CAPITOLO 1

INTRODUZIONE ALLA GUIDA AL SOFTWARE ENGINEERING BODY OF KNOWLEDGE (SWEBOK, 2004)

Ercole Colonese, Consulente di management e servizi IT, 2011

Questo capitolo, insieme ad altri che seguiranno a breve, è la traduzione libera (in quanto non autorizzata ufficialmente da IEEE) del documento "Guide to the Software Engineering Body of Knowledge (SWEBOK)". Lo scopo è di fornire, ai lettori con una minore padronanza della lingua inglese, un documento in italiano che descriva il modello considerato il riferimento internazionale per la professione dell'Ingegneria del software (pubblicato come standard ISO TR 19759:2005).

A dispetto dei milioni di professionisti del software che operano a livello mondiale e della vastità del software presente nella società moderna, l'ingegneria del software non ha raggiunto ancora lo stato di legittima disciplina e di professione riconosciuta.

Un primo traguardo nel definire una disciplina è di raggiungere il consenso sui contenuti di base della professione (*Core Body of Knowledge*). Operando in questa direzione, il comitato IEEE Computer Society ha definito il corpo delle competenze di base della nuova professione dell'Ingegneria del software. La Guida, scritta sotto l'auspicio del Comitato delle Pratiche Professionali, è parte del progetto realizzato in molti anni per raggiungere tale consenso.

CHE COS'È L'INGEGNERIA DEL SOFTWARE?

La IEEE Computer Society definisce l'ingegneria del software come:

«(1) L'applicazione di un sistematico, disciplinato e quantificabile approccio allo sviluppo, all'operatività e alla manutenzione del software; cioè l'applicazione dell'ingegneria al software.

(2) Lo studio degli approcci definiti al punto precedente»[1].

CHE COS'È UNA PROFESSIONE RICONOSCIUTA?

Affinché l'ingegneria del software sia legittimata come disciplina dell'ingegneria e sia riconosciuta come professione occorre raggiungere il consenso sulle competenze di base (*Core Body of Knowledge*). Questo fatto è ben illustrato da P. Starr quando definisce che cosa può essere considerata una legittima disciplina e una professione riconosciuta. Nel suo libro di successo (vincitore del Premio Pulitzer) sulla storia della professione medica negli USA, egli afferma:

«La legittimazione dell'autorità professionale coinvolge tre distinte affermazioni: primo, che le conoscenze e le competenze dei professionisti siano state validate da una comunità di suoi pari; secondo, che la validazione delle conoscenze sia basata sulla scienza e sulla razionalità; e terzo, che la valutazione professionale sia orientata a un insieme di valori, come il benessere e la salute. Questi aspetti della legittimazione corrispondono a tipi di attributi – collegiali, cognitivi e morali - generalmente inclusi nel termine "professione»[2].

QUALI SONO LE CARATTERISTICHE DI UNA PROFESSIONE?

Gary Ford e Norman Gibbs hanno studiato diverse professioni riconosciute, incluse medicina, legge, ingegneria e economia. [3] Essi hanno concluso che la professione dell'ingegneria è caratterizzata da diverse componenti:

- *Formazione iniziale* eseguita all'interno di un curriculum validato da un'organizzazione apposita tramite l'*accreditamento* (es.: frequenza dei corsi di un piano di formazione definito e superamento degli esami previsti);

[1] "IEEE Standard Glossary of Software Engineering Terminology, 1990."

[2] P. Starr, *The Social Transformation of American Medicine*, Basic Books, 1982, p.15.

[3] G. Ford and E. Gibbs, "*A Mature Profession of Software Engineering*" Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, January 1996.

- Registrazione volontaria dell'aderenza alle pratiche riconosciute tramite il *superamento di esami particolari* (es.: iscrizione all'albo degli ingegneri, certificazione interna ecc.);
- *Sviluppo continuo delle competenze* tramite frequenza di corsi di specializzazione;
- Supporto da parte di una *comunità accreditata*;
- Impegno a seguire un *comportamento etico*.

Questa guida (*Guide to SWEBOK*) contribuisce ai primi tre elementi menzionati. L'articolazione di un *Body of Knowledge* (*BOK*) rappresenta un passo importante verso lo sviluppo di una professione in quanto costituisce un largo consenso verso che cosa s'intende si debba conoscere nell'ingegneria del software. Senza un tale consenso, non si può eseguire una valutazione delle conoscenze possedute, non si può stilare un curriculum da sottoporre alla valutazione, non si può accreditare alcun curriculum. Lo sviluppo del consenso costituisce anche un prerequisito per l'adozione di un programma di formazione e di sviluppo continuo della professione.

QUALI OBIETTIVI SI PONE IL PROGETTO SWEBOK?

La Guida non dovrebbe essere confusa con il Corpo delle conoscenze (*Body of Knowledge*), che esiste di per sé nella letteratura disponibile. Lo scopo della Guida è di descrivere quale porzione del Corpo delle conoscenze è generalmente accettato, di organizzar tale porzione e di fornire un accesso a esso. Informazioni aggiuntive sul significato dato al termine "generalmente accettate" sono fornite in Appendice A.

La *Guida al Software Engineering Body of Knowledge* (SWEBOK) è stata costruita per indirizzare i seguenti cinque obiettivi:

1. Promuovere una vista consistente dell'Ingegneria del software a livello mondiale;
2. Chiarire il campo di applicazione - e definirne i confini - dell'Ingegneria del software rispetto ad altre discipline come, ad esempio, l'Informatica, il Project Management, la Scienza dei computer e la Matematica;
3. Caratterizzare i contenuti della disciplina dell'Ingegneria del software;
4. Fornire un accesso unico al Corpo delle competenze dell'Ingegneria del software;
5. Fornire un fondamento per lo sviluppo del curriculum per la certificazione individuale e la costruzione del materiale da rilasciare.

Il primo obiettivo – fornire una vista internazionale dell'Ingegneria del software – è stato raggiunto grazie al supporto fornito da 500 revisori di 42 nazioni nella prima fase del progetto (1998-2001) e da 120 revisori di 21 paesi nella fase successiva (2003) che ha portato alla pubblicazione dell'attuale versione 2004.

Il secondo obiettivo – chiarire il campo di applicazione e i confini della disciplina – costituisce il motivo della strutturazione della Guida. Il materiale riconosciuto essere appartenente alla disciplina è organizzato nelle prime dieci Aree di conoscenza (Knowledge Area - KA) elencate nella Tabella 1. Ciascuna KA è trattata in un singolo capitolo della Guida.

Nello stabilire i confini della disciplina, è anche importante identificare quali altre discipline condividono tali confini e, spesso, quali competenze specifiche sono condivise.

Tabella 1 Aree di conoscenza (KA) di SWEBOK

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

A tal fine, la Guida riconosce otto discipline correlate, elencate nella Tabella 2 (maggiori dettagli sono forniti nel Capitolo 12 "Discipline correlate all'Ingegneria del software"). Gli Ingegneri del software dovrebbero, quindi, possedere conoscenze su tali materiali. No è, tuttavia, un obiettivo della Guida al SWEBOK caratterizzare la conoscenza delle relative discipline, ma piuttosto quali conoscenze sono viste come specifiche dell'Ingegneria del software.

Tabella 2 Discipline correlate

Cognitive Sciences and Human Factors
Computer Engineering
Computer Science
Management Science
Mathematics
Project Management
Systems Engineering

ORGANIZZAZIONE GERARCHICA

L'organizzazione delle Aree di conoscenza (KA) e i relativi capitoli del documento forniscono il supporto al terzo obiettivo – caratterizzazione dei contenuti dell'Ingegneria del software -.

La Guida utilizza un'organizzazione gerarchica per decomporre ciascuna KA nell'insieme dei contenuti fornendo, a ciascuno di essi, un titolo ben specifico per il

suo riconoscimento. La scomposizione degli argomenti si spinge fino a un secondo o terzo livello per consentire un chiaro riconoscimento di ciascun argomento. La Guida tratta ciascun argomento in maniera il più possibile compatibile con quanto generalmente insegnato nelle scuole o università ed è compatibile con gli standard e la letteratura riguardante l'Ingegneria del software. La scomposizione degli argomenti non pone particolare pressione su specifici domini applicativi, utilizzo da parte degli utenti, filosofie manageriali, metodi di sviluppo, e altri aspetti. La descrizione di ogni argomento è fatta in modo da inquadrare il soggetto nella sua naturale accezione onde permettere al lettore di fare riferimento al materiale disponibile. In sintesi, il Corpo delle conoscenze (Body of Knowledge) non è specifico nella Guida bensì nel materiale cui si fa riferimento.

MATERIALE DI RIFERIMENTO E MATRICE

Per fornire un accesso completo alle conoscenze – il quarto obiettivo – la Guida identifica del materiale di riferimento per ciascuna KA e fornisce, al termine di ogni capitolo, un elenco di libri, articoli o altre sorgenti d'informazioni autorevoli. Ciascun capitolo, inoltre, include una Matrice che mette in relazione gli argomenti trattati e i riferimenti forniti.

La scelta e l'estensione dei riferimenti sono soggettive e non pretendono di essere esaustive.

LIVELLO DI APPROFONDIMENTO

Il livello di approfondimento cui arrivare con la presentazione di un argomento è questione antica e di difficile risoluzione. Il gruppo di lavoro ha adottato un approccio che sia in grado di supportare tutti e cinque gli obiettivi – fornire la base per lo sviluppo di un curriculum, la certificazione e la licenza -. E' stato quindi accettato il criterio delle conoscenze "generalmente accettate" per distinguerle da quelle avanzate e oggetto di ricerca e dalle conoscenze specialistiche. La definizione proviene dal Project Management Institute (PMI):

"*La conoscenza generalmente accettata si applica alla maggior parte dei progetti il maggior numero di volte, e un largo consenso valida il suo valore e la sua efficacia*".[4]

Tuttavia, il termine "generalmente accettate" non implica che una conoscenza indicata debba essere sempre applicata in maniera uniforme in tutti gli ambienti software – ciascun progetto determina quali conoscenze debbano essere presenti nel gruppo di lavoro e come queste debbano essere adottate –; nello stesso tempo non esclude che i professionisti debbano possedere tali competenze. Tali conoscenze devono quindi essere apprese durante il corso di studi seguito dai professionisti del software e dimostrate negli esami; la loro applicazione nei singoli progetti dipenderà dalle necessità e opportunità.

Pratiche specializzate usate solo per alcuni tipi di software	Generalmente accettate Pratiche tradizionali stabilite e raccomandate da molte organizzazioni
	Avanzate e ricerche Pratiche innovative testate e usate solamente da alcune organizzazioni e concetti sviluppati e testati in ambienti di ricerca

Figura 1 Categorie della conoscenza

LIMITAZIONI DOVUTE AL FORMATO DEL LIBRO

Il formato del libro deciso per questa edizione ha dei suoi limiti intrinseci. La natura del contesto trattato richiederebbe una struttura ipertestuale, in cui ogni argomento possa essere collegato ad uno successivo o ad uno precedente.

Alcuni limiti tra aree, sottoaree e argomenti specifici sono spesso arbitrari e vanno presi, quindi, con una certa flessibilità. Quando possibile sono stati forniti puntatori e collegamenti con altri argomenti.

I collegamenti tra le KA non sono da prendere come Input/Output tra le Aree, ma piuttosto come conoscenze da possedere per svolgere le attività specifiche di ciascuna di esse. La scomposizione della disciplina all'interno di ciascuna KA, così come l'ordine con cui le KA sono presentate, non sono da assimilare ad alcun metodo o modello. I metodi sono descritti in una particolare Area della Guida (e la Guida non rappresenta, di per sé, alcun metodo).

AREE DI CONOSCENZA (KNOWLEDGE AREAS)

La Figura 1 fornisce una mappa degli undici capitoli e degli argomenti principali. The prime cinque KA sono presentate nella sequenza tradizionale del ciclo di vita "a cascata" (*Waterfall*). Tuttavia, ciò non significa che la Guida suggerisca tale ciclo di vita o alcun altro ciclo di vita. The altre cinque KA sono invece presentate in ordine alfabetico, mentre le altre discipline sono presentate nell'ultimo capitolo.

[4] *A Guide to the Project Management Body of Knowledge* (PMBOK), 2000 Ed., Project Management Institute, www.pmi.org.

STRUTTURA DELLA DESCRIZIONE DELLE KA

La descrizione delle KA è strutturata come segue.

Nell'introduzione, viene fornita una breve definizione della KA e viene presentata una sintesi del suo ambito e delle relazioni con le altre KA.

La scomposizione degli argomenti costituisce il cuore di ciascuna KA, descrivendo la scomposizione della KA nelle sue sottoaree, argomenti e sottoargomenti. Per ciascun argomento, o sottoargomento, si fornisce una descrizione e uno o più rifermenti.

Il materiale di riferimento è stato scelto in modo da fornire la migliore presentazione di ciascun argomento trattato, pur con le limitazioni di cui si è detto in precedenza.

L'ultima parte della descrizione della KA è costituita da una lista dei riferimenti raccomandati. L'Appendice A di ciascuna KA include suggerimenti per i lettori che volessero approfondire gli argomenti della KA. L'Appendice B presenta la lista degli standard più rilevanti della KA. Le citazioni riportate tra parentesi quadre "[ ]" nel testo identificano riferimenti raccomandati, mentre quelli racchiusi tra parentesi tonde "( )" identificano riferimenti usuali utilizzati per scrivere o giustificare una parte del testo.

SOFTWARE REQUIREMENTS

Un requisito è definito come la proprietà da esibire per dimostrare che si risolve un problema nel mondo reale.

La prima sottoarea di conoscenza è *Software Requirements Fundamentals*. Essa include le definizioni dei requisiti stessi, ma anche della maggior parte di requisiti: di prodotto e di processo, funzionali e non funzionali, proprietà emergenti. La sottoarea descrive anche l'importanza della quantificazione dei requisiti e la differenza tra requisiti di sistema e requisiti software.

La seconda sottoarea di conoscenz*a è Requirements Process*, che introduce il processo che orienta le rimanenti cinque sottoaree e mostra come l'ingegneria dei requisiti si correla con gli altri processi dell'ingegneria del software Esso descrive i modelli di processo, gli attori coinvolti nel processo, il supporto e la gestione del processo, la qualità e il miglioramento del processo.

La terza sottoarea, *Requirements Elicitation*, tratta le sorgenti che possono generare i requisiti e come questi possono essere raccolti dagli ingegneri software. Include le sorgenti dei requisiti e le tecniche per la loro elicitazione.

La quarta sottoarea, *Requirements Analysis*, tratta del processo di analisi dei requisiti per:

- Rilevare e risolvere eventuali conflitti tra i requisiti;
- Scoprire i limiti e i confini del software e come esso interagisca con l'ambiente in cui è destinato operare;
- Elaborare i requisiti di sistema per ricavare i requisiti del software.

L'analisi dei requisiti include le attività relative alla classificazione dei requisiti, modellazione concettuale, progettazione dell'architettura, allocazione dei requisiti e negoziazione dei requisiti.

La quinta sottoarea, *Requirements Specification*, si riferisce alla produzione di un documento, o a una sua registrazione elettronica equivalente (es.: attraverso un tool per la gestione dei requisiti), che possa essere sistematicamente rivisto, valutato e approvato. Per sistemi complessi, particolarmente per quelli che includano componenti anche non-software, si suggerisce la produzione di tre tipi differenti di documenti: System Definition, System Requirements Specification, Software Requirements Specification. La sottoarea descrive tutti e tre i documenti elencati e le attività da svolgere.

La sesta sottoarea, *Requirements Validation*, ha il compito di scoprire i possibili problemi legati ai requisiti prima che si assumano impegni su di essi e si assegnino le risorse. La validazione dei requisiti consiste nella revisione della documentazione dei requisiti per assicurare che il sistema definito sia quello "corretto" (cioè quello "giusto", quello atteso dagli utenti). Essa è suddivisa nella descrizione di come condurre le revisioni dei requisiti, nella prototipazione e nei modelli test di validazione e di accettazione.

La settima ed ultima sotto-area, *Practical Considerations*, descrive gli argomenti pratici ch devono essere presi in considerazione. Il primo argomento da capire è la natura iterativa del processo dei requisiti. I successivi tre argomenti trattano fondamentalmente la gestione delle modifiche e la manutenzione dei requisiti in uno stato che rispecchi accuratamente il software da sviluppare o già sviluppato. Esso include la gestione delle modifiche, gli attributi dei requisiti e il tracciamento dei requisiti. L'ultimo argomento tratta della misurazione dei requisiti.

SOFTWARE DESIGN

Secondo la definizione data da IEEE [IEEE 610.12-90], la progettazione del software è "il processo di definire l'architettura, i componenti, le interface e le altre caratteristiche di un sistema o componente" e "il risultato di tale processo". La KA è divisa in sei sottoaree.

La prima sottoarea, *Software Design Fundamentals*, definisce i concetti basilari e le nozioni relative alla progettazione del software. Si tratta di concetti generali sull'ambito di applicazione della progettazione del software, il processo di progettazione e le relative tecniche.

La seconda sottoarea, *Key Issues in Software Design*, indirizza la concorrenzialità, il controllo e la gestione degli eventi, la gestione degli errori e delle situazioni eccezionali e loro distribuzione, l'interattività dei sistemi e loro persistenza.

La terza sottoarea, *Software Structure and Architecture*, indirizza le problematiche relative alla struttura ed architettura del software, ai diversi punti di vista da cui essa può essere valutata, agli stili e ai disegni (*pattern*) architetturali, alle famiglie di programmi, ambienti e strutture architetturali (*Design Frameworks*).

La quarta sottoarea, *Software Design Quality Analysis and Evaluation*, indirizza gli elementi principali della qualità di una progettazione: attributi della qualità, analisi della qualità, misure e tecniche per la valutazione. Il quadro generale della qualità del software è descritta nell'area specifica "*Software Quality*".

La quinta sottoarea, Software *Design Notations*, indirizza le notazioni utilizzate nella progettazione del software divise in due categorie: descrizioni strutturali e di comportamento.

La sesta e ultima sottoarea descrive *Software Design Strategies and Methods*. Sono descritte le strategie generali e i metodi progettazione orientati alle funzioni (*Function-Oriented*), agli oggetti (*Object-Oriented*), quelli centrati sulle strutture dei dati (*Data-Structure-Centered*), quelli basati sui componenti (Component-Based) e altri.

SOFTWARE CONSTRUCTION

La costruzione del software (*Software Construction*) si riferisce a una serie di attività di dettaglio per la realizzazione del software che vanno dalla codifica al test unitario, al test d'integrazione e debugging. Sono previste tre sottoaree.

La prima sottoarea, *Software Construction Fundamentals*, descrive i tre principi della costruzione del software: Riduzione della complessità, Anticipazione delle modifiche, Strutturazione per la validazione. E' discusso anche l'utilizzo di standard di disegno.

La seconda sottoarea, *Managing Construction*, descrive i modelli di costruzione, la pianificazione della costruzione e la misurazione della costruzione.

La terza sottoarea, *Practical Considerations*, descrive gli elementi della costruzione quali: progettazione della costruzione, linguaggi di costruzione, codifica, test della costruzione, riuso, qualità della costruzione, e integrazione.

SOFTWARE TESTING

Il test del software consiste nella verifica dinamica del comportamento del software realizzato tramite un numero finito di casi di prova selezionati tra un numero pressoché infinito di modalità diverse di utilizzo e a fronte di un comportamento specificato ed atteso. Sono definite cinque sottoaree di competenza.

La prima sottoarea, *Software Testing Fundamentals*, presenta la terminologia del testing, le problematiche principali indirizzate dal testing e le relazioni con le altre attività del ciclo di vita del software.

La seconda sottoarea, *Test Levels*, indirizza i target e gli obiettivi dei diversi test.

La terza sottoarea, *Test Techniques*, descrive due categorie di tecniche. La prima categoria raggruppa le tecniche basate sull'intuizione e l'esperienza dei tester. La seconda categoria raggruppa le tecniche basate sulle specifiche (*Specification-Based*), quelle basate sul codice (*Code-Based*), quelle basate sui possibili errori (*Fault-Based*), quelle basate sull'utilizzo del software (*Usage-Based*) e quelle basate sulla natura dell'applicazione. E' anche discusso come selezionare e combinare le diverse tecniche.

La quarta sottoarea, *Test-Related Measures*, copre la misurazione delle attività di test in termini di valutazione del programma software sottoposto al test e valutazione dei test eseguiti.

L'ultima sottoarea, *Test Process*, fa delle considerazioni pratiche e descrive le attività di test da eseguire.

SOFTWARE MAINTENANCE

La fase di manutenzione del ciclo di vita inizia appena il software è rilasciato in esercizio, sebbene le attività di manutenzione inizino molto prima. Essa indirizza la risoluzione dei problemi rilevati dagli utenti, le modifiche dell'ambiente operativo, i nuovi requisiti. Sono definite quattro sottoaree di competenza.

La prima sottoarea, *Software Maintenance Fundamentals*, fornisce le definizioni e la terminologia, descrive la natura della manutenzione, la necessità di effettuare la manutenzione, i costi principali della manutenzione, l'evoluzione del software e le categorie di manutenzione.

La seconda sottoarea, *Key Issues in Software Maintenance*, indirizza le problematiche tipiche della manutenzione del software. Gli aspetti indirizzati sono quelli tecnici, gestionali, relativi a stime e costi, relative alle misure.

La terza sottoarea, *Maintenance Process*, descrive il processo tipico di manutenzione del software e le attività svolte.

Le quarta ed ultima sottoarea, *Techniques for Maintenance*, indirizza le tecniche relative alla comprensione del software, alla reingegnerizzazione del software e al Reverse Engineering.

SOFTWARE CONFIGURATION MANAGEMENT

Il *Software Configuration Management* (SCM) è la disciplina atta a identificare la configurazione software di un sistema in diversi momenti della sua vita per consentire il controllo sistematico delle modifiche e mantenere l'integrità e la tracciabilità della configurazione stessa. Sono definite sei sottoaree di competenza.

La prima sottoarea, *Management of the SCM Process*, indirizza il contesto organizzativo del SCM, fornisce linee guida per il SCM, definisce le modalità per la pianificazione SCM, per la produzione del piano SCM in particolare e per la sorveglianza dell'implementazione del SCM.

La seconda sottoarea, *Software Configuration Identification*, descrive come identificare gli elementi da controllare, stabilisce gli schemi per l'identificazione degli elementi e le loro versioni, stabilisce le tecniche e gli strumenti da adottare per l'acquisizione ed il controllo degli elementi identificati. Il primo argomento trattato riguarda l'identificazione degli elementi di configurazione da porre sotto controllo e la libreria software.

La terza sottoarea, *Software Configuration Control*, indirizza la gestione delle modifiche durante il ciclo di vita del software. Le attività si articolano su tre fasi successive: prima fase, richiesta, valutazione e approvazione delle modifiche al software; seconda fase, implementazione delle modifiche software; terza fase, deviazioni e rinunce.

La quarta sottoarea, *Software Configuration Status Accounting*, descrive le modalità per la rilevazione delle informazioni relative allo stato della configurazione e la produzione della relativa reportistica.

La quinta sottoarea, *Software Configuration Auditing*, descrive le attività di verifica della configurazione sia funzionale sia fisica del software e la verifica della baseline in corso.

La sesta e ultima sottoarea, *Software Release Management and Delivery*, copre le attività relative alla costruzione del software e alla gestione del suo rilascio in esercizio.

SOFTWARE ENGINEERING MANAGEMENT

L'area Software Engineering Management indirizza la gestione e la misurazione dell'ingegnerizzazione del software. Pur essendo la misurazione un aspetto importante di tutte le KA, le attività specifiche di misurazione sono descritte proprio in questa area di conoscenza. Sono previste sei sottoaree. Le prime cinque coprono le attività tipiche di Software Project Management, mentre la sesta e ultima sottoarea descrive i programmi di misurazione del software.

La prima sottoarea, *Initiation and Scope Definition*, descrive le attività di determinazione e negoziazione dei requisiti, l'analisi di fattibilità e il processo di revisione dei requisiti.

La seconda sottoarea, *Software Project Planning*, indirizza il processo di pianificazione, la determinazione dei deliverable, la stima dell'impegno (effort), della schedulazione e dei costi, l'allocazione delle risorse, la gestione del rischio, la gestione della qualità e la gestione del piano.

La terza sottoarea, *Software Project Enactment*, copre l'implementazione dei piani, la gestione dei contratti con i fornitori, l'implementazione del processo di misurazione, il processo di monitoraggio, di controllo e di reporting.

La quarta sottoarea, *Review and Evaluation*, include la determinazione della soddisfazione dei requisiti e la revisione e valutazione delle prestazioni.

La quinta sottoarea, *Closure*, descrive le attività di chiusura del progetto software.

La sesta e ultima sottoarea, *Software Engineering Measurement*, indirizza, più specificatamente, i programmi di misurazione. In particolare, la descrizione delle misure del prodotto e al processo è riportata nella KA Software Engineering Process. Molte altre misure sono descritte nelle singole KA. Nella presente area si descrive, in particolare, l'impegno nello stabilire e nel sostenere i programmi di misurazione, la pianificazione del processo di misurazione, l'effettuazione delle misure e la valutazione dei risultati delle misure effettuate.

SOFTWARE ENGINEERING PROCESS

L'area di competenza indirizza la definizione, implementazione, misurazione, valutazione, misurazione, gestione, modifica e miglioramento del processo d'ingegneria del software. Sono definite quattro sottoaree di competenza.

La prima sottoarea, *Process Implementation and Change*, presenta i concetti di base, l'infrastruttura del processo, il ciclo di gestione del processo, modelli per l'implementazione e la modifica del processo e considerazioni pratiche.

La seconda sottoarea, *Process Definition*, descrive i modelli del ciclo di vita del software, i processi del ciclo di vita del software, le notazioni per la definizione del processo, le modalità per l'adattamento e l'automazione del processo.

La terza sottoarea, *Process Assessment*, presenta i modelli e i metodi per la valutazione del processo software.

La quarta sottoarea, *Process and Product Measurements*, descrive i concetti generali della misurazione del prodotto e del processo software. Le misurazioni delle specifiche KA sono descritte nelle singole aree; qui si descrivono i concetti generali della misurazione dei prodotti software, dei processi, dei risultati delle misurazioni della qualità, i modelli informativi del software e le tecniche di misurazione dei processi.

SOFTWARE ENGINEERING TOOLS AND METHODS

L'area di competenza indirizza sia i metodi sia gli strumenti a supporto dello sviluppo e manutenzione del software.

La prima sottoarea, *Software Engineering Tools*, utilizza la stessa struttura della presente Guida, con un argomento per ciascuna delle nove aree di competenza. Un argomento aggiuntivo descrive una miscellanea di strumenti come,ad esempio, le tecniche d'integrazione, che sono applicabili a tutte le classi di strumenti.

La seconda sottoarea, *Software Engineering Methods*, indirizza quattro categorie di metodi: metodi euristici relativi ad approcci informali, metodi formali relativi ad approcci matematici, metodi prototipali relativi a tecniche di sviluppo software basate su diverse forme di approcci alla prototipazione.

SOFTWARE QUALITY

Quest'area di competenza descrive le considerazioni generali sulla qualità del software che prescindono dal processo di sviluppo adottato. Poiché la qualità del software ha aspetti diversi, essa è trattata in molte altre aree di competenza. In quest'area si fa riferimento a tali aspetti e alle relative aree di competenza in cui sono approfonditi i vari temi. Sono previste tre sottoaree.

La prima sottoarea, *Software Quality Fundamentals*, descrive i concetti base della qualità del software come la cultura e l'etica, il valore e i costi della qualità, i modelli e le caratteristiche della qualità e il miglioramento della qualità.

La seconda sottoarea, *Software Quality Management Processes*, copre gli argomenti dell'assicurazione della qualità del software, la verifica e la validazione, le revisioni e gli audit.

La terza e ultima sottoarea, *Practical Considerations*, è relativa alla qualità del software. Gli argomenti trattati sono quelli relativi alla qualità dei requisiti, la caratterizzazione dei difetti, le tecniche di gestione della qualità del software e le misure della qualità del software.

DISCIPLINE COLLEGATE ALLA SOFTWARE ENGINEERING

L'ultimo capitolo della Guida è intitolata *Related Disciplines of Software Engineering*. Al fine di circoscrivere l'Ingegneria del software, è necessario identificare le discipline con le quali essa interagisce e condivide un ambito specifico. Il capitolo identifica e riporta, in ordine alfabetico, tali discipline. Per ciascuna di queste discipline condivise, si utilizzano sorgenti riconosciute e basate su di un consenso e si riportano:

- una definizione informativa (quando possibile);
- una lista di Aree di conoscenza (KA).

Le discipline collegate di cui si parla sono elencate qui di seguito.

Tabella 3 Discipline collegate a Software Engineering

• Computer Engineering	• Project Management
• Computer Science	• Quality Management
• Management	• Software Ergonomics
• Mathematics	• System Engineering

APPENDICI

APPENDICE A. SPECIFICHE PER LA DESCRIZIONE DELLE KA

L'appendice descrive le specifiche fornite per i contenuti, i riferimenti raccomandati, il formato, e lo stile della descrizione delle KA.

APPENDICE B. EVOLUZIONE DELLA GUIDA

La seconda appendice descrive la proposta del progetto per l'evoluzione della Guida. La Guida 2004 rappresenta l'attuale edizione di una guida che continuerà a evolversi per indirizzare i bisogni della comunità dell'Ingegneria del software. La pianificazione di tale evoluzione non è completata, ma un tentativo di indirizzo è fornito in questa appendice.

APPENDICE C. ALLOCAZIONE DEGLI STANDARD NELLE KA

La terza appendice contiene una matrice degli standard più rilevanti, principalmente quelli IEEE e ISO, allocati nelle diverse KA della Guida al SWEBOK.

APPENDICE D. VALUTAZIONE DI BLOOM

Come aiuto allo sviluppo del curriculum (ma anche per altri usi) e a supporto del quinto obiettivo della Guida –

Fornire un fondamento per lo sviluppo del curriculum per la certificazione individuale e la costruzione del materiale da rilasciare – quest'appendice fornisce una valutazione di ciascun argomento trattato secondo una delle categorie pedagogiche attribuite a Benjamin Bloom[5]. Il concetto è che gli obiettivi formativi possono essere classificati in una delle sei categorie che rappresentano il livello di crescita: conoscenza, comprensione, applicazione, analisi, sintesi e valutazione. Il risultato di questo esercizio per tutte le KA è riportato nell'appendice D. Tale classificazione non va vista, comunque, come definitiva, bensì come un punto di partenza.

Le prime due figure che seguono rappresentano la scomposizione delle competenze definite dalla Guida al SWEBOK in dieci aree di competenza KA e, ciascuna di esse, in sottoaree.

La terza figura mostra invece l'area di competenza adiacente con relative sottoaree.

[5] Psicologo e pedagogista statunitense (1913 - 1999) è noto per le sue ricerche nel campo delle tassonomie degli obiettivi educativi e dei problemi della valutazione scolastica, con particolare riferimento alla comparazione del profitto in diverse situazioni ambientali, nonché dei metodi di conquista della "padronanza" nell'apprendimento.

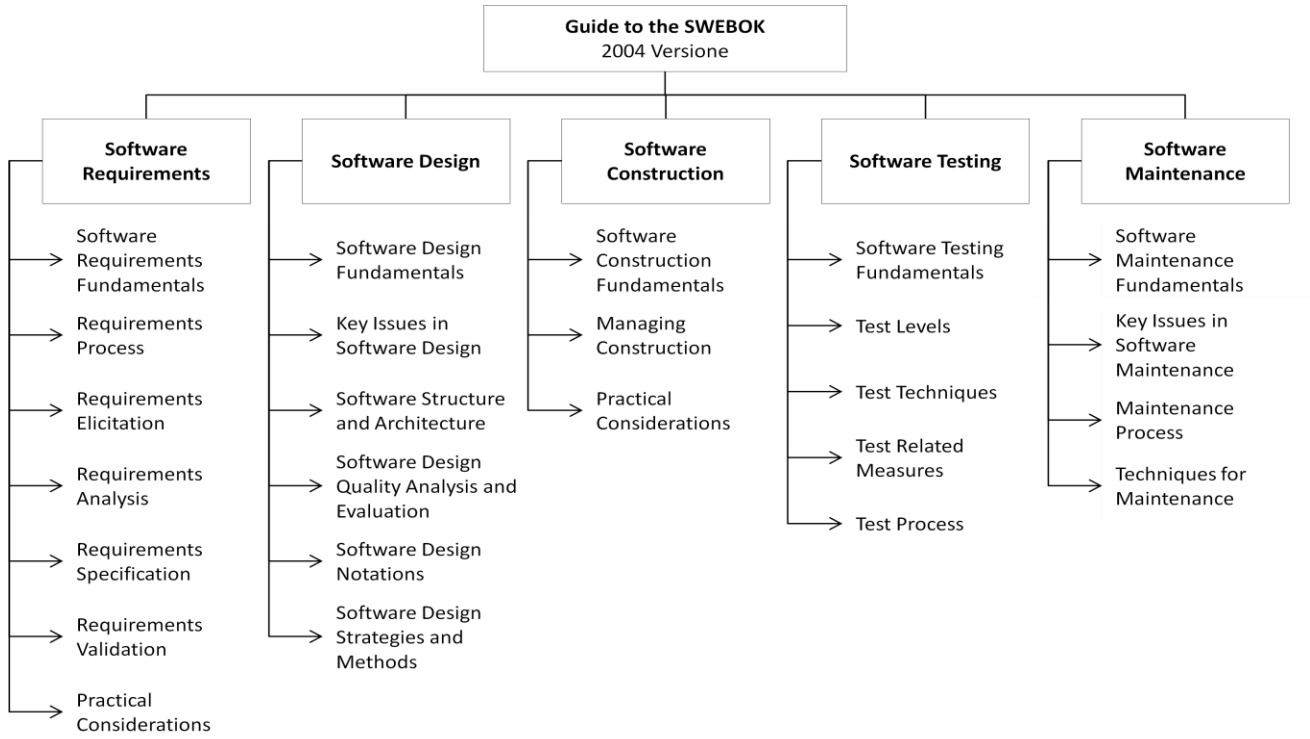

Figura 2 Prime cinque KA

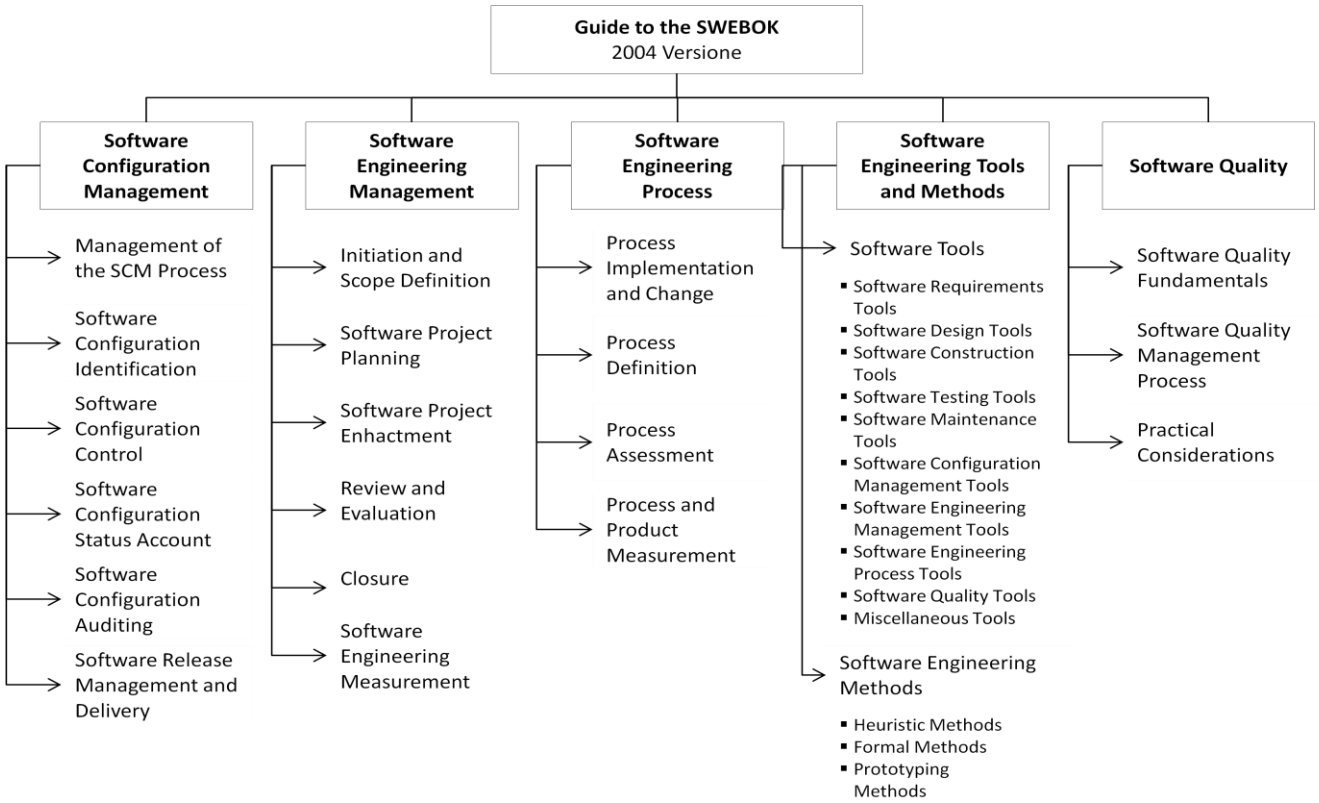

Figura 3 Seconde cinque KA

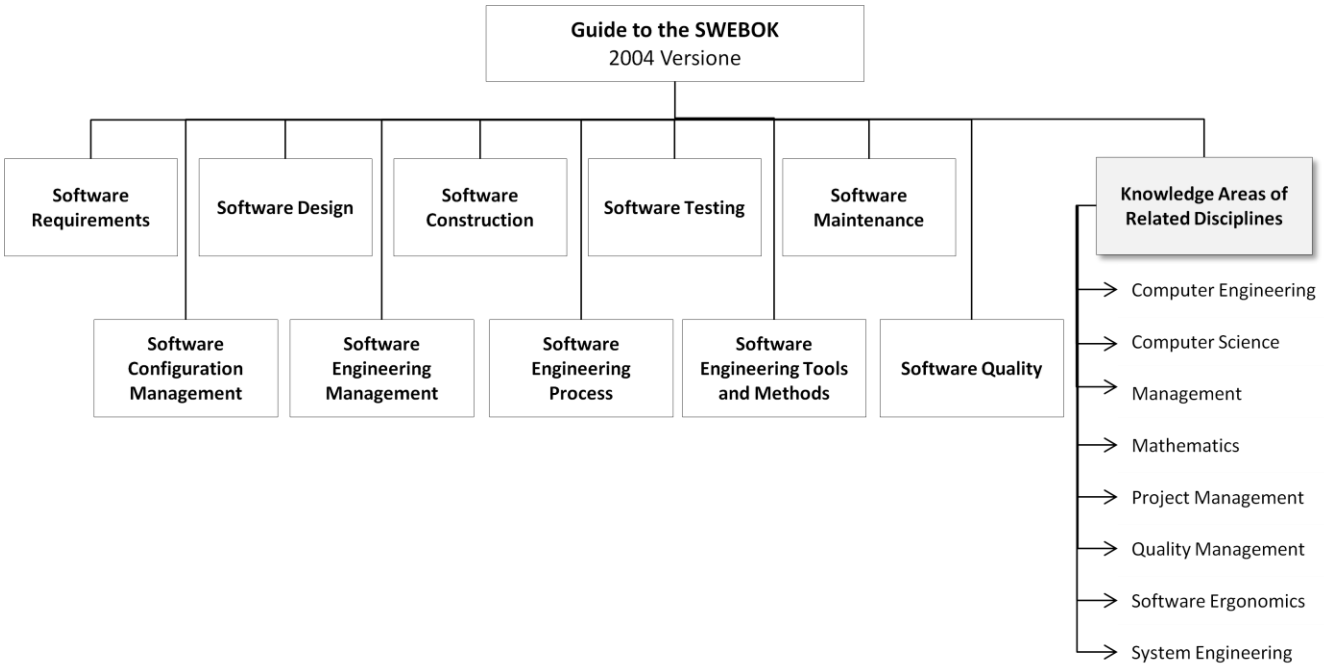

Figura 4 Undicesima KA

BIBLIOGRAFIA

Di seguito è riportata la bibliografia essenziale sui temi generali del Software Engineering. Ulteriori riferimenti saranno forniti nei successivi capitoli dedicati ai singoli argomenti trattati.

[1] *Guide to the Software Engineering Body of Knowledge, SWEBOK, A Project of the Software Engineering Coordinating Committee*. IEEE-Version 2 (2004).

[2] McConnell Steve – *Professional Software Development* – Addison-Wesley (2003).

[3] *Capability Maturity Model Integration* (CMMI), Version 1.1, Staged representation – March 2002 – Software Engineering Institute - Carnegy Mellon University.

[4] Pressman Roger S. - *Software Engineering, A Practitioner's Approach* - McGraw-Hills (2000).

[5] James Martin - *Information Engineering Books: I (introduction), II (planning and analysis), III (design and construction)* - Prentice Hall (1990).

[6] Brooks Frederick .P., Jr - *The Mithical Man-Month: Essays on Software Engineering* – Anniversary Edition Addison-Wesley (1995).

[7] ISO/IEC 9126:2001, *Information technology – Software product evaluation – Quality characteristics and guidelines for their use*.

[8] UNI EN ISO 9001:2000, *Sistema qualità – Modello per l'assicurazione della qualità nella progettazione, sviluppo, fabbricazione, installazione e assistenza – Guida all'uso*.

[9] ISO 9000-3:2004, *Linee guida per l'applicazione di ISO 9001 allo sviluppo, fornitura e manutenzione del software*.

[10] ISO 14598: *Valutazione del prodotto* software.

[11] ISO 12207:1995, *Ciclo di vita del software*.

[12] IEEE *Standard Glossary of Software Engineering Terminology*, 1990.

[13] G. Ford and E. Gibbs, *A Mature Profession of Software Engineering*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, January 1996.

REVISIONI

Il documento è stato sottoposto a revisione interna non formale.