Sono sorpreso e deliziato: dopo vent'anni, questo libro continua a essere molto diffuso. Ne sono state stampate più di 250.000 copie. Spesso le persone mi chiedono quali delle opinioni e delle raccomandazioni proposte nel 1975 siano ancora valide per me, quali siano cambiate e come. Ogni tanto ho affrontato queste domande nelle conferenze, ma da tempo volevo mettere qualcosa per iscritto.

Peter Gordon, ora Publishing Partner di Addison-Wesley, ha lavorato pazientemente con me sin dal 1980 e mi ha proposto di preparare una "Anniversary Edition". Abbiamo deciso di non toccare il testo originale, bensì di ristamparlo immutato (tranne poche correzioni banali) e di aggiungervi le mie idee attuali.

Il Capitolo 16 riproduce un saggio dell'IFIPS del 1986, "No Silver Bullet: Essence and Accidents of Software Engineering", nato dalla mia esperienza come direttore di uno studio del Defense Science Board sul software militare. I coautori di quello studio e il nostro segretari esecutivo, Robert L. Patrick, sono stati preziosi nel riportarmi a contatto con i grandi progetti software del mondo reale. Il saggio è stato ristampato nel 1987 nella rivista *Computer* dell'IEEE, grazie alla quale ha goduto di ampia circolazione.

Quel saggio si è dimostrato provocatorio. Facevo una previsione: che il decennio seguente non avrebbe visto alcuna tecnica di programma che da sola potesse generare un miglioramento di un ordine di grandezza nella produttività del software. Manca solo un anno alla fine del decennio, perciò la mia previsione sembra confermata. Il saggio ha stimolato nella letteratura una quantità di discussioni ancora superiore a quella provocata dal libro. Il Capitolo 17, perciò, contiene commenti su alcune critiche pubblicate e aggiorna le opinioni che avevo presentato nel 1986.

Nel preparare la retrospettiva e l'aggiornamento del libro, sono stato colpito da quanto poche fra le affermazioni che contiene siano

state criticate, dimostrate o confutate da ricerche ed esperienze in corso nell'ingegneria del software. È stato utile per me catalogare queste affermazioni in forma grezza, senza argomentazioni e dati a sostegno. Con la speranza che questi enunciati "nudi" sollecitino argomenti e fatti che possano dimostrare, refutare, aggiornare o perfezionare quelle affermazioni, ho incluso quell'elenco come Capitolo 18.

Il Capitolo 19 è il saggio di aggiornamento stesso. Un'avvertenza per chi legge: le nuove opinioni non sono affatto bene informate dall'esperienza sul campo come quelle del libro originale. In questi anni ho lavorato in un'università, non nell'industria, e su progetti a piccola, non grande scala. Dal 1986, ho solo insegnato ingegneria del software, non ho più condotto ricerche in questo ambito. Le mie ricerche sono state invece sugli ambienti virtuali e le loro applicazioni.

Nel preparare questa retrospettiva, ho cercato di raccogliere le idee correnti di amici che lavorano effettivamente nell'ingegneria del software. Per la loro meravigliosa disponibilità a condividere idee, a commentare saggiamente sulle bozze e a ri-educarmi, sono in debito con Barry Boehm, Ken Brooks, Dick Case, James Coggins, Tom DeMarco, Jim McCarthy, David Parnas, Earl Wheeler ed Edward Yourdon. Fay Ward ha gestito in modo superbo la produzione tecnica dei nuovi capitoli.

Ringrazio Gordon Bell, Bruce Buchanan, Rick Hayes-Roth, i miei colleghi della Defence Science Board Task Force on Military Software e, in modo particolare, David Parnas per le loro intuizioni e le idee stimolanti per il saggio presentato qui come Capitolo 16, e Rekah Bierly per la produzione tecnica dello stesso capitolo. L'analisi del problema del software che fa appello alle categorie di *sostanza* e *accidente* è stata ispirata da Nancy Greenwood Brooks, che ha usato un'analisi di questo genere in un saggio sulla pedagogia violinistica Suzuki.

Le usanze della casa editrice Addison-Wesley non mi hanno permesso di segnalare nell'edizione del 1975 il ruolo fondamentale svolto dal suo staff. Vanno citati in modo particolare i contributi di due persone: Norman Stanton, allora Executive Editor, e Herbert Boes, allora Art Director. Boes ha sviluppato lo stile elegante della prima edizione, che un recensore ha citato espressamente: "margini ampi [e] un uso creativo di tipografia e layout".

Soli Deo gloria.

F.P.B. Jr.
Chapel Hill, N.C., Marzo 1995

Per molti versi, gestire un grande progetto di programmazione è come gestire qualsiasi altra grande impresa, più di quanto creda la maggior parte dei programmatori. Per molti altri aspetti, però, è diverso, più di quanto pensi la maggior parte dei manager di professione.

La tradizione del settore si sta accumulando. Ci sono stati convegni, sessioni ai convegni AFIPS, qualche libro e vari articoli, ma non è ancora affatto nella forma adatta per la trattazione sistematica di un manuale. Sembra giusto, però, offrire questo piccolo libro, che rispecchia sostanzialmente un punto di vista personale.

Anche se inizialmente mi sono formato sul fronte della programmazione, sono stato coinvolto principalmente nell'architettura dell'hardware durante gli anni (1956-1963) nei quali sono stati sviluppati il programma a controllo autonomo e i compilatori per linguaggio di alto livello. Quando nel 1964 sono diventato manager dell'Operating System/360, ho trovato il mondo della programmazione molto cambiato grazie ai passi avanti degli anni precedenti.

Gestire lo sviluppo di OS/360 è stata un'esperienza molto formativa, anche se molto frustrante. Il team, compreso F.M. Trapnell che ha poi preso il mio posto, ha molto di cui essere orgoglioso. Il sistema contiene molte eccellenze, di design e di esecuzione, e ha avuto successo nel raggiungere una grande diffusione. Certe idee, in primo luogo l'input-output indipendente dal dispositivo e la gestione esterna delle librerie, sono state innovazioni tecniche oggi ampiamente copiate. Oggi il sistema è molto affidabile, abbastanza efficiente e molto versatile.

Quell'impresa non si può dire comunque che abbia avuto un successo totale. Ogni utente di OS/360 si rende conto presto di quanto dovrebbe essere migliore. I difetti di progettazione ed esecuzione pervadono in particolare il programma di controllo, in quanto distinto dai compilatori dei linguaggi. La maggior parte di quei difetti risale al periodo di progettazione 1964-65 e pertanto va addebitata al periodo del-

la mia direzione. Inoltre il prodotto è arrivato in ritardo, occupava più memoria del previsto, i costi sono stati di molto superiori a quelli stimati e non ha funzionato molto bene se non varie release dopo la prima.

Lasciata la IBM nel 1965 per Chapel Hill, come stabilito quando avevo accettato l'incarico per OS/360, ho iniziato ad analizzare l'esperienza per capire quali insegnamenti gestionali e tecnici se ne potevano trarre. In particolare, volevo spiegare la grande differenza delle esperienze di management nello sviluppo dell'hardware del System/360 e in quello del software OS/360. Il libro è una risposta (tardiva) alla domanda di Tom Watson sul perché sia difficile gestire la programmazione.

In questa ricerca ho potuto approfittare di lunghe conversazioni con R.P. Case, assistant manager negli anni 1964-65, e F.M. Trapnell, manager dal 1965 al 1968. Ho confrontato le mie conclusioni con altri manager di grandi progetti di programmazione, fra cui F.J. Corbató del MIT, John Harr e V. Vyssotsky dei Bell Telephone Laboratories, Charles Portman di International Computers Limited, A.P. Ershov del Computation Laboratory della divisione siberiana dell'Accademia delle Scienze dell'URSS e A.M. Pietrasanta di IBM.

Le conclusioni si trovano nei saggi che seguono, rivolti a programmatori di professione, manager di professione e in particolare a manager programmatori. Anche se sono saggi separati, c'è un'argomentazione centrale, contenuta in particolare nei Capitoli 2-7. In breve, credo che i grandi progetti di programmazione soffrano di problemi di gestione di natura diversa rispetto a quelli di piccole dimensioni, a causa della divisione del lavoro. Credo che la necessità fondamentale sia il mantenimento dell'integrità concettuale del prodotto. Questi capitoli esplorano sia le difficoltà nell'ottenere questa unità, sia i metodi per ottenerla; i successivi esplorano altri aspetti dell'ingegneria del software.

La letteratura in questo campo non è abbondante, ma è molto dispersa. Per questo ho cercato di fornire riferimenti che possano illuminare punti particolari e al contempo indirizzare il lettore interessato ad altri lavori utili. Molti amici hanno letto il manoscritto e alcuni hanno preparato molti commenti utili; quando mi sembravano preziosi ma non rientravano bene nel flusso del testo, li ho inclusi nelle note.

Dato che questo libro è una raccolta di saggi e non un manuale, tutti i riferimenti bibliografici e le note sono stati relegati a fine volume e consiglio di ignorarli, alla prima lettura.

Sono profondamente debitore a Sara Elizabeth Moore, David Wagner e Rebecca Burris per il loro aiuto nella preparazione del manoscritto, e al professor Joseph C. Sloane per i consigli sulle illustrazioni.

F.P.B. Jr.
Chapel Hill, N.C., Ottobre 1974

Il titolo originale del libro di Frederick P. Brooks Jr. è *The Mythical Man-Month* e fa riferimento a un'unità di misura dell'impegno nella costruzione del software: il mese-uomo. Nella traduzione italiana del testo è stata mantenuta questa unità di misura, per coerenza con l'opera dell'autore. Il titolo è stato invece adattato all'unità più comune in Italia per stimare il lavoro di sviluppo di progetti software, la giornata-uomo.