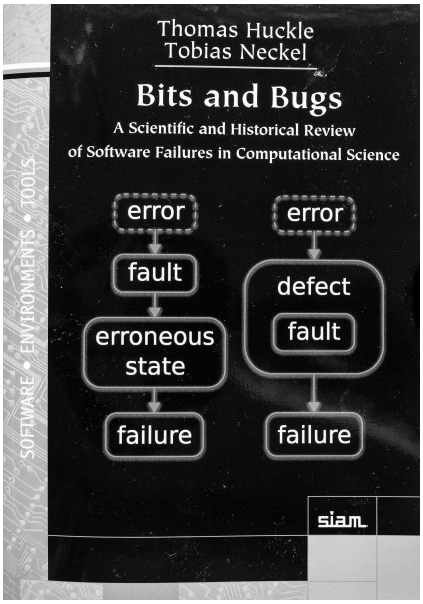
Thomas Huckle in Tobias Neckel, Bits and Bugs: A Scientific and Historical Review of Software Failures in Computational Sciences, SIAM, Philadelphia 2019, 251 str.



Prvi avtor te knjige je profesor na Tehniški univerzi v Münchnu, drugi raziskovalec na isti ustanovi. Prvi avtor vzdržuje spletno stran [1], na kateri zbira poročila o programskih napakah, ki so povzročile nesreče in druge nezaželene dogodke.

Knjiga je namenjena zelo širokemu krogu ljudi, ki jih zanimajo take zgodbe: od strokovnjakov in predavateljev, ki želijo popestriti pouk, do laikov, ki bodo preskočili težje razumljive dele. Vsebuje tudi razlage nekaterih bolj tehničnih stvari. V drugem poglavju tako začne s predstavitvijo celih števil v računalniku, potem nadaljuje z zapisom realnih števil, plavajočo vejico, pretvorbo med raznimi formati zapisa in na koncu z ravnanjem v izjemnih primerih. Če recimo pride do deljenja z 0, se sistem ne sme sesuti. (Avtor te recenzije se spomni, kako so jih pred pol stoletja pri predmetu *Računski praktikum* posvarili, naj ne poskušajo deliti z 0 na dragih švedskih elektromehaničnih računskih strojih. Ko je nekdo vseeno naredil tako napako, se je težki stroj »zaciklal«, se hitro vrtel brez prestanka in po nekaj minutah se je iz njega začelo kaditi.) Dobro je, če je sistem kar se da zaščiten pred takimi nepričakovanimi stanji.

Vse to je potrebno za razlago **nesreče prve rakete Ariane 5** (let 501) leta 1996. Raketa, ki bi v orbito morala prenesti štiri satelite, je slabo minuto po izstrelitvi zavila iz smeri in eksplodirala. Škoda je znašala okrog 500 milijonov dolarjev.

Preiskavo je vodil znani francoski matematik Jacques-Louis Lions. Knjiga dobro analizira serijo napak, ki so povzročile katastrofo. Če nekoliko poenostavimo, so uporabili programe šibkejše rakete Ariane 4. Podatki inercialnega sistema o položaju in vodoravnem gibanju rakete so bili 64-bitni in nato pretvorjeni v 16-bitna cela predznačena števila. Pri tem je prišlo do prekoračitve obsega, kar je povzročilo ustavitev in nato ponovni zagon inercialnega sistema. Ponovni zagon se je začel s testnimi podatki, ki niti približno niso odražali dejanskega stanja. Interni računalnik rakete pa

je to interpretiral kot trenutno stanje in ukazal močan zasuk pogonskih šob. Sile, ki so ob tem nastale, so prekinile povezavo osnovne rakete in pomožnih raket. Čeprav je bil inercialni sistem podvojen, je do iste napake prišlo v obeh sistemih in tako ta rezerva ni prav nič pomagala. Ob vsakem takem podrobno analiziranem primeru knjiga na kratko navede še vrsto podobnih.

Naslednji je na vrsti problem **prehoda v novo tisočletje**. Letnica je bila do takrat podana le z zadnjima dvema številkama. Zato je bilo treba pravočasno spremeniti številne programe in letnice podajati v celoti ali vsaj s tremi številkami. Prehod je začuda potekal brez katastrof velikega obsega. Vseeno je seznam zapletov dolg – od smešnih, ko so v Veliki Britaniji tisoče dojenčkov uvrstili med stoletnike, do tragičnih, ko so v Sheffieldu zaradi napačno izračunanega rizika za Downov sindrom izvedli dva splava. Nekaj zapletov so kot običajno povzročili slabo izvedeni programski popravki, ki naj bi odpravili problem.

Vpliv **zaokrožitvenih napak** je predstavljen z zgledom Vancouverske borze v Kanadi. Računalniški sistem je računal borzni indeks kot uteženo povprečje cene okrog 1500 delnic. Pravzaprav je seštel cene teh delnic in jih pomnožil s faktorjem w, ki je bil izbran tako, da je bila začetna vrednost enaka 1000. Po 22 mesecih je indeks padel na 524,811, torej na dobro polovico začetne vrednosti. Zdi se neverjetno, da naj bi šele takrat opazili, da je nekaj močno narobe. Knjiga tega paradoksa ne razloži. Vendar naj bi bil po Wikipediji [2] sloves te borze izredno slab (bila naj bi polna delnic ničvrednih rudnikov). Poleg tega je bil to eden izmed prvih primerov borznih indeksov in tako verjetno številni teh vrednosti tako in tako niso jemali resno.

Borza je končno poklicala zunanje strokovnjake. Tem je stvar hitro postala jasna. Računalniški program je računal na štiri decimalke in rezultate skrajšal na tri decimalke z rezanjem zadnje decimalke. Če je bila četrta decimalka 0, napake zaradi rezanja ni bilo. V povprečju pa se je rezultat zmanjšal za $\epsilon = 0{,}00045$ glede na pravo vrednost. Indeks je program posodabljal s podatki o spremembah cen delnic. Vsakič, ko se je cena neke delnice spremenila, je k staremu indeksu prištel razliko cen, pomnoženo z w, in odrezal četrto decimalko. To se je zgodilo približno 2800-krat na dan, kar je v povprečju dnevno zmanjšalo pravo vrednost za približno $2800 \times 0,00045 = 1{,}26$. Dvaindvajset mesecev je 440 delovnih dni ... Ko so novembra 1983 indeks znova izračunali »z ničelne točke«, je vrednost čez vikend poskočila s 524,811 na 1098,892.

Simulacije so pokazale, da bi bila ob normalnem zaokroževanju napaka tudi po tako dolgem času zanemarljiva, saj takrat zaokrožamo navzgor in navzdol približno enako pogosto in se napake izničijo.

Bolj zapleten je neuspeh protiraketnega sistema Patriot v Zalivski vojni leta 1991. Tu je šlo med drugim za slabo posodobitev precej starega sistema: nekateri podatki o isti spremenljivki so bili 24-bitni, drugi po novem 48-bitni.

Posledično naj bi prišlo do kopičenja zaokrožitvenih napak. Sistem je bil prvotno zamišljen za prestrezanje letal in ne raket.

Izgubo vrtalne ploščadi *Sleipner A* leta 1991 knjiga začne s kratko predstavitvijo *Metode končnih elementov*. Velikansko plavajočo ploščad iz votlih valjastih železobetonskih delov so sestavili v enem od norveških fjordov. Ko so jo začeli testno potapljati, je na stiku treh valjev popustila ena od sten. Prostor med valji je namreč načrtovano napolnila voda, ki je bila v globini pod tlakom približno 7 barov, konstrukcija pa tega (nenačrtovano) ni zdržala. Ploščad je v 18 minutah za vedno izginila v globinah in ob udarcu v dno povzročila potres tretje stopnje po Richterju. Neposredne škode je bilo za 180–250 milijonov dolarjev, posredne za 700–1000 milijonov USD. Knjiga podrobno opiše vrsto problematičnih ravnanj, ki so vodila do nesreče.

V Severnem morju je bilo že pred tem postavljenih več takih ploščadi. Gradbeno podjetje je bilo sicer isto kot pri nekaterih prejšnjih ploščadih, a noben od takratnih inženirjev ni sodeloval pri novem projektu. Morda je bil vzrok menjava lastnika: namesto inženirjev so zdaj podjetje vodili poslovneži. Varnostni faktorji za ploščadi niso tako visoki kot recimo za mostove. Struktura mora plavati, da jo lahko odvlečejo do kraja, kjer bo pritrjena na morsko dno, zato morajo biti stene dovolj tanke.

Za analizo napetosti so uporabili serijo programov. (Pri prejšnjih projektih so precej te analize opravila specializirana podjetja.) Prvi program je narisal mrežo celic za metodo končnih elementov, žal ne ravno dobro. Mreža je bila tudi groba. Za nekatera kritična mesta so uporabili kvadratno ekstrapolacijo, kar se je izkazalo kot neposrečeno. Tako so močno podcenili napetosti v kritičnih točkah. Za armaturo so zaradi varčevanja ponekod uporabili preostalo železje iz starih projektov, ki ni segalo dovolj globoko v beton. Če primerjamo sliko novega tipa ojačitve s preizkušenim iz prejšnjih projektov, je razlika velikanska in očitna vsakomur z malo mehaničnega znanja. Nesreča vsaj ni zahtevala žrtev in je prispevala k zmanjšanju takih napak. Kot pravi poročilo o nezgodi (str. 72 knjige):

> Verjetno največja lekcija iz študija tega primera je, da računalniške analize nikdar ne smemo obravnavati kot »proces v črni škatli«. Računalniška analiza je dobra le toliko, kot je dober uporabnik, ki vstavlja podatke in interpretira rezultate. Pravo modeliranje in interpretacija zahtevata resnično razumevanje teoretičnega in praktičnega delovanja programa in polno razumevanje pomena rezultatov. Zmeraj moramo uporabiti racionalne metode preverjanja rezultatov. Metode zagotavljanja kakovosti morajo zagotoviti čas za pregled takih podrobnosti.

Od leta 1978 je NASA s satelitom Nimbus med drugim spremljala koncentracijo ozona v atmosferi. Meritve niso kazale bistvenih sprememb. Leta

1985 pa je britanska odprava s tal na Antarktiki izmerila 40-odstotno zmanjšanje ozonskega plašča. Kako je bilo mogoče, da Nasini instrumenti tega niso zaznali? Izkazalo se je, da je bil satelit programiran tako, da ni upošteval podatkov, ki so bili daleč od pričakovanih vrednosti. To je večkrat uporabljana metoda v statistiki: ignoriramo rezultate, ki zelo odstopajo od povprečja ali pričakovanja. Ker je bila koncentracija ozona veliko manjša od pričakovane, tega satelit enostavno ni poročal.

Leta 1999 nemški meteorologi niso napovedali katastrofalnega neurja Lothar. Spet je bilo za to zaslužnih več faktorjev. Vremenski balon, ki so ga spustili na kanadski obali, je eksplodiral. Zato so dve uri po nesreči spustili novega. Podatke novega balona pa so Nemci vnesli s časom, kot da bi šlo za originalni balon. Očitno pa se je atmosfera v vmesnem času precej spremenila. Podatke so tudi sicer vnašali le za vsakih šest ur. Majhne spremembe v začetnih podatkih lahko hitro privedejo do velikih razlik, ker so vremenski modeli slabo pogojeni. Meteorološke službe, ki so ignorirale novi balon, so imele boljše napovedi. Danes se vremenski modeli posodabljajo v krajših časovnih intervalih.

Poglavje **Sinhronizacija in časovni načrti** podrobneje obravnava problem odpovedi prvega poleta vesoljskega plovila Columbia leta 1981. Plovilo je bilo opremljeno s petimi računalniki. Štirje so imeli naložen isti program. Če bi eden izpadel, bi preostali trije še zmeraj lahko delovali po principu večine: se pravi, če bi se vsaj dva računalnika »strinjala«, bi to bila izbrana odločitev. Peti računalnik pa je imel naložen drugačen operacijski sistem in drugačen program, ki bi lahko prevzel upravljanje, če bi se izkazalo, da je v programski opremi preostalih štirih računalnikov napaka. Teoretično je to zelo dobro premišljeno. Vendar pa je moral peti računalnik spremljati dogajanje in deloma tudi rezultate delovanja preostalih štirih računalnikov, da bi lahko takoj reagiral. Sistema sta imela različen način časovnega delovanja. Zapleteno sinhronizacijo so rahlo pokvarili naknadni ne dovolj premišljeni popravki in tako se je start ponesrečil. Teoretično večja zanesljivost je zaradi dodane kompleksnosti privedla do odpovedi sistema. Podrobnosti so verjetno razumljive predvsem strokovnjakom s tega področja. Zanimivo je, da je bila verjetnost, da pride do napake, le 1 : 67, tako da tudi večkratno testiranje ne bi nujno odkrilo problema.

Profesor Thomas Nicely je junija 1994 pri raziskavah v teoriji števil seštevaL inverzne vrednosti praštevil z računalnikom. Rezultati pa so bili včasih napačni. Na koncu je ugotovil, da je nekaj narobe z Intelovim procesorjem Pentium. Ta je imel vgrajen nov, hitrejši algoritem za deljenje. Obvestil je Intelovo servisno službo, ki mu je pričakovano poslala vzvišen odgovor, da s procesorjem ni nič narobe in da ga bodo poklicali. (Kasneje se je izkazalo, da je napako odkril že mesec prej Tom Kraljevic, ki je študiral na univerzi Purdue in obenem delal za Intel. Vendar je informacija ostala nekje v podjetju.) Po šestih dneh čakanja je Nicely obvestil nekaj prijateljev in prišlo je

do objave na spletu. Navedeno je bilo več primerov, ko je izračun inverzne vrednosti števila dal ne prav točen rezultat. Več neodvisnih strokovnjakov je podrobneje raziskalo vzrok napak in verjetnost, da bo do njih prišlo. Res je, da so bili taki primeri zelo redki. Tako je po [3] Pentium izračunal, da je 5506153 deljeno z 294911 enako 18,66990... namesto 18,670558...

Po objavi je Nicely z Intelom sklenil dogovor o molku, ki pa je bil prepozen. Novica se je razširila in krožile so šale kot: »Kaj pomeni nalepka *Intel inside*? To je varnostno opozorilo!« ali »Intel inside = vsebuje tudi napako«. Že konec leta je Intel po takem in drugačnem izogibanju oznanil, da je pripravljen zamenjati vse defektne procesorje. Strošek: pol milijarde dolarjev. Knjiga podrobno razlaga, kje in kako je bil algoritem deljenja pomanjkljiv. To bodo laže razumeli tisti, ki imajo nekaj izkušenj na tem področju. K razumevanju lahko pripomore tudi razlaga [3].

V poglavju o **kompleksnosti** se knjiga ukvarja z nesrečami medicinske naprave Therac-25 za obsevanje tumorjev. Podjetje Atomic Energy of Canada Limited jo je sestavilo iz francoskega linearnega pospeševalnika elektronov (5-25 MeV) in druge opreme, vključno z računalnikom. Imela je dva načina delovanja: obsevanje z elektroni za površinske dele tkiva in obsevanje z rentgenskimi žarki (zavorno sevanje) za globlje predele. Operater se je včasih zatipkal in vnesel napačen način delovanja. Ko je to na hitro poskušal popraviti, je bilo videti, da je sistem zablokiral. Zato je postopek večkrat ponovil. V resnici je vsakič prišlo do nekontroliranega izredno močnega sevanja in posledično več smrti pacientov. Vse to se je dogajalo v letih od 1985 do 1987 v ZDA in Kanadi. Zgodbe so prav grozljive. Tako je eden od pacientov začutil pravi udarec v hrbet. Ker se je zavedel, da je nekaj narobe, je vstal z mize in poskušal pobegniti, pa ga je nova masivna doza zadela v roko.

Vzrok so sprva iskali v napakah stikal in druge električne opreme. Po knjigi naj bi bilo takrat zaupanje v računalniške programe izredno veliko. Šele kasneje se je izkazalo, da so bili krivi programska oprema, slabosti v dokumentaciji in navodilih za operaterje ter odsotnost varnostnih mehanizmov. Programska oprema je bila delo enega samega programerja.

Knjiga kratko omenja še veliko hujšo katastrofo v letih od 2005 do 2009. Po oceni ameriške Food and Drug Administration je pomanjkljiva programska oprema infuzijskih črpalk povzročila okrog 700 smrti in skoraj 20000 poškodb.

Naslednji podrobno obdelan primer je polomija avtomatiziranega sistema upravljanja prtljage na mednarodnem letališču v Denverju. Na javni razpis so se od 16 kontaktiranih podjetij javila le tri, ki pa se niso hotela obvezati, da prvi tak velik sistem na svetu naredijo v nekaj letih. Tako je v začetku leta 1992 delo dobilo projektantsko podjetje, ki ga je privlekla letalska družba United. To podjetje v resnici ni imelo izkušenj s sistemi, ki morajo delovati sproti. Imelo pa je izredno velikopotezne načrte in je

uporabilo revolucionarne novosti na številnih področjih. Vozički za prtljago naj bi potovali z veliko hitrostjo, se polnili in praznili kar med upočasnjeno vožnjo; uporabljali naj bi prepoznavanje vozičkov s čipi RFID; dovoljena je bila prtljaga, večja od standardnih mer. Prvič naj bi sistem upravljala mreža računalnikov namesto centralnega računalnika. Po zakasnitvah so spomladi 1994 imeli veliko otvoritev, ki se je sprevrgla v popolno polomijo. Kovčki in torbe so padali s tekočih trakov, se odpirali, obleka je letela po zraku, vozički so se zaletavali ... Programerji se niso zavedali, da je optimizacija takega transporta NP-zahteven problem. Iskanje optimuma je računsko prezahtevno; zadovoljiti se je treba s kolikor toliko dobrimi rešitvami, kar pa zahteva veliko preizkušanja in testiranja. Tehničnih novosti je bilo prav tako veliko preveč. Tudi inženirske rešitve so bile problematične. Tirnice transportnih linij so imele preostre zavoje, hitrosti so bile prevelike, tako da je zračni upor razmetaval prtljago. Vse skupaj so morali opustiti in škoda je bila velikanska.

Desetletje kasneje so manj zapletene podobne sisteme uresničili v Evropi. Na letališču Heathrow se je pri otvoritvi sistema za upravljanje prtljage tudi zapletlo, a so po kakem mesecu težave odpravili.

Knjiga dokazuje, da je računalništvo nekoliko različno od matematike. Ob primerih eksponentne rasti števila operacij namreč na strani 198 navaja:

n	2^n	e^n
100	$1{,}26765\ldots \times 10^{30}$	$2{,}688\ldots \times 10^{43}$
1000	$1{,}0715\ldots \times 10^{301}$	Inf

Prava vrednost spodaj desno je $1{,}970\ldots \times 10^{434}$, kar pa presega omejitve pri zapisu velikih števil v računalniku v dvojni natančnosti.

To je le nekaj primerov iz obsežne zbirke zgodb v knjigi. Kot rečeno, je veliko pripovedi zanimivih tudi za nepoznavalce in dobro napisanih. Nekatere razlage so dostopne in informativne, druge za matematika nič novega, tretje zahtevne in bolj suhoparne. Knjiga seveda opozarja, da lahko poznavalci številne razlage preskočijo.

Na koncu knjige imamo še razdelek **Urbane legende in druge zgodbe** ter nekaj programov v Matlabu, ki ilustrirajo obravnavane primere.

LITERATURA

[1] T. Huckle, *Collection of Software Bugs*, dostopno na `www5.in.tum.de/persons/huckle/bugse.html`, ogled 13. 5. 2020.

[2] *Vancouver Stock Exchange*, dostopno na `en.wikipedia.org/wiki/Vancouver_Stock_Exchange`, ogled 13. 5. 2020.

[3] D. W. Deley, *The Pentium division Flaw*, 1995, dostopno na `daviddeley.com/pentbug/pentbug4.htm`, ogled 13. 5. 2020.

Peter Legiša