ČÁST III: Chyby – Proč software zabíjí (a jak tomu AI (ne)pomůže)

> *„První princip je, že sám sebe nesmíte oklamat – a sami sebe oklamáte nejsnáze.“ — Richard Feynman*

V programování a systémech neexistuje „skoro správně“. Malá chyba na začátku se v komplexním systému neztratí – ona se v něm rozroste. Tomuto jevu říkáme entropie.

— Anatomie katastrof

1. Therac-25: Race Condition (1985-1987)

Lineární urychlovač pro léčbu rakoviny. Kvůli chybě v kódu (souběh procesů) dostali pacienti 100x vyšší dávku radiace, než bylo určeno. Software „předpokládal“, že operátor zadá data pomalu.

Lekce: Software, který neumí pracovat s nepředvídatelným chováním (včetně rychlosti člověka), je nebezpečný.

2. Patriot Missile: Drift (1991)

Během války v Zálivu selhal systém Patriot při zachycení rakety Scud. Příčina? Systém byl zapnutý příliš dlouho. Vnitřní hodiny driftovaly o 0,000000095 sekundy každou sekundu. Po 100 hodinách to dělalo 0,34 sekundy – dost na to, aby raketa minula cíl o půl kilometru.

Lekce: Malé, neviditelné chyby se v čase sčítají (compound error).

3. Ariane 5: Overflow (1996)

Nejdražší „chyba v typu“ v historii. Rocket se pokusila narvat 64-bitové číslo s plovoucí čárkou do 16-bitového celého čísla. Došlo k přetečení (overflow), software se zhroutil a raketa se sama odpálila 37 sekund po startu.

Lekce: Předpoklad, že „data budou vždy v určitém rozsahu“, je cesta do pekel.

— Entropie a komplexita

Ludwig Boltzmann definoval entropii jako $S = k \ln W$. V softwaru je W počet možných stavů systému. Čím je systém komplexnější, tím více stavů (včetně těch katastrofických) může

nastat. AI nám pomáhá kód psát rychleji, ale zároveň zvyšuje W – tedy i riziko, že vytvoříme systém, kterému už nikdo nerozumí a který nelze plně otestovat.

— Druhý druh chyby: když selže úsudek, ne kód

Therac-25, Patriot i Ariane spadají do jedné kategorie: **kód udělal něco jiného, než jsme chtěli.** Existuje ale zákeřnější druh chyby — kód (nebo statistika) udělá přesně to, co jsme zadali, výpočet je *správně*, a přesto je závěr **fatálně špatný.** Protože jsme se ptali na špatně vymezených datech. A na rozdíl od pádu rakety tahle chyba nevydá žádný zvuk.

Paradox nízké porodní váhy (Yerushalmy, 1971)

Jacob Yerushalmy, statistik z Kalifornské univerzity v Berkeley, v roce 1971 publikoval analýzu kouření matek, porodní váhy a kojenecké úmrtnosti. Data ukázala dvě věci, které dávaly smysl, a jednu, která neměla dávat:

1. Děti kuřaček se rodí **lehčí** a častěji spadají do kategorie „nízká porodní váha".
2. Děti s nízkou porodní váhou umírají v prvním měsíci **22× častěji** než děti s normální váhou.
3. Ale **uvnitř skupiny dětí s nízkou porodní váhou** měly děti kuřaček úmrtnost **2× nižší** než děti nekuřaček.

Závěr, který se nabízel: u ohrožených, malých dětí kouření matky jako by **chránilo**. Nesmysl, který zní jako data. A měl reálné následky — podle retrospektivy v *International Journal of Epidemiology* (2014) tento výsledek „o dekádu pozdržel protikuřácká opatření u těhotných" v USA a „o několik let odložil jakoukoli kampaň na změnu kuřáckých návyků matek" ve Velké Británii.

Kde byla chyba? Nikoli ve výpočtu — ten byl správně. Chyba byla v tom, že jsme **podmínili na následku** (porodní váze). Nízkou porodní váhu způsobuje kouření, ale i mnohem horší věci (vrozené vady, vážné komplikace). Když se díváme jen na malé děti, srovnáváme „malé kvůli cigaretě" s „malé kvůli něčemu mnohem nebezpečnějšímu" — a kuřačky najednou vypadají líp. Statistici tomu říkají **collider bias** (zkreslení podmíněním na společném následku); je to stejná past jako Berksonův a Simpsonův paradox. Kouření je pro dítě škodlivé, ať má nízkou porodní váhu, nebo ne.

> *Příklad rozebral Allen Downey v knize Probably Overthinking It (kap. 7) a v přednášce Talks at Google — viz demo Berksonova paradoxu.*

Most k AI: tohle je přesně ten typ chyby, který vibe coding zhmotní v průmyslovém měřítku. LLM vám s naprostou jistotou napíše dotaz, který agreguje data po špatně zvolené

skupině, a výsledek bude vypadat čistě, přesvědčivě a publikovatelně. Korektní výpočet nad špatně vymezenou otázkou je nebezpečnější než pád rakety — protože pád je vidět hned, kdežto tohle se dostane do tisíce článků dřív, než to někdo zpochybní.

Pedagogický průvodce pro lektora

Cíl této části: Vyvolat respekt k preciznosti a pochopení rizik slepé důvěry v AI kód.

1. **Analýza kódu:** Ukažte jednoduchý kód s chybou „overflow“ nebo „drift“. Nechte účastníky hádat, co se stane po milionu iterací.
2. **Debata o odpovědnosti:** Kdo mohl za Ariane 5? Programátor? Tester? Manažer? (Odpověď: Systémová chyba v procesech validace).
3. **Feynmanův test:** Nechte účastníky vysvětlit princip „driftu“ tak, aby to pochopil někdo, kdo si jen seřizuje hodinky.
4. **Cvičení „Red Flags“:** Dejte účastníkům kód vygenerovaný AI a nechte je hledat „tiché zabijáky“ (nepošetřené okrajové stavy, typové nekonzistence).
