# IL SOFTWARE FA SCHIFO

Disattenzioni dei programmatori, fretta, voglia di strafare. Così gli utenti continuano a impazzire. E le aziende a fare soldi

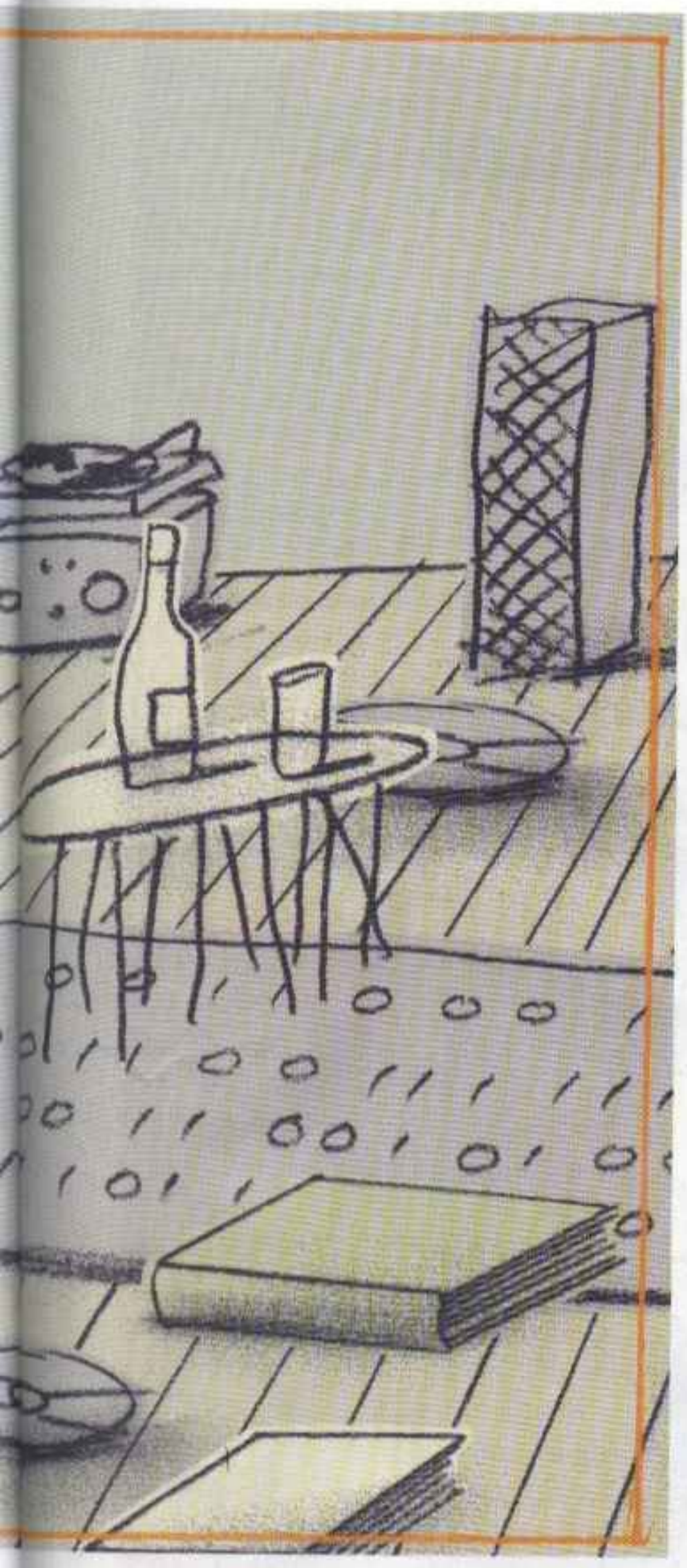
CHARLES C. MANN, TECHNOLOGY REVIEW, STATI UNITI. ILLUSTRAZIONI DI GUIDO SCARABOTTOLO

**È UNA DELLE BARZELLETTE PIÙ VECCHIE DELLA RETE, e gira di continuo da un indirizzo di posta elettronica all'altro. Un pezzo grosso dell'industria del software – di solito Bill Gates, ma a volte un altro – spiega: "Se il settore automobilistico si fosse sviluppato come l'industria informatica, guideremmo automobili da venticinque dollari che fanno cinquecento chilometri con un litro".**

Allora un dirigente dell'industria dell'auto replica: "Sì, e se le auto fossero come i programmi, si bloccherebbero due volte al giorno senza un motivo e l'assistenza direbbe che l'unica soluzione è reinstallare il motore". La battuta riassume uno dei più grandi rompicapo della tecnologia contemporanea. In un tempo sorprendentemente breve il software è diventato cruciale per quasi ogni aspetto della vita moderna. Dai caveau delle banche ai semafori delle città, dalle reti telefoniche ai lettori dvd, dagli airbag delle automobili ai sistemi di controllo del traffico aereo, il mondo intorno a noi è regolato da linee di codice. Eppure gran parte dei programmi è semplicemente inaffidabile.

Chiedetelo a chiunque abbia osservato lo schermo del computer diventare all'improvviso blu, mandando in fumo ore di fatica. Secondo gli ingegneri informatici, troppo spesso il codice è ridondante, confuso, inefficiente e progettato male; e anche quando i programmi funzionano, gli utenti li trovano troppo difficili. Curvi sotto il peso di manuali grossi come mattoni, gli scaffali delle librerie testimoniano la scarsa funzionalità dei programmi.

"Oggi il software è decisamente pessimo. E peggiora in continuazione", dice Watts S. Humphrey, del Software Engineering Institute della Carnegie Mellon University, a Pittsburgh in Pennsylvania, autore di diversi libri sulla qualità dei programmi. Secondo Humphrey, il buon software "è facile da usare, affidabile, privo di difetti, redditizio e durevole. E oggi i programmi non sono nulla di tutto questo. Non sai mai se quello che tiri fuori dalla scatola funzionerà". Secondo Edsger W. Dijkstra, informatico dell'università del Texas a Austin, negli anni l'u-



tente medio di computer "si è talmente abituato ai difetti che si aspetta sempre che il sistema si pianti. Nel mondo vengono distribuiti programmi così pieni di errori che sono una vergogna".

Jim McCarthy è più generoso. Fondatore con la moglie Michele di un istituto di formazione per la qualità del software a Woodinville, nello stato di Washington, McCarthy ritiene che "vale la pena di comprare e usare la maggior parte dei prodotti informatici". Ma riconosce che "solo perché sono molto utili sopportiamo programmi che hanno difetti enormi". A lezione, spesso McCarthy comincia con una presentazione in formato PowerPoint. La prima schermata dice: "Il software fa quasi sempre schifo".

Non è esagerato dire che i problemi del software sono unici. Non si è mai sentito dire da un ingegnere automobilistico che le macchine di oggi hanno gli stessi difetti di dieci o quindici anni fa. Nessun ingegnere aeronautico sostiene che Boeing o Airbus fanno aerei scadenti. E gli ingegneri elettronici non si lamentano di chip e circuiti. Come suggerì nel 1992 lo storico dell'ingegneria Henry Petroski nel libro *The evolution of useful things* (L'evoluzione delle cose utili), la continua messa a punto è la regola abituale nella tecnologia. Gli ingegneri trovano sempre dei punti deboli nei loro progetti e un po' alla volta li correggono; un processo sintetizzato ironicamente da Petroski con la formula: "Al fiasco segue la forma". Ovvero: i prodotti migliorano in continuazione.

**Molti informatici sono convinti che la qualità del software non migliora. Semmai, dicono, è il contrario**

### Modificare

Il software, purtroppo, è un'altra cosa. Da un programma con 45 milioni di linee di codice come Windows Xp, il più recente sistema operativo della Microsoft, ci si aspetta qualche errore. L'informatica d'altronde è una disciplina più giovane dell'ingegneria meccanica o elettrica: i primi veri programmi sono stati creati solo cinquant'anni fa. Ma la cosa assurda è che molti ingegneri informatici sono convinti che la qualità del software non migliora. Semmai, dicono, è il contrario. È come se le auto prodotte nel 2002 fossero meno affidabili di quelle del 1982.

Man mano che aumenta l'importanza del software, saranno più gravi le conseguenze eventuali di un codice scadente, dice Peter G. Neumann, informatico di Sri International, centro privato di ricerca e sviluppo a Menlo Park, in California. Negli ultimi quindici anni a causa di errori di programmazione è saltato il lancio di un satellite europeo, è stata ritardata di un anno l'apertura del costosissimo aeroporto di Denver, una missione della Nasa su Marte è stata cancellata, quattro soldati sono morti per lo schianto di un elicottero, una nave della marina statunitense ha abbattuto un aereo passeggeri e i sistemi per le chiamate delle ambulanze a Londra si sono bloccati causando la morte di almeno trenta persone. E vista la crescente dipendenza dalla rete, spiega Neumann, "stiamo molto peggio di cinque anni fa. I rischi sono aumentati, ma le difese no. Andiamo a ritroso, ed è preoccupante".

Per rispondere alle critiche, alcune società di software stanno modificando le procedure di lavoro. In testa c'è la Microsoft, accusata di vendere prodotti pieni di bachi. Ma i problemi di qualità del software si trascinano da tanto di quel tempo e sembrano così radicati nella cultura informatica che alcuni programmatori stanno cominciando a pensare l'impensabile: per risolvere il problema del software servono, forse, dei buoni avvocati.

La Microsoft ha rilasciato Windows Xp il 25 ottobre 2001. Lo stesso giorno, con una specie di record, l'azienda ha reso disponibile sul suo sito web 18 megabyte di *patch*: correzioni di errori, aggiornamenti per consentire la compatibilità coi vecchi programmi e miglioramenti. Due patch correggevano gravi lacune della sicurezza. O meglio, una lo faceva; l'altra non funzionava. La Microsoft ha avvertito gli utenti (e lo consiglia ancora) di fare una copia dei file più importanti prima di installare le patch. Gli acquirenti della versione per uso domestico di Windows Xp hanno scoperto tuttavia che il sistema non dava modo di ripristinare i file originali se le cose andavano male. Come ha spiegato imperturbabile l'assistenza online della Microsoft, gli speciali dischetti di backup creati da Windows Xp Home "non funzionano con Windows Xp Home".

### Correggere

Sviste del genere, dicono i critici, sono solo errori superficiali, segnali che gli sviluppatori del programma sono stati troppo frettolosi o approssimativi per sistemare difetti ovvi. Secondo R.A. Downes, della società di consulenza informatica Radsoft, i problemi veri stanno nella progettazione di base del software. O meglio nella mancanza di progettazione. Da questo punto di vista, dice Downes, il

**IN RETE**

**AUTOMOBILI MICROSOFT**
http://digital5.ece.tntech.edu/Humor/MoreBillGM.htm
*La versione completa della barzelletta su Bill Gates e la General Motors, e altro umorismo da* nerd

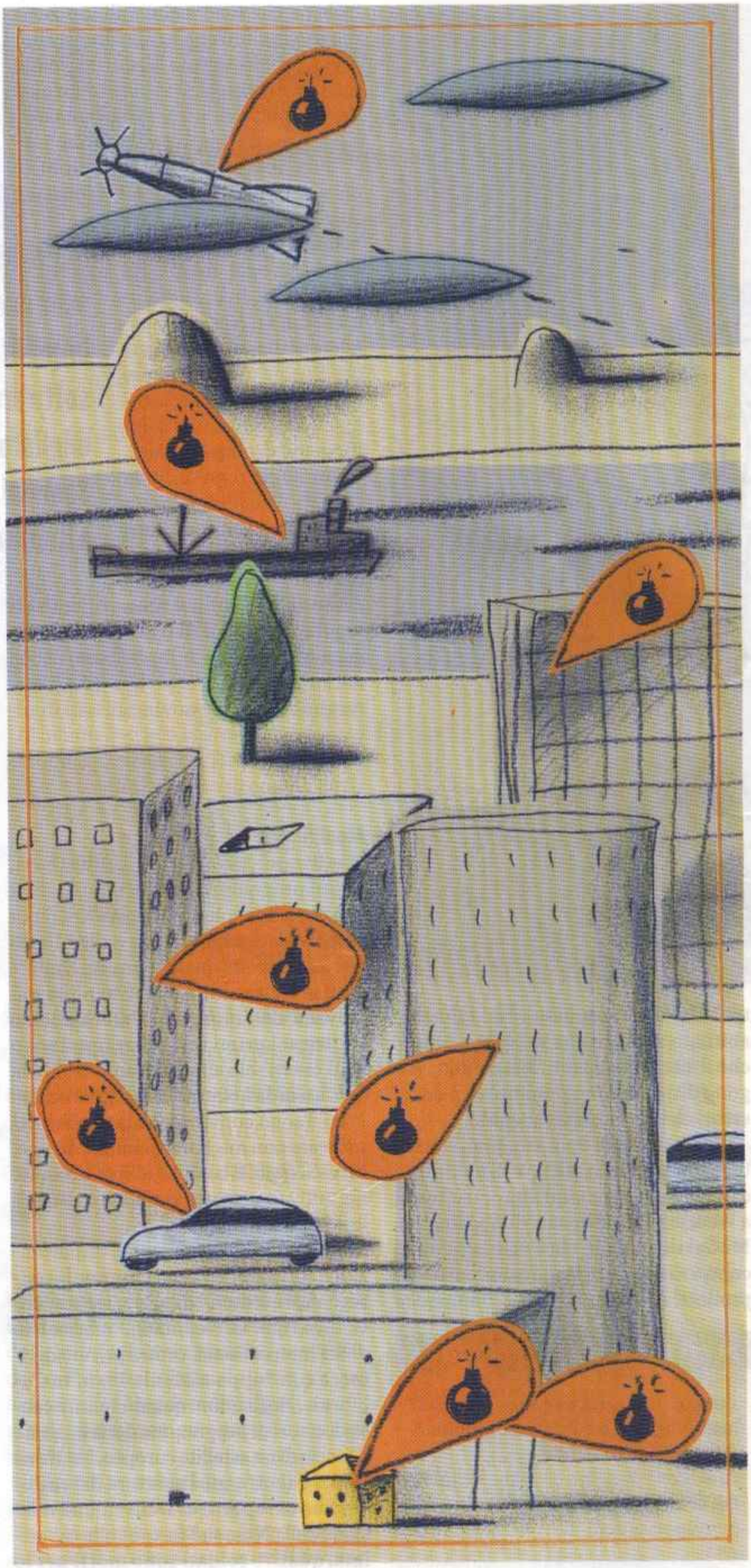
popolare software di programmazione Visual Studio della Microsoft ne è un esempio. Puntando il cursore sulla finestra di Visual Studio, il microprocessore viene bombardato da migliaia di messaggi superflui, anche se il programma non sta facendo niente. "Un disastro...", si lamenta Downes. "È il caos totale". Secondo Dan Wallach, informatico della Rice University di Houston in Texas, il problema non è l'inutile sovraccarico del processore: dopo tutto, osserva, "la capacità di elaborazione costa poco". Né i programmi Microsoft sono particolarmente difettosi; spesso i critici prendono a esempio i suoi prodotti perché sono i più conosciuti e non perché siano peggiori della media. Per Wallach è evidente che le tecniche per scrivere programmi non sono riuscite a tenere il passo dell'aumento esplosivo della loro complessità.

## Compilare

I programmatori scrivono il codice in linguaggi come Java, C e C++ che possono essere letti da esseri umani. Programmi specializzati noti come "compilatori" trasformano questo codice nelle stringhe di uno e di zeri usate dai computer. I compilatori si rifiutano di trasformare il codice che ha problemi banali: piuttosto sputano fuori messaggi di errore. Fino agli anni settanta i compilatori si trovavano in grandi elaboratori prenotati spesso con giorni o settimane di anticipo. Per evitare i ritardi dovuti a eventuali errori, i programmatori – all'epoca per lo più matematici o fisici – facevano le ore piccole a controllare minuziosamente il lavoro. Scrivere programmi era un po' come scrivere articoli scientifici. Rigore, documentazione e controllo accurato da parte dei colleghi erano la norma.

Ma quando i computer hanno cominciato a diffondersi, l'atteggiamento è cambiato. Invece di pianificare meticolosamente il codice, i programmatori restavano svegli a forza di caffè per esaminare i risultati del compilatore. Le raffiche di errori venivano corrette una per una finché il software non era compilato correttamente. "Oggi si pensa di poter lasciare i difetti nel codice e aspettare le correzioni del compilatore", dice Neumann. "Se non saltano fuori messaggi di errore, significa che il programma è a posto".

Col crescere di dimensioni e complessità del software, però, i limiti dell'approccio "codifica e aggiusta" sono diven-



tati evidenti. In media, secondo uno studio pluriennale condotto da Humphrey su 13mila software, i programmatori di professione compiono 100-150 errori ogni mille righe di codice. In base a queste stime, il sistema operativo per le aziende Windows Nt 4, con i suoi 16 milioni di linee di codice, dovrebbe avere circa due milioni di errori.
Nella maggior parte saranno troppo piccoli per avere conseguenze, ma alcuni – molte migliaia – causeranno problemi seri.

Naturalmente la Microsoft ha testato a lungo Windows Nt 4 prima di metterlo sul mercato, ma "in ogni fase di test si scoprono in genere meno della metà dei difetti", dice Humphrey. Se la Microsoft avesse effettuato quattro serie di test – una procedura lunga e costosa – avrebbe scoperto al massimo 15 errori su 16. "Resterebbero comunque cinque errori ogni mille linee di codice", dice Humphrey. "Che è molto poco", ma con ciò il software avrebbe ancora altri 80mila errori.

**In un progetto informatico l'80 per cento del budget è speso per correggere i difetti del software**

## Assemblare

Gli informatici sanno bene che spesso un codice è pieno di lacune e hanno cercato il modo di prevenire i guasti. Per gestire progetti sempre più voluminosi come Windows, per esempio, hanno sviluppato una serie di tecniche tra cui la più nota è la progettazione per componenti. Allo stesso modo in cui per costruire le case si usano materiali standardizzati, così i programmi per componenti sono fatti con elementi modulari intercambiabili. Un esempio è la barra dei menu quasi identica in ogni programma Windows o Macintosh. Secondo Wallach, non solo è una buona prassi tecnica, ma è anche "l'unico modo per far funzionare qualcosa delle dimensioni di Microsoft Office". E proprio la Microsoft, spiega, è stata una delle prime, aggressive fautrici di questo approccio: "L'unica decisione tecnica giusta che abbiano mai preso".

Purtroppo, dicono i critici, spesso le componenti sono messe insieme senza nessun disegno centrale; come se un costruttore cercasse di tirar su una grande struttura senza un progetto. A volte, dice Humphrey, il disegno di grandi progetti informatici "non è altro che un abbozzo sul retro di una busta". Peggio, per motivi di marketing le aziende intrecciano nei nuovi programmi tutti gli elementi che possono, annullando i benefici della costruzione modulare. L'esempio più diffuso è lo stesso Windows. In una seduta del processo antitrust contro la Microsoft, Bill Gates ha ammesso che il sistema operativo non funzionerebbe se gli utenti eliminassero singole componenti come browser, file manager o programmi di posta elettronica. "È un'affermazione incredibile", dice Neumann. "Significa che non c'è struttura, architettura né logica nel modo in cui hanno costruito questi sistemi. Sono assemblati in modo che togliendo una parte qualsiasi tutto va in pezzi".

Il disegno inadeguato dei prodotti finali riflette una progettazione sbagliata nel processo d'ideazione. Secondo uno studio dello Standish Group, società di consulenza di West Yarmouth, nel Massachusetts, i progetti statunitensi di software commerciale sono pianificati e gestiti così male che nel 2000 ne è stato cancellato circa un quarto, senza mai arrivare al prodotto finale. I progetti abbandonati costano alle aziende 67 miliardi di dollari; le revisioni altri 21 miliardi di dollari. Ma poiché il "codifica e aggiusta" richiede fasi di test molto lunghe e costose, anche i progetti vincenti possono rivelarsi inefficienti.

Può sembrare incredibile, ma spesso l'80 per cento del budget per un progetto informatico serve a correggere i suoi stessi difetti, e la cifra non comprende i costi ancora più alti dell'assistenza al prodotto e dello sviluppo di patch per i problemi scoperti dopo il lancio sul mercato. "Quasi la metà del lavoro consiste nel collaudo", dice Humphrey. E anche quando "alla fine le cose funzionano, non c'è ancora nessun disegno complessivo". Perciò non c'è nessuna certezza che gli aggiornamenti non introdurranno nuovi errori.

I rischi di un software scadente furono tristemente evidenti tra il 1985 e il 1987, quando una macchina per la radioterapia computerizzata costruita dalla canadese Atomic Energy, con la sovvenzione del governo, somministrò radiazioni eccessive ad alcuni pazienti nordamericani, uccidendone almeno tre. Dopo un'indagine approfondita, Nancy Leveson, oggi informatica del Massachusetts Institute of Technology (Mit), attribuì gran parte della responsabilità a prassi inadeguate di progettazione del software. Poiché il programma usato per stabilire l'intensità delle radiazioni non era stato progettato né testato attentamente, semplici errori di battitura producevano emissioni letali. Nonostante questa tragica esperienza, macchine simili con software della Multidata Systems International, di Saint Louis, somministrarono radiazioni eccessive a pazienti di Panama nel 2000 e nel 2001, causando altri otto morti.

Una squadra dell'Agenzia internazionale per l'energia atomica (Aiea) attribuì i decessi ai dati inseriti in un modo che i programmatori non avevano previsto. Come osserva Leveson, però, semplici errori d'inserimento dei dati non dovrebbero avere conseguenze letali.

## Perseverare

Gli esperti di programmazione ammettono che disastri simili, purtroppo, sono comuni. Ma alcuni sostengono che non si può giudicare e migliorare il software allo stesso modo di altri prodotti della tecnica. "È un dato di fatto che ci sono cose che altri ingegneri possono fare e noi no", dice Shari Pfleeger, ricercatrice della Rand Corporation di Washington e autrice nel 1998 del libro *Software engineering: theory and practise* (Progettazione software: teoria e pratica). La Pfleeger osserva che se un ponte resiste a un peso di cinquecento chili e a uno di cinquantamila, gli ingegneri possono dedurre che reggerà tutti i valori intermedi. Con i software, dice, "ipotesi del genere non si possono fare. Non si può interpolare".

I produttori di software lavorano inoltre sotto l'assillo di richieste straordinarie. Ford e General Motors fabbricano lo stesso prodotto – una scatola con quattro ruote e un motore a combustione interna – da decenni. Di conseguenza, dice Charles H. Connell, ex capo ingegnere di Lotus Development (oggi inglobata nell'Ibm), hanno potuto migliorare i loro prodotti in maniera costante. Ma alle aziende informatiche viene chiesto ogni volta di creare prodotti mai visti prima: browser all'inizio degli anni novanta, nuove interfacce per i telefonini oggi. "È come se un fabbricante di automobili dicesse: 'Quest'anno invece di un'auto faremo un'astronave'", dice Con-

nell. "Naturalmente avranno dei problemi".

Secondo Nathan Myhrvold, ex massimo responsabile per la tecnologia alla Microsoft, "il dilemma classico nel mondo del software è che la gente vuole sempre più cose". Così il software è sempre "in fase di taglio al vivo", quando i prodotti sono intrinsecamente meno affidabili. Nel 1983, racconta Myhrvold, Microsoft Word aveva solo 27mila linee di codice. "Il problema era che non faceva un granché", e i clienti di oggi non lo accetterebbero. Se la Microsoft non avesse continuato a gonfiare Word con nuove funzioni, il prodotto non esisterebbe più. La sintesi sarcastica di Myhrvold è: "Il software fa schifo perché lo chiedono gli utenti".

**La vendita di software difettosi è la prassi. È per questo che non si compra mai la versione 1.0**

### Rimandare

D'altra parte, sostiene Cem Kaner, informatico e avvocato del Florida Institute of Technology, "per l'industria la qualità è secondaria". Prima di lanciare un prodotto sul mercato le società fanno "riunioni sul rinvio dei bachi" per decidere quali difetti aggiustare subito, quali sistemare più tardi (costringendo i clienti a scaricare patch o a comprare aggiornamenti) e quali ignorare completamente. "Quando le altre industrie trascurano i difetti dei loro prodotti finiscono in tribunale", dice Kaner. "Nel mondo del software ignorare i difetti è la prassi. È per questo che non si compra mai una versione 1.0". I fornitori di software distribuiscono prodotti difettosi, progettati male e con manuali incomprensibili; e poi fanno pagare tariffe costose per le inevitabili richieste d'assistenza. Così, paradossalmente, i metodi di programmazione scadenti fanno guadagnare le aziende.

Quando gli ingegneri di una società informatica scelgono di ignorare un difetto notevole, di solito ci sono un mucchio di recensori, esperti, hacker e altri utenti pronti a rivelarlo. È un fatto positivo. Come ha scritto Petroski in *The evolution of useful things*, "un'opinione pubblica competente e intelligente è il freno migliore alla progettazione lacunosa". Purtroppo le aziende cercano sempre più di scoraggiare le discussioni pubbliche. Le clausole invisibili di molte licenze per il software vietano di pubblicare le prove comparative. Quando nel 1999 il mensile d'informatica Pc Magazine tentò di fare un confronto tra i database di Oracle e Microsoft, Oracle usò i termini della licenza per bloccarlo, anche se la rivista si era impegnata ad assicurare una prova equa e aveva chiesto alle due società di collaborare all'installazione dei programmi. Per acquistare il popolare McAfee VirusScan della Network Associates i clienti devono promettere di non pubblicare recensioni senza il consenso del produttore, una condizione così assurda che a febbraio lo stato di New York ha citato in giudizio la società per aver creato un "patto illegale e restrittivo" che "limita la libertà di parola".

Ma anche per la scuola di pensiero "il software è un'altra cosa" alcune abitudini devono essere riviste. "Non impariamo dai nostri errori", dice la Pfleeger. Nel 1996, per esempio, il razzo francese Ariane 5 esplose quaranta secondi dopo il decollo nel suo viaggio inaugurale. La missione che avrebbe dovuto mettere in orbita un satellite da cinquecento milioni di dollari fu una sconfitta totale. Secondo la commissione d'inchiesta sulla tragedia, l'incidente fu dovuto a "errori sistematici di progettazione del software".

Nella maggior parte dei settori di progettazione, disastri simili innescano riforme profonde in tutta l'industria. È quello che accadrà probabilmente nell'edilizia antincendio dopo il crollo del World Trade Center. Ma nell'industria del software "non c'è un metodo d'indagine definito sulle prove fallite e nessun meccanismo che assicuri la divulgazione delle informazioni". Se ai programmatori francesi fosse stata inculcata, come ad altri ingegneri, la storia della loro disciplina, il fiasco dell'Ariane avrebbe potuto essere evitato.

Comunque sia, alcuni informatici pronosticano un cambiamento nella cultura del software. L'industria è relativamente immune ad azioni penali per la responsabilità sulle caratteristiche dei prodotti (la cosiddetta *product liability*). Per esempio, il virus I Love You si è diffuso in gran parte perché la Microsoft, nonostante gli energici avvertimenti di molti esperti di sicurezza, ha progettato il suo programma di posta elettronica, Outlook, in modo da eseguire facilmente i programmi allegati alle email. Secondo Computer Economics, un gruppo di consulenza di Carlsbad, in California, il costo totale di questa decisione è stato di 8,75 miliardi di dollari. "È sorprendente che non ci sia stata una valanga di cause", dice Wallach.

### Rimborsare

Le società informatiche sono riuscite a evitare vertenze per la responsabilità del prodotto in parte perché le licenze per il software costringono i clienti all'arbitrato, spesso con termini sfavorevoli, e in parte perché le cause sarebbero molto tecniche, e i querelanti dovrebbero pagare esperti costosi per costruire l'accusa. I procedimenti penali, comunque, alla fine arriveranno. E quando i costi delle liti giudiziarie diventeranno abbastanza alti, le aziende avranno un motivo per scrivere codice a prova di bomba. Lo svantaggio di imporre la qualità attraverso vertenze di gruppo, naturalmente, è che cause infondate possono estorcere risarcimenti immeritati. Ma, come dice Wallach, "potrebbe essere semplicemente una cattiva idea di cui è arrivato il momento".

In realtà molti ingegneri informatici pensano che i computer siano diventati così essenziali alla vita quotidiana che nessuno sarà più disposto a concedere l'impunità alle aziende informatiche. "O ci sarà una grande vertenza per la responsabilità del prodotto, o il governo interverrà a regolamentare l'industria", dice Jeffrey Voas, scienziato della Cigital Labs, società per il collaudo di programmi di Dulles, in Virginia. "Qualcuno dovrà cedere. Non sarà carino, ma quando avranno una pistola puntata alla testa, le aziende troveranno il modo di migliorare il software". *nm* ■

**IN LIBRERIA**

*Sul software e la sua industria si può leggere:*

- Mariella Berra e Angelo R. Meo, *Informatica solidale. Storia e prospettive del software libero*, Bollati Boringhieri 2001, 14,46 euro
- Paolo Rocchi, *Cultura e tecnologia del software*, Franco Angeli 2000, 24,79 euro