriebene Kompetenzprofil erreicht werden kann.

Einige Kompetenzen sind in einem universitären Kontext nur schwer vermittelbar und erst nach einiger Erfahrung sinnvoll erwerbbar; es bietet sich an, diese im Rahmen einer beruflichen Weiterqualifikation zu erwerben.

Wegen der vielen Überlappungen mit anderen Informatik-Hauptfächern ist es durchaus möglich, daß viele Kompetenzen in Lehrveranstaltungen vermittelt werden, die nicht als Softwaretechnik-Lehrveranstaltungen bezeichnet werden. In [CCSE03] wird eine Vielzahl von Curriculumsvarianten vorgestellt, mit denen das im Guide beschriebene Kompetenzprofil erreicht werden kann.

2 Klassifikation softwaretechnischer Themengebiete nach dem Computing Classification System der ACM

Die Association for Computing Machinery (ACM) unterhält seit sehr langer Zeit das **Computing Classification System** [ACMCCS1998, ACMCCS2012], ein Klassifikationssystem, das die komplette Informatik in eine Vielzahl von Teilgebieten untergliedert. Das System ist vor allem mit der Intention gebildet worden, wissenschaftliche Literatur thematisch einordnen zu können.

Das System existiert bereits seit Jahrzehnten (Erstversion von 1964) und steht vor dem Problem, daß sich in derart langen Zeiträumen Technologien, Begriffe und Themenschwerpunkte erheblich verändern. Etwa alle 10 Jahre ist daher eine Generalüberarbeitung notwendig.

Die Softwaretechnik bildet in dem Klassifikationssystem den Bereich *D.2 Software Engineering*, der wie folgt weiter untergliedert ist:

D.2.0 Allgemeines (*General*)

D.2.1 Anforderungsanalyse (*Requirements/Specifications*)

D.2.2 Entwurfsmethoden (*Design Tools and Techniques*)

D.2.3 Programmierung (*Coding Tools and Techniques*)

D.2.4 Verifikation (*Software/Program Verification*)

D.2.5 Testen und Fehlerbehebung (*Testing and Debugging*)

D.2.6 Software-Entwicklungsumgebungen (*Programming Environments*)

D.2.7 Software-Distribution und -Wartung (*Distribution, Maintenance, and Enhancement*)

D.2.8 Metriken (*Metrics*)

D.2.9 Management (*Management*)

D.2.10 Entwurf (*Design*) (wird nicht mehr benutzt; ersetzt durch D.2.2)

D.2.11 Software-Architekturen (*Software Architectures*)

D.2.12 Interoperabilität (*Interoperability*)

D.2.13 Wiederverwendung (*Reusable Software*)

D.2.m Verschiedenes (*Miscellaneous*)

Die einzelnen Themenbiete werden nur mit 5 - 10 weiteren Schlagworten umrissen.

3 Kompetenzprofile von Informatik-Studiengängen aus Sicht von Akkreditierungsagenturen

Wie schon im Abschnitt über das SWEBOK erwähnt muß man trennen zwischen erreichten Kompetenzen und dem Weg dorthin, also der Struktur von Studiengängen.

In früheren Zeiten, d.h. vor der Einführung von Bachelor- und Masterstrukturen - waren (Informatik-) Studiengänge vor allem durch ihr Curriculum definiert; die Kompetenzen, die ein Informatikstudiengang vermittelte, konnte man zwar in der Praxis gut einschätzen, sie waren aber formell nur vage fixiert. Bei Studiengängen, die den Studierenden relativ viele individuelle Schwerpunktsetzungen durch Wahlpflichtbereiche und den Lehrenden bei der inhaltlichen Ausgestaltung einzelner Fächer viel Flexibilität erlauben, ist dies kaum anders denkbar.

Im Rahmen der Umstellung auf Bachelor- und Masterstrukturen wurde gleichzeitig massiver politischer Druck ausgeübt, die Ausbildung stärker zu rationalisieren, also zumindest in den Bachelorstudiengängen

die Wahlmöglichkeiten einzuschränken, und die erreichten Kompetenzen vorab genau einzugrenzen und die Qualifikation der Absolventen reproduzierbar zu machen.

Da Bachelor- und Masterstudiengänge oft die gleichen Themenfelder behandeln, entstand ferner das neue Problem, unterschiedliche Qualifikationsniveaus von Bachelor- und Masterstudiengängen definieren zu müssen.

Die Durchführung und inhaltliche Ausgestaltung der politischen Vorgaben wurde sog. Akkreditierungsagenturen übertragen. Es gibt mehrere Akkreditierungsagenturen, Informatik-Studiengänge wurden meistens bei der ASIIN e.V. (Akkreditierungsagentur für Studiengänge der Ingenieurwissenschaften, der Informatik, der Naturwissenschaften und der Mathematik e.V.) akkreditiert. Diese hat in [ASIIN06] fachspezifische Kriterien definiert, die Bachelor- und Masterstudiengänge der Informatik erfüllen müssen[2]. Die allgemeinen Kompetenzziele von Bachelor- und Masterstudiengängen sind dort wie folgt gruppiert:

1. fachliche Kompetenzen:
   - formale, algorithmische, mathematische Kompetenzen
   - Analyse-, Design- und Realisierungs-Kompetenzen
   - technologische Kompetenzen
   - fachübergreifende Kompetenzen
   - Methodenkompetenzen
2. soziale Kompetenzen
   - Projektmanagement-Kompetenz
   - soziale Kompetenz und Selbstkompetenz

[2] Andere Fächer haben fachspezifische Kriterien, die vielfach erheblich anders strukturiert sind. Hiermit sollen einerseits die speziellen Verhältnisse und Fachkulturen berücksichtigt werden, andererseits entsteht ein gewisser Eindruck von Willkürlichkeit bei der Wahl der Kriterien. Ein Indiz hierfür liefert ein größeres Projekt, in dem die akademischen Kompetenzziele mehrerer Ingenieurstudiengänge verglichen wurden, s. [CsR09]; dort wurde wieder eine andere Strukturierung der Kompetenzziele benutzt.

Für Bachelor- und Masterstudiengänge werden hierzu jeweils unterschiedliche Qualifikationsniveaus definiert, die aber willkürlich interpretierbar erscheinen.

Ferner werden abhängig von Studiengangstypen minimale und maximale Anteile dieser Themenfelder definiert, gemessen in Leistungspunkten. Dies kann im Einzelfall bedeuten, daß ein Modul in mehrere Anteile aufgeteilt werden muß, die z.B. technologische bzw. soziale Kompetenzen vermitteln.

Man erkennt unschwer, daß viele Themenfelder des SWEBOK auch hier auftauchen. Wegen der unklaren Qualifikationsniveaus und den eher dubiosen Methoden der Zurechnung von Modulanteilen kann man jedoch nicht sinnvoll rückwärts rechnen, wie umfangreiche softwaretechnische Anteile ein Informatik-Studiengang enthalten sollte, und erst recht nicht, welche konkreten Anteile. Immerhin kann vermutet werden, daß die fachspezifischen Kriterien den Umfang und Vertiefungsgrad der sozialen Kompetenzen (die in etwa dem Abschnitt 7. Software Engineering Management im SWEBOK entsprechen) gegenüber früheren Studiengängen merklich erhöhen wollten.

A Inhaltsverzeichnis des SWEBOK

A.0.1 Foreword

Associate Editors ... Industrial Advisory Board ... Panel Of Experts ... Review Team

A.0.2 Preface

Purpose ... Intended Audience ... Evolution Of The Guide ... Changes Since The Trial Version ... Limitations ... Acknowledgments

A.0.3 Chapter 1 - Introduction To The Guide

What Is Software Engineering? ... What Is A Recognized Profession? ... What Are The Objectives Of The SWEBOK Project? ... Reference Material And Matrix ... Depth Of Treatment ... Limitations Related To The Book Format ... The Knowledge Areas ... Structure Of The KA Descriptions ... Software Requirements KA ... Software Design KA ... Software Construction KA ... Software Testing ... Software Maintenance ... Software Configuration Management ... Software Engineering Management ... Software Engineering Process ... Software Engineering Tools And Methods ... Software Quality ... Related Disciplines Of Software Engineering ... Appendices ... Appendix A. KA Description Specifications ... Appendix B. Evolution Of The Guide ... Appendix C. Allocation Of Standards To KAs ... Appendix D. Bloom Ratings

A.0.4 Chapter 2 - Software Requirements

Acronyms ... Introduction ... Breakdown Of Topics For Software Requirements

[ ] 1. Software Requirements Fundamentals
[B] 1.1. Definition of a Software Requirement
[B] 1.2. Product and Process Requirements
[B] 1.3. Functional and Nonfunctional Requirements
[B] 1.4. Emergent Properties
[B] 1.5. Quantifiable Requirements
[B] 1.6. System Requirements and Software Requirements
[ ] 2. Requirements Process
[B] 2.1. Process Models
[B] 2.2. Process Actors
[B] 2.3. Process Support and Management
[B] 2.4. Process Quality and Improvement
[ ] 3. Requirements Elicitation
[B] 3.1. Requirements Sources

[C] 3.2. Elicitation Techniques
[ ] 4. Requirements Analysis
[C] 4.1. Requirements Classification
[D] 4.2. Conceptual Modeling
[D] 4.3. Architectural Design and Requirements Allocation
[D] 4.4. Requirements Negotiation
[ ] 5. Requirements Specification
[B] 5.1. The System Definition Document
[B] 5.2. System Requirements Specification
[C] 5.3. Software Requirements Specification
[ ] 6. Requirements validation
[C] 6.1. Requirements Reviews
[C] 6.2. Prototyping
[B] 6.3. Model Validation
[C] 6.4. Acceptance Tests
[ ] 7. Practical Considerations
[B] 7.1. Iterative Nature of the Requirements Process
[C] 7.2. Change Management
[B] 7.3. Requirements Attributes
[C] 7.4. Requirements Tracing
[C] 7.5. Measuring Requirements

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Requirements ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.5 Chapter 3 - Software Design

Acronyms ... Introduction ... Breakdown Of Topics For Software Design

[ ] 1. Software Design Fundamentals
[B] 1.1. General Design Concepts
[B] 1.2. Context of Software Design
[B] 1.3. Software Design Process
[ ] 1.3.1. Architectural design
[ ] 1.3.2. Detailed design
[D] 1.4. Enabling Techniques
[ ] 1.4.1. Abstraction
[ ] 1.4.2. Coupling and cohesion
[ ] 1.4.3. Decomposition and modularization
[ ] 1.4.4. Encapsulation/information hiding
[ ] 1.4.5. Separation of interface and implementation
[ ] 1.4.6. Sufficiency, completeness and primitiveness
[ ] 2. Key Issues in Software Design

[C] 2.1. Concurrency
[C] 2.2. Control and Handling of Events
[C] 2.3. Distribution of Components
[C] 2.4. Error and Exception Handling and Fault Tolerance
[C] 2.5. Interaction and Presentation
[C] 2.6. Data Persistence
[ ] 3. Software Structure and Architecture
[C] 3.1. Architectural Structures and Viewpoints
[D] 3.2. Design Patterns (microarchitectural patterns)
[B] 3.3. Families of Programs and Frameworks
[ ] 4. Software Design Quality Analysis and Evaluation
[B] 4.1. Quality Attributes
[D] 4.2. Quality Analysis and Evaluation Techniques
[B] 4.3. Measures
[ ] 5. Software Design Notations
[C] 5.1. Structural Descriptions (static view)
[C] 5.2. Behavioral Descriptions (dynamic view)
[ ] 6. Software Design Strategies and Methods
[D] 6.1. General Strategies
[C] 6.2. Function-Oriented (Structured) Design
[D] 6.3. Object-Oriented Design
[B] 6.4. Data-Structure-Centered Design
[B] 6.5. Component-Based Design (CBD)
[B] 6.6. Other Methods

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Design ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.6 Chapter 4 - Software Construction

Acronyms ... Introduction ... Breakdown Of Topics For Software Construction

[ ] 1. Software Construction Fundamentals
[D] 1.1. Minimizing Complexity
[D] 1.2. Anticipating Change
[D] 1.3. Constructing for Verification
[C] 1.4. Standards in Construction
[ ] 2. Managing Construction
[B] 2.1. Construction Models
[C] 2.2. Construction Planning
[C] 2.3. Construction Measurement
[ ] 3. Practical considerations
[D] 3.1. Construction Design
[C] 3.2. Construction Languages

[D] 3.2. Coding
[C] 3.3. Construction Testing
[??] 3.4. Reuse
[D] 3.5. Construction Quality
[C] 3.7. Integration

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Construction ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.7 Chapter 5 - Software Testing

Acronyms ... Introduction ... Breakdown Of Topics For Software Testing

[ ] 1. Software Testing Fundamentals
[B] 1.1. Testing-related terminology
[ ] 1.1.1. Definitions of testing and related terminology
[ ] 1.1.2. Faults vs. Failures
[C] 1.2. Key issues
[ ] 1.2.1. Test selection criteria/Test adequacy criteria (or stopping rules)
[ ] 1.2.2. Testing effectiveness/Objectives for testing
[ ] 1.2.3. Testing for defect identification
[ ] 1.2.4. The oracle problem
[ ] 1.2.5. Theoretical and practical limitations of testing
[ ] 1.2.6. The problem of infeasible paths
[ ] 1.2.7. Testability
[B] 1.3. Relationships of testing to other activities
[ ] 2. Test Levels
[C] 2.1. The target of the test
[ ] 2.1.1. Unit testing
[ ] 2.1.2. Integration testing
[ ] 2.1.3. System testing
[C] 2.2. Objectives of Testing
[ ] 2.2.1. Acceptance/qualification testing
[ ] 2.2.2. Installation testing
[ ] 2.2.3. Alpha and beta testing
[ ] 2.2.4. Conformance testing/Functional testing/Correctness testing
[ ] 2.2.5. Reliability achievement and evaluation
[ ] 2.2.6. Regression testing
[ ] 2.2.7. Performance testing
[ ] 2.2.8. Stress testing
[ ] 2.2.9. Back-to-back testing
[ ] 2.2.10. Recovery testing
[ ] 2.2.11. Configuration testing

[ ] 2.2.12. Usability testing
[ ] 2.2.13. Test-driven development
[ ] 3. Test Techniques
[C] 3.1. Based on the software engineer's intuition and experience
[ ] 3.1.1. Ad hoc testing
[ ] 3.1.2. Exploratory testing
[C] 3.2. Specification-based techniques
[ ] 3.2.1. Equivalence partitioning
[ ] 3.2.2. Boundary-value analysis
[ ] 3.2.3. Decision table
[ ] 3.2.4. Finite-state machine-based
[ ] 3.2.5. Testing from formal specifications
[ ] 3.2.6. Random testing
[C] 3.3. Code-based techniques
[ ] 3.3.1. Control-flow-based criteria
[ ] 3.3.2. Data flow-based criteria
[ ] 3.3.3. Reference models for code-based testing (flowgraph, call graph)
[C] 3.4. Fault-based techniques
[ ] 3.4.1. Error guessing
[ ] 3.4.2. Mutation testing
[C] 3.5. Usage-based techniques
[ ] 3.5.1. Operational profile
[ ] 3.5.2. Software Reliability Engineered Testing
[C] 3.6. Techniques based on the nature of the application
[C] 3.7. Selecting and combining techniques
[ ] 3.7.1. Functional and structural
[ ] 3.7.2. Deterministic vs. random
[ ] 4. Test-related measures
[D] 4.1. Evaluation of the program under test (IEEE982.1-98)
[ ] 4.1.1. Program measurements to aid in planning and designing testing
[ ] 4.1.2. Fault types, classification, and statistics
[ ] 4.1.3. Fault density
[ ] 4.1.4. Life test, reliability evaluation
[ ] 4.1.5. Reliability growth models
[D] 4.2. Evaluation of the tests performed
[ ] 4.2.1. Coverage/thoroughness measures
[ ] 4.2.2. Fault seeding
[ ] 4.2.3. Mutation score
[ ] 4.2.4. Comparison and relative effectiveness of different techniques
[ ] 5. Test Process
[B] 5.1. Practical considerations
[ ] 5.1.1. Attitudes/Egoless programming
[ ] 5.1.2. Test guides

[ ] 5.1.3. Test process management
[ ] 5.1.4. Test documentation and work products
[ ] 5.1.5. Internal vs. independent test team
[ ] 5.1.6. Cost/effort estimation and other process measures
[ ] 5.1.7. Termination
[ ] 5.1.8. Test reuse and test patterns
[D] 5.2. Test Activities
[ ] 5.2.1. Planning
[ ] 5.2.2. Test-case generation
[ ] 5.2.3. Test environment development
[ ] 5.2.4. Execution
[ ] 5.2.5. Test results evaluation
[ ] 5.2.6. Problem reporting/Test log
[ ] 5.2.7. Defect tracking

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Testing ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.8 Chapter 6 - Software Maintenance

Acronyms ... Introduction ... Breakdown Of Topics For Software Maintenance

[ ] 1. Software Maintenance Fundamentals
[B] 1.1. Definitions and Terminology
[B] 1.2. Nature of Maintenance
[B] 1.3. Need for Maintenance
[B] 1.4. Majority of Maintenance Costs
[B] 1.5. Evolution of Software
[C] 1.6. Categories of Maintenance
[ ] 2. Key Issues in Software Maintenance
[ ] 2.1. Technical Issues
[B] 2.1.1. Limited understanding
[C] 2.1.2. Testing
[D] 2.1.3. Impact analysis
[D] 2.1.4. Maintainability
[ ] 2.2. Management Issues
[B] 2.2.1. Alignment with organizational objectives
[B] 2.2.2. Staffing
[B] 2.2.3. Process
[B] 2.2.4. Organizational aspects of maintenance
[??] 2.2.5. Outsourcing
[ ] 2.3. Maintenance Cost Estimation
[C] 2.3.1. Cost estimation

[B] 2.3.2. Parametric models
[C] 2.3.3. Experience
[C] 2.4. Software Maintenance Measurement
[ ] 2.4.1. Specific Measures
[ ] 3. Maintenance Process
[B] 3.1. Maintenance Processes
[ ] 3.2. Maintenance Activities
[C] 3.2.1. Unique activities
[C] 3.2.2. Supporting activities
[??] 3.2.3. Maintenance planning activity
[??] 3.2.4. Software configuration management
[??] 3.2.5. Software quality
[ ] 4. Techniques for Maintenance
[D] 4.1. Program Comprehension
[B] 4.2. Reengineering
[B] 4.3. Reverse engineering

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Maintenance ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.9 Chapter 7 - Software Configuration Management

Acronyms ... Introduction ... Breakdown Of Topics For SCM

[ ] 1. Management of the SCM Process
[B] 1.1. Organizational Context for SCM
[B] 1.2. Constraints and Guidance for the SCM Process
[ ] 1.3. Planning for SCM
[C] 1.3.1. SCM organization and responsibilities
[C] 1.3.2. SCM resources and schedules
[C] 1.3.3. Tool selection and implementation
[B] 1.3.4. Vendor/Subcontractor Control
[B] 1.3.5. Interface control
[B] 1.4. SCM Plan
[ ] 1.5. Surveillance of Software Configuration Management
[C] 1.5.1. SCM measures and measurement
[B] 1.5.2. In-process audits of SCM
[ ] 2. Software Configuration Identification
[ ] 2.1. Identifying Items to Be Controlled
[C] 2.1.1. Software configuration
[C] 2.1.2. Software configuration item
[C] 2.1.3. Software configuration item relationships
[C] 2.1.4. Software version

[C] 2.1.5. Baseline
[C] 2.1.6. Acquiring software configuration items
[B] 2.2. Software Library
[ ] 3. Software Configuration Control
[ ] 3.1. Requesting, Evaluating, and Approving Software Changes
[C] 3.1.1. Software Configuration Control Board
[C] 3.1.2. Software change request process
[C] 3.2. Implementing Software Changes
[B] 3.3. Deviations and Waivers
[ ] 4. Software Configuration Status Accounting
[B] 4.1. Software Configuration Status Information
[C] 4.2. Software Configuration Status Reporting
[ ] 5. Software Configuration Auditing
[B] 5.1. Software Functional Configuration Audit
[B] 5.2. Software Physical Configuration Audit
[B] 5.3. In-process Audits of a Software Baseline
[ ] 6. Software Release Management and Delivery
[C] 6.1. Software Building
[B] 6.2. Software Release Management

Matrix Of Topics Vs. Reference Material ... Recommended References For SCM ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.10 Chapter 8 - Software Engineering Management

Acronyms ... Introduction ... Breakdown Of Topics For Software Engineering Management

[ ] 1. Initiation and Scope Definition
[C] 1.1. Determination and Negotiation of Requirements
[C] 1.2. Feasibility Analysis (Technical, Operational, Financial, Social/Political)
[B] 1.3. Process for the Review and Revision of Requirements
[ ] 2. Software Project Planning
[B] 2.1. Process Planning
[C] 2.2. Determine Deliverables
[C] 2.3. Effort, Schedule, and Cost Estimation
[C] 2.4. Resource Allocation
[C] 2.5. Risk Management
[C] 2.6. Quality Management
[B] 2.7. Plan Management
[ ] 3. Software Project Enactment
[C] 3.1. Implementation of Plans
[B] 3.2. Supplier Contract Management
[C] 3.3. Implementation of measurement process

[D] 3.4. Monitor Process
[C] 3.5. Control Process
[C] 3.6. Reporting
[ ] 4. Review and Evaluation
[C] 4.1. Determining Satisfaction of Requirements
[C] 4.2. Reviewing and Evaluating Performance
[ ] 5. Closure
[C] 5.1. Determining Closure
[C] 5.2. Closure Activities
[ ] 6. Software Engineering Measurement
[B] 6.1. Establish and Sustain Measurement Commitment
[B] 6.2. Plan the Measurement Process
[B] 6.3. Perform the Measurement Process
[B] 6.4. Evaluate Measurement

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Engineering Management ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.11 Chapter 9 - Software Engineering Process

Acronyms ... Introduction ... Breakdown Of Topics For Software Engineering Process

[ ] 1. Process Implementation and Change
[ ] 1.1. Process Infrastructure
[B] 1.1.1. Software Engineering Process Group (SEPG)
[B] 1.1.2. Experience Factory (EF)
[C] 1.2. Software Process Management Cycle
[A] 1.3. Models For Process Implementation And Change
[B] 1.4. Practical Considerations
[ ] 2. Process Definition
[C] 2.1. Software Life Cycle Models
[B] 2.2. Software Life Cycle Processes
[B] 2.3. Notations for Process Definitions
[B] 2.4. Process Adaptation
[B] 2.5. Automation
[ ] 3. Process Assessment
[B] 3.1. Process Assessment Models
[B] 3.2. Process Assessment Methods
[ ] 4. Process and Product Measurement
[C] 4.1. Process Measurement
[C] 4.1.x Software Product Measurement
[C] 4.1.1. Size measurement

[C] 4.1.2. Structure measurement
[C] 4.1.3. Quality measurement
[D] 4.2. Quality Of Measurement Results
[ ] 4.3. Software Information Models
[C] 4.3.1. Model building
[C] 4.3.2. Model implementation
[ ] 4.4. Process Measurement Techniques
[C] 4.4.1. Analytical techniques
[B] 4.4.2. Benchmarking techniques

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Engineering Process ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.12 Chapter 10 - Software Engineering Tools And Methods

Acronyms ... Introduction ... Breakdown Of Topics For Software Engineering Tools And Methods

[ ] 1. Software Engineering Tools
[C] 1.1. Software Requirements Tools
[C] 1.2. Software Design Tools
[C] 1.3. Software Construction Tools
[C] 1.4. Software Testing Tools
[C] 1.5. Software Maintenance Tools
[C] 1.6. Software Configuration Management Tools
[C] 1.7. Software Engineering Management Tools
[C] 1.8. Software Engineering Process Tools
[C] 1.9. Software Quality Tools
[C] 1.10. Miscellaneous Tool Issues
[ ] 2. Software Engineering Methods
[C] 2.1. Heuristic methods
[B] 2.2. Formal Methods
[C] 2.3. Prototyping Methods

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Engineering Tools And Methods ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.13 Chapter 11 - Software Quality

Acronyms ... Introduction ... Breakdown Of Topics For Software Quality

[ ] 1. Software Quality Fundamentals

[D] 1.1. Software Engineering Culture and Ethics
[D] 1.2. Value and Costs of Quality
[D] 1.3. Models and Quality Characteristics
[D] 1.3.1. Software engineering process quality
[D] 1.3.2. Software product quality
[C] 1.4. Quality Improvement
[ ] 2. Software Quality Management Processes
[C] 2.1. Software Quality Assurance
[C] 2.2. Verification & Validation
[ ] 2.3. Reviews and Audits
[C] 2.3.1. Management reviews
[C] 2.3.2. Technical reviews
[C] 2.3.3. Inspections
[C] 2.3.4. Walk-throughs
[B] 2.3.5. Audits
[ ] 3. Practical Considerations
[ ] 3.1. Software Quality Requirements
[B] 3.1.1. Influence factors
[B] 3.1.2. Dependability
[B] 3.1.3. Integrity levels of software
[C] 3.2. Defect Characterization
[ ] 3.3. Software Quality Management Techniques
[C] 3.3.1. Static techniques
[C] 3.3.2. People-intensive techniques
[C] 3.3.3. Analytical techniques
[C] 3.3.4. Dynamic techniques
[C] 3.3.5. testing
[C] 3.4. Software Quality Measurement

Matrix Of Topics Vs. Reference Material ... Recommended References For Software Quality ... Appendix A. List Of Further Readings ... Appendix B. List Of Standards

A.0.14 Chapter 12 - Related Disciplines Of Software Engineering

Introduction
List of Related Disciplines And Their Knowledge Areas
ComputerEngineering ... Computer Science ... Management ... Mathematics ... Project Management ... Quality Management ... Software Ergonomics ... Systems Engineering

A.0.15 Appendix A – Knowledge Area Description Specifications For The Ironman Version Of The Guide To The Software Engineering Body Of Knowledge

Introduction ... Criteria And Requirements For Proposing The Breakdown(S) Of Topics Within A Knowledge Area ... Criteria And Requirements For Describing Topics ... Criteria And Requirements For Selecting Reference Material ... Criteria And Requirements For Identifying Knowledge Areas Of The Related Disciplines ... Common Table Of Contents ... What Do We Mean By "Generally Accepted Knowledge"? ... Length Of Knowledge Area Description ... Role Of Editorial Team ... Important Related Documents (In Alphabetical Order Of First Author) ... Style And Technical Guidelines ... Other Detailed Guidelines ... Editing ... Release Of Copyright

A.0.16 Appendix B – Evolution Of The Guide To The Software Engineering Body Of Knowledge

Introduction ... Stakeholders ... The Evolution Process ... Anticipated Changes

A.0.17 Appendix C – Allocation Of IEEE And Iso Software Engineering Standards To Swebok Knowledge Areas

A.0.18 Appendix D – Classification Of Topics According To Bloom's Taxonomy

Introduction ... Software Requirements ... Software Design ... Software Construction ... Software Testing ... Software Maintenance ... Software Configuration Management ... Software Engineering Management ... Software Engineering Process ... Software Engineering Tools And Methods ... Software Quality

Literatur

[ASIIN06] Fachspezifisch ergänzende Hinweise zur Akkreditierung von Bachelor- und Masterstudiengängen der Informatik (Stand 08. Dezember 2006); ASIIN e.V.; 2006 `http:www.asiin.de/deutsch/download/krit_fa4.pdf`

[Bl56] Bloom, B. (ed.): Taxonomy of Educational Objectives: The Classification of Educational Goals; Mackay; 1956

[ACMCCS1998] Computing Classification System; ACM, http://www.acm.org/class/1998/homepage.html; 2006

[ACMCCS2012] Computing Classification System; ACM, http://www.acm.org/about/class/201 2012

[CCSE03] Joint Task Force on Computing Curricula IEEE Computer Society / Association for Computing Machinery, Computing Curricula Software Engineering Volume Public Draft 1 Computing Curriculum Software Engineering, 2003; http://sites.computer.org/ccse/.

[CsR09] Csonka, Nadine; Raue Cornelia: Analyse akademischer Kompetenzziele - Ergebnisbericht für die Fakultät IV; TU Berlin; 2009

[GSB04] IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK); IEEE, http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf; 2004

