XX

Co nezabije systém, to ho posílí?

Po této kapitole přestaneš snít o systému bez chyb a začneš stavět systém, který chybu přežije — a z každé rány vyjde silnější. Naučíš se zakódovat jednotky do typů tak, aby past nešla spustit; poznáš žebřík křehké → robustní → antifragilní; a chaosem, který si předepíšeš sám, zjistíš dřív než tvůj zákazník, co ve tvém systému doopravdy drží.

Vítr zhasne svíčku, ale rozdmýchá oheň. Totéž platí o náhodě, nejistotě, chaosu: chceš je využívat, ne se před nimi schovávat. … Buď oheň a přej si vítr.

— Nassim Nicholas Taleb, Antifragile: Things That Gain from Disorder, *prolog „How to Love the Wind“, Random House, 2012*

Kluzák jménem Boeing 767

Třiadvacátého července 1983 startuje z Montrealu do Edmontonu let Air Canada 143 — zbrusu nový Boeing 767, jeden z prvních v metrických jednotkách. Ve výšce jedenačtyřiceti tisíc stop, přes dvanáct kilometrů nad Manitobou, se ozve varování o tlaku paliva. Posádka ho ještě stačí brát jako závadu čerpadla; za chvíli zhasne levý motor. A o pár minut později, když už kapitán klesá k nejbližšímu letišti, dodýchá i pravý. V pilotní kabině dopravního letadla nastane ticho, které tam nemá co dělat. Sedmašedesát cestujících a dva piloti letí ve stotunovém kluzáku bez motorů — a bez většiny přístrojů, protože ty jelo na tomtéž proudu, který právě zmizel.

Jak se stroj tak dokonale vyprázdnil? Nikdo ho nezapomněl natankovat. Právě naopak — palivo se počítalo pečlivě, jen ve špatných jednotkách. Palivoměr byl porouchaný, tak posádka i pozemní technici dopočítali litry na kilogramy ručně. Jenže do výpočtu vsypali koeficient 1,77 — hustotu paliva v *librách* na litr —, místo metrických 0,8 kilogramu na litr. Kanada zrovna přecházela na metrickou soustavu a nový Boeing byl jednou nohou v každém světě. Výsledkem bylo, že do nádrží nateklo

zhruba o polovinu paliva méně, než papír tvrdil. A tady je jádro, kvůli němuž ten příběh vyprávíme: chybu neudělal jeden nešika. Udělal ji *systém*. Porouchaný přístroj, rozpůlená soustava jednotek, ruční přepočet bez nezávislé kontroly, únava, spěch — každý článek zvlášť byl přežitelný; teprve dohromady poslaly letadlo do vzduchu poloprázdné.

Kapitán Bob Pearson měl jednu kvalifikaci navíc, kterou letový řád nevyžaduje: byl zkušený plachtař. Uměl to, co dopravní piloti nedělají nikdy — číst stroj bez tahu, hospodařit s výškou jako s posledními penězi, klouzat. Navigoval s prvním důstojníkem Mauricem Quintalem na opuštěnou vojenskou základnu Gimli, o níž Quintal věděl z dob služby. Jenže Gimli už dávno nebylo letiště. Byl to závodní areál a zrovna toho dne se tam konal rodinný den motoristů — na dráze motokáry, kolem ní stany, karavany a děti. Na tohle asfaltové hřiště Pearson stotunový kluzák posadil. Praskl přední podvozek, zajiskřilo se, letoun se zastavil pár metrů před lidmi. Sedmašedesát cestujících vyšlo ven po svých. Nikdo nezemřel. Boeing dostal přezdívku, kterou nesl až do vyřazení: *Gimli Glider*, kluzák z Gimli.

Prameny: vyšetřovací zpráva (Board of Inquiry, soudce George Lockwood, 1985); Air Canada let 143, 23. 7. 1983, Boeing 767. První motor zhasl ve 41 000 ft (12,5 km), druhý o několik minut později při klesání (35 000 ft, 10,7 km). Přepočtový faktor 1,77 lb/l místo 0,8 kg/l. Piloti: kpt. Robert Pearson (plachtař), první důstojník Maurice Quintal. 69 lidí na palubě, žádná oběť.

Gimli byla bývalá základna RCAF přestavěná na Gimli Motorsports Park; 23. 7. 1983 tam probíhal „Family Day" místního automobilového klubu — proto ty motokáry, karavany a rodiny přímo u dráhy.

Zastav se u toho, co příběh *není*. Není to historka o hrdinovi, který zachránil letadlo — i když Pearson hrdina byl. A není to historka o troubovi, který popletl jednotky — protože žádný jediný trouba tam nebyl. Je to příběh o tom, že vážná porucha skoro nikdy nemá jednu příčinu. Má jich řetěz: každá dílčí chyba by sama o sobě neublížila, protože by ji zachytila jiná pojistka — jenže pojistky selhaly jedna po druhé, a mezera propustila stroj až na zem. Tomu se v inženýrství říká *řetěz nehody* a je to nejdůležitější věc, kterou si z Gimli odneseš: nehledej viníka, hledej články.

→ kap. 19: Therac-25 byl týž vzorec — souběžná chyba i přetečení čítače zvlášť přežitelné, teprve chybějící hardwarová pojistka je pustila na pacienta. Řetěz, ne jeden viník.

A protože jeden příběh svede vypadat jako náhoda, přidám druhý — o šestnáct let mladší a o čtyřista milionů dolarů dražší. Třiadvacátého září 1999 ztratila NASA spojení se sondou Mars Climate Orbiter přesně ve chvíli, kdy měla zabrzdit do oběžné dráhy kolem Marsu. Sonda vlétla do atmosféry příliš nízko a shořela. Vyšetřování našlo přesně tutéž třídu chyby jako v Gimli: jeden software počítal impulz motorků v *librosekundách* (americká soustavová jednotka), druhý ho četl jako by šlo o *newtonsekundy* — a mezi nimi je faktor 4,45. Navigace tak systematicky podceňovala účinek zážehů a sonda se řítila o osmnáct kilometrů níž, než měla. A pozor na pointu, na níž trvala i sama NASA: vinu nesvalila na dodavatele, který jednotky popletl, ale na *sebe* — protože chyběl proces, který by nesoulad odhalil. Zase žádný jediný viník. Zase děravý řetěz.

Pramen: Mars Climate Orbiter Mishap Investigation Board, Phase I Report (NASA/JPL, 10. 11. 1999). Chyba v pozemním souboru „Small Forces": librosekundy vs. newtonsekundy, faktor 4,45. Sonda ztracena 23. 9. 1999, cena mise ≈ 327 mil. USD. Board výslovně: příčinou nebyl jeden člověk, ale absence nezávislé kontroly jednotek.

Dvakrát tatáž chyba, v odstupu šestnácti let, v odvětvích posedlých bezpečností jako žádná jiná. To ti něco říká o cíli téhle kapitoly. Kdyby stačilo „být opatrný“ a „nezaměňovat jednotky“, letectví ani NASA by druhou katastrofu nezažily — byli opatrní až běda. Cíl proto *není* nedělat chyby. Chyby uděláš, tvůj tým je udělá, tvůj asistent je vyrobí po tuctech (kapitola šestnáct). Cíl je postavit systém, který jednu chybu *přežije* — a v tom nejlepším případě z ní vyjde silnější. O tom je celá tahle kapitola a začneme, jak jinak, u sebe.

Turbína, která se nevešla

Slíbil jsem v úvodu knihy pátou jizvu a měří se v palcích. Tady je — otevřená; zavřu ji, spolu s Gimli, až v příští kapitole o rozhraních.

> Zákazník v Turecku si objednal turbínu a my ji vyrobili přesně podle výkresu. Přesně — na tisícinu. Problém byl, že „přesně“ znamenalo na každé straně něco jiného. Náš konstrukční tým počítal a kreslil v centimetrech, jak se u nás sluší; zákazníkovo zadání i jeho základ na místě byly v *palcích*, jak se sluší tam. Nikde v celém řetězu nebyl jediný kus papíru, který by ty dva světy postavil vedle sebe a řekl: pozor, tady se převádí. Turbína dojela přes půl Evropy, jeřáb ji zvedl nad základ — a ona se nevešla. O kus, který byl přesně tak velký, jak velký je rozdíl mezi palcem a centimetrem, natažený přes celý stroj. Stálo to nový termín, nové náklady na dopravu a den, na který nezapomenu, protože jsem ho prostál u telefonu a vysvětloval, jak je možné, že jsme vyrobili *správně* a *nefunguje to*. Byli jsme přesní. Nebyli jsme antifragilní. Nikdo z nás nechyboval o víc než o převodní faktor — a přesně jako v Gimli to dohromady stačilo. Jednotka té bolesti jsou palce. Zůstala mi.

Všimni si, že moje turbína je ta samá chyba jako Gimli a jako Mars: dva světy jednotek, žádný most mezi nimi, a v mezeře propadne stroj. Kdybych tehdy uměl to, co bude v této kapitole ve věži — zakódovat jednotku přímo do čísla tak, aby palec a centimetr nešly beztrestně sečíst —, past by se nedala ani spustit. Ale k tomu se dostaneme. Nejdřív potřebuješ jazyk, ve kterém se o téhle věci vůbec dá přemýšlet. Ten jazyk mají tři slova a nejstarší z nich je řecké.

DEMO

`lesk-a-bida-vibecodingu.gaussalgo.com/d/jednotkova-past`

Jednotková past naživo: přepočítej palivo Boeingu a impulz sondy sám — jednou s jednotkou v čísle, jednou bez ní. S holými čísly ti past projde a stroj spadne; s jednotkou zabudovanou do typu se sčítání palce a centimetru odmítne provést. Zkus obojí a uvidíš, kde přesně Gimli, Mars i turbína vznikly.

Damoklés, Fénix a Hydra

Nassim Taleb, autor našeho epigrafu, si vypůjčil trojici obrazů, které rozdělují všechny systémy na světě podle jediné otázky: *co s tebou udělá rána?*

TEENAGER · MECHANISMUS

Tři odezvy na náraz — zapamatuj si je jako tři postavy, protože se ti budou vracet do konce knihy:

```
KŘEHKÉ  ·  Damoklés
   Rána ŠKODÍ. Meč nad hlavou visí na vlásku;
   stačí průvan a je konec. Cíl křehkého
   systému: aby rána vůbec nepřišla.

ROBUSTNÍ  ·  Fénix
   Rána NEVADÍ. Uhoří a vstane přesně stejný
   jako předtím — ani slabší, ani silnější.

ANTIFRAGILNÍ  ·  Hydra
   Rána POSILUJE. Usekneš hlavu, narostou dvě.
   Systém z každého nárazu vyjde odolnější.
```

Většina lidí míří na robustnost — „ať to vydrží“ — a to je dobrý cíl. Ale nejsilnější systémy jdou o krok dál: jsou nastavené tak, že je náraz *zlepší*. Ne magií; mechanismem, který si za chvíli ukážeme přesně.

Damoklés: dvořan, nad nímž Dionýsios pověsil meč na koňské žíni, aby ukázal cenu moci. Fénix: pták, který shoří a vstane z popela. Hydra: netvor, jemuž místo useknuté hlavy narostou dvě. Taleb tuhle mytologii bere úmyslně — antifragilita není „hodně robustní“, je to jiná kategorie: robustní se ráně brání, antifragilní ji vítá.

Tady je nutné zabrzdit, protože „antifragilita“ je slovo, které v sobě nese celý sud marketingové vaty. Nabízí se říct „buď jako Hydra!“ a jet dál — a to bych ti prodal reklamu, ne řemeslo. Antifragilita není nálada ani nálepka. Je to *měřitelná vlastnost tvaru*, jak uvidíš ve věži, a bez toho tvaru je to prázdné slovo. Proto se teď rozdělíme: kdo chce obraz, ten už ho má v Hydře; kdo chce vědět, *co přesně* antifragilita znamená v číslech, jde se mnou o dvě patra výš, kde ji definujeme jako konvexitu výplaty. Nejdřív ale to, co s tím uděláš zítra ráno u klávesnice.

Řemeslo: obrana do hloubky a předepsaný chaos

Mezi „chci systém, který přežije chybu“ a hotovým systémem leží hrst konkrétních zvyků. Žádný z nich není nový a žádný není chytrý; jsou to staré známé pojistky, které ti letecký i kosmický průmysl platí krví. Dají se shrnout do jedné věty: *nespoléhej na jednu vrstvu.*

TEENAGER · MECHANISMUS

Obrana do hloubky — pět vrstev, každá chytá, co propustila předchozí. Kdyby kterákoli jediná v Gimli fungovala, letadlo dolétne.

```
1. VALIDACE NA HRANICI
   Na vstupu ověř, že data dávají smysl, a
   špatná odmítni hned: záporné palivo? počet
   nad kapacitu? → STOP, ne „nějak to
dopočítáme".

2. JEDNOTKY V TYPECH
   Nedrž „vzdálenost = 42". Drž „= 42 cm".
   Typ hlídá, že palce a centimetry nejde sečíst
   → past se nedá ani spustit (viz věž).

3. ASSERTY UVNITŘ
   Kde „tohle nikdy nenastane", napiš, že to
   nikdy nenastane. Když přece → chceš to vědět
TEĎ.

4. FAIL FAST vs. GRACEFUL DEGRADATION
   Spadni hlasitě tam, kde chyba škodí (motor,
   platba) — nebo se degraduj s grácií tam, kde
   půl služby > žádná (stránka bez doporučení).

5. RETRY S BACKOFFEM
   Síť škytla? Zkus znovu, ale ne hned a ne
pořád:
   čekej 1 s, 2 s, 4 s, 8 s… (exponenciální
backoff),
   ať zotavující se službu nedorazíš pokusy.
```

Zastavím se u dvou řádků, protože je lidé pletou. *Fail fast* a *graceful degradation* nejsou soupeři; jsou to dva nástroje pro dvě situace. Když chyba znamená škodu — účet naskočí dvakrát, motor dostane špatný povel —, chceš *spadnout hned a nahlas*, protože tiché pokračování je horší než zastavení. To je Therac z minulé kapitoly: stroj místo zastavení blikl „Malfunction 54“ a jel dál. Ale když chyba znamená jen *míň služby* — spadl modul s doporučeními, výpadek jedné ze sta funkcí —, chceš se *degradovat s grácií*: ukázat uživateli stránku bez doporučení, ne bílou obrazovku. Řemeslo je poznat, ve které situaci zrovna jsi. Motor v Gimli

klienti zkusí přesně v téže vteřině a znovu vyrobí špičku.

měl selhat hlasitě (a selhal); přístroje na palubě se měly degradovat s grácií (a nedegradovaly — zhasly s proudem, a to byla další slabina řetězu).

A pak je tu poslední zvyk, který nepatří ke kódu, ale k lidem kolem něj, a je z celé pětky nejdůležitější: **postmortem bez hledání viníka**. Když systém spadne, sejdeš se s ostatními a popíšeš, *jak* se to stalo — celý řetěz článků —, ale nehledáš, *kdo* to udělal. Ne z měkkosrdcatosti. Z čistě sobeckého inženýrství: v okamžiku, kdy začneš hledat viníka, všichni ostatní přestanou mluvit. A ty tím oslepneš přesně vůči těm článkům řetězu, které nutně musíš vidět, abys je příště zavřel. Gimli ani Mars nedopadly katastrofou proto, že by tam byli špatní lidé; dopadly tak proto, že *systém* neměl vrstvu, která chybu zachytí. Kdo hledá viníka, opravuje člověka. Kdo hledá článek, opravuje systém — a jen ten druhý opravdu opravuje.

→ kap. 9: omyl je zaplacená informace, ne ostuda. Postmortem je ta faktura přečtená nahlas — a přečíst ji lze jen tam, kde se za omyl netrestá. Blame-free postmortem není laskavost, je to podmínka učení.

Předepiš si vlastní chaos

Teď to nejsilnější celé dílny — a přímo pokračování „předepsané gravitace“ z kapitoly šesté. Tam jsme si předpisovali odpor, aby neatrofovalo tělo a úsudek. Tady předepíšeme odpor *systému*, aby neatrofovala jeho odolnost. Protože platí nepříjemná pravda: vrstvu obrany, kterou jsi nikdy neviděl zabrat, ve skutečnosti nemáš. Máš jen *víru*, že ji máš — a to je Therac.

TEENAGER · MECHANISMUS

Chaosový protokol pro jednotlivce — schválně rozbíjej, co „určitě nespadne“, a dívej se, co přežije. Nepotřebuješ Netflix; stačí terminál a odvaha:

```
chaosový_protokol(můj_systém):
  vypni síť       # běží offline, nebo zamrzne?
  zabij proces    # nastartuje se, nebo je konec?
  zaplň disk      # pozná to, nebo tiše ztrácí
data?
  podstrč špatný  # „palivo = -500", „30. února":
    vstup         #   odmítne to, nebo počítá
dál?
  zdrž závislost  # služba mlčí 30 s: čeká věčně,
                  #   nebo se po limitu vzdá s
grácií?

  → co spadlo CELÉ = křehké místo nalezené
LEVNĚ,
    dřív než ho v noci našel tvůj zákazník
```

Klíč je slovo *schválně*. Rozdíl mezi tebou a obětí Gimli není v tom, že ty chyby neuděláš — uděláš. Rozdíl je, že ty si tu chybu způsobíš sám, za denního světla, s kávou v ruce a s možností to vrátit — a ne v noci, v produkci, když spíš.

Tuhle myšlenku dotáhl do institucionální podoby Netflix. V roce 2011 nasadil nástroj se jménem *Chaos Monkey* — opici chaosu —, který za bílého dne a v ostrém provozu *náhodně vypíná servery*. Ne v testu; v produkci, kde běží skutečné filmy skutečným divákům. Logika je přesně ta naše, jen ve velkém: když víš, že ti opice každou chvíli něco vypne, *musíš* postavit systém, který jeden výpadek přežije bez mrknutí — protože jinak tě opice usvědčí dřív než zákazník. Z výpadku, kterého se všichni bojí, udělali *rutinu*, kterou zvládají poslepu. To je Hydra v praxi: každá uťatá hlava (vypnutý server) je pobídka narůst dvě (postavit systém, jemuž jeden server nechybí). Antifragilita není fráze — je to Chaos Monkey přeložený do zvyku.

Chaos Monkey (opice chaosu) — Netflix, 2011. Náhodně ukončuje instance v ostrém provozu, aby vynutil odolnost vůči výpadku jednoho uzlu. Později součást širší „Simian Army“. Institucionalizovaná verze chaosového protokolu z kola výše.

Pramen: Netflix Tech Blog, „The Netflix Simian Army“ (2011); github.com/Netflix/chaosmonkey.

DEMO

`lesk-a-bida-vibecodingu.gaussalgo.com/d/chaos-opice`

Opice chaosu naživo: klikáním vypínej komponenty simulovaného systému — jednou zapojeného sériově, jednou paralelně — a sleduj, kdo výpadek ustojí a kdo spadne celý. Poznáš na vlastní oči rozdíl mezi „drží, jen když drží všechny“ a „padne, jen když padnou všechny“.

Než vyjdeme do věže, jeden zvyk navíc, který mi zachránil víc nocí než všechny ostatní dohromady: obrácení testů z minulé kapitoly na *regresní pomník* každé nehody. Kdykoli něco spadne — v chaosovém protokolu nebo v ostrém provozu —, napiš na tu konkrétní chybu test *dřív*, než ji opravíš. Tím z každého pádu uděláš trvalou pojistku: stejná rána už podruhé neprojde. To je doslova mechanismus Hydry přeložený do kódu — z uťaté hlavy narostou dvě. Systém, který na každý svůj pád odpoví novým testem, je s časem stále odolnější, aniž bys musel předvídat, kde příště praskne.

→ kap. 19: pomník bugu (regresní test). Tady dostává druhý význam: není to jen paměť omylu, je to mechanismus antifragility *— každá rána přidá vrstvu obrany, kterou jsi předtím neměl.*

Věž: rozměry, spolehlivost a konvexita výplaty

EINSTEIN · MATEMATIKA — přeskočitelné

Rozměrová analýza: proč se turbína nevešla. Fyzikální veličina není jen číslo — je to číslo *krát jednotka*, a jednotka je rovnoprávná část hodnoty. Když sečteš 42 (cm) a 5 (palce), nedostaneš 47 ničeho; dostaneš nesmysl, protože jsi sečetl jablka a hrušky. Rozměrová analýza je disciplína, která na tohle dohlíží: v každé rovnici musí *rozměry* na obou stranách souhlasit. Rychlost má rozměr délka/čas; sečíst ji s délkou nelze, ať jsou čísla jakákoli.

Mars i moji turbínu zabilo přesně porušení téhle zásady: kód sčítal a porovnával čísla, jejichž jednotky se rozcházely, a překladač mlčel, protože pro něj to byla jen `float` čísla. Lék je *zakódovat jednotku do typu* — udělat z ní součást datového druhu, ne poznámku v hlavě programátora:

$$\text{vzdálenost}_{\text{cm}} + \text{vzdálenost}_{\text{palce}} \implies \text{CHYBA TYPU}$$

Kompilátor tuhle větu odmítne přeložit — stejně jako odmítne sečíst text a číslo. Past se nedá spustit, protože nejde ani napsat.

V praxi: F# má units of measure *přímo v jazyce (`1.0<cm>` je jiný typ než `1.0<inch>`); v Pythonu totéž zařídí knihovna* pint *(`42 * ureg.cm`); C++ má `std::chrono` pro čas. Princip je všude stejný: ať čísla nesou rozměr a překladač hlídá, že se nesčítá jablko s hruškou.*

Tohle je přesně ta vrstva 2 z obrany do hloubky — a jediná, která by spolehlivě zastavila Gimli, Mars i moji turbínu dřív, než vzniknou.

EINSTEIN · MATEMATIKA — přeskočitelné

Spolehlivost sériově a paralelně. Označ R_i pravděpodobnost, že komponenta i přežije danou dobu (její *spolehlivost*, číslo mezi 0 a 1). Otázka zní: jakou spolehlivost má z nich složený systém? Odpověď závisí jen na jednom — na tom, jak jsou komponenty zapojené.

Sériově (řetěz): systém drží, jen když drží úplně *všechny* články. Pravděpodobnosti nezávislých jevů se násobí, takže

$$R_{\text{sér}} = \prod_{i=1}^{n} R_i .$$

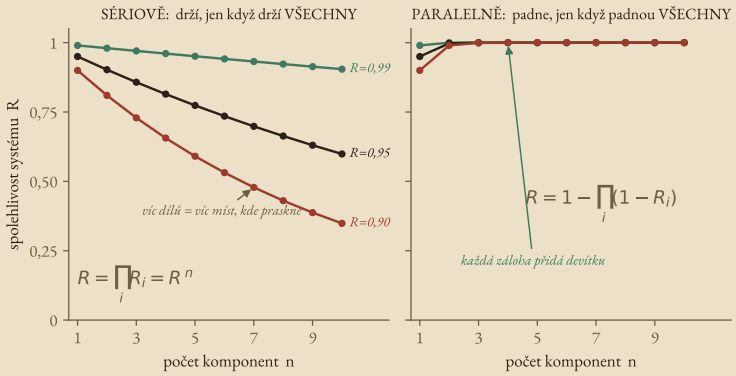
Protože násobíš čísla menší než jedna, výsledek s každým dalším článkem *klesá*. Deset komponent, každá spolehlivá na 99 %, dá systém spolehlivý jen na $0{,}99^{10} \approx 0{,}90$ — devět procent výpadku vyrobila sama délka řetězu. Víc dílů zapojených do řady znamená víc míst, kde to praskne.

EINSTEIN · MATEMATIKA — přeskočitelné

Paralelní zapojení obrací směr. Zapoj komponenty *paralelně* jako zálohy — systém padne, jen když padnou *všechny* najednou. Teď násobíš pravděpodobnosti *selhání* $(1 - R_i)$, a protože i ty jsou menší než jedna, jejich součin se s každou zálohou zmenšuje:

$$R_{\text{par}} = 1 - \prod_{i=1}^{n} (1 - R_i).$$

Dvě zálohy, každá mizerná na 90 %, dají dohromady $1 - 0{,}1^2 = 0{,}99$; tři dají $0{,}999$. Každá další záloha přidá k jistotě jednu „devítku“. Tohle je matematické jádro obrany do hloubky: vrstvy nezapojuj do řady (kde se násobí slabiny), ale paralelně (kde se násobí síly). Gimli mělo motory paralelně — a přesto zhasly oba, protože je *spojoval* jeden sériový článek: společná prázdná nádrž. Skrytá sériová závislost pod zdánlivě paralelní zálohou je nejzákeřnější past spolehlivosti.



Vlevo sériové zapojení: $R = R^n$ klesá — čím delší řetěz, tím slabší celek, i z dokonalých dílů. Vpravo paralelní záloha: $R = 1 - (1 - R)^n$ roste k jistotě — každá záloha přidá devítku. Tři křivky podle spolehlivosti jedné komponenty (0,90 / 0,95 / 0,99). Pozor na skrytý sériový článek pod paralelní zálohou (Gimli: dva motory, jedna nádrž).

EINSTEIN · MATEMATIKA — přeskočitelné

MTBF: kolik vydrží, než praskne. Spolehlivost se v čase rozpadá. Vžitá míra je **MTBF** (mean time between failures) — střední doba mezi poruchami. Když poruchy přicházejí náhodně a nezávisle s intenzitou λ poruch za hodinu, je $\mathrm{MTBF} = 1/\lambda$ a pravděpodobnost, že komponenta přežije dobu t bez poruchy, klesá exponenciálně:

$$R(t) = e^{-\lambda t} = e^{-t/\,\mathrm{MTBF}}.$$

Pointa pro tebe: MTBF *není* záruka. Disk s MTBF 100 000 hodin (jedenáct let) neznamená, že vydrží jedenáct let — znamená, že v poli tisíce takových disků praskne v průměru jeden za sto hodin. Statistika o populaci, ne slib jednomu kusu. Kdo čte MTBF jako záruku, staví Damokla a myslí si, že staví Fénixe.

Vzorec $R(t) = e^{-\lambda t}$ platí pro konstantní intenzitu poruch — tzv. paměťová exponenciála. Skutečné komponenty mají „vanovou křivku“: vyšší poruchovost na začátku (dětské nemoci) i na konci (opotřebení). MTBF popisuje jen ploché dno vany.

Teď to nejtěžší a nejcennější v celé kapitole — místo, kde se z „antifragility“ stane číslo. Odložme mytologii Hydry a zeptejme se přesně: *co to vlastně matematicky znamená, že systému rána prospívá?*

EINSTEIN · MATEMATIKA — přeskočitelné

Antifragilita = konvexita výplaty vůči stresoru. Ať x je stresor — velikost výkyvu, nárazu, zátěže — a $f(x)$ výplata, tedy to, co ze systému při daném stresoru vyjde. Klíč je *tvar* funkce f. Nech stresor kolísat kolem klidové hodnoty μ (jednou o $-d$, jednou o $+d$, stejně velké rány na obě strany). Otázka: bude průměrný výsledek při kolísání lepší, nebo horší než výsledek za klidu? Odpověď dává **Jensenova nerovnost**.

Pro *konvexní* f (prohnutou vzhůru) platí

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]).$$

Levá strana je průměrný výsledek za kolísání; pravá je výsledek za klidu. Konvexní systém tedy z kolísání *profituje*: průměr přes rány leží nad klidem. To je operační definice antifragility — žádná nálada, jen znaménko druhé derivace $f'' > 0$.

Jensenova nerovnost (Johan Jensen, 1906): pro konvexní f je $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$, pro konkávní obráceně. Rovnost jen pro lineární f. Rozdíl mezi oběma stranami se jmenuje Jensenova mezera *a je přesně tou hodnotou, o kterou kolísání pomáhá (konvexní), nebo škodí (konkávní).*

EINSTEIN · MATEMATIKA — přeskočitelné

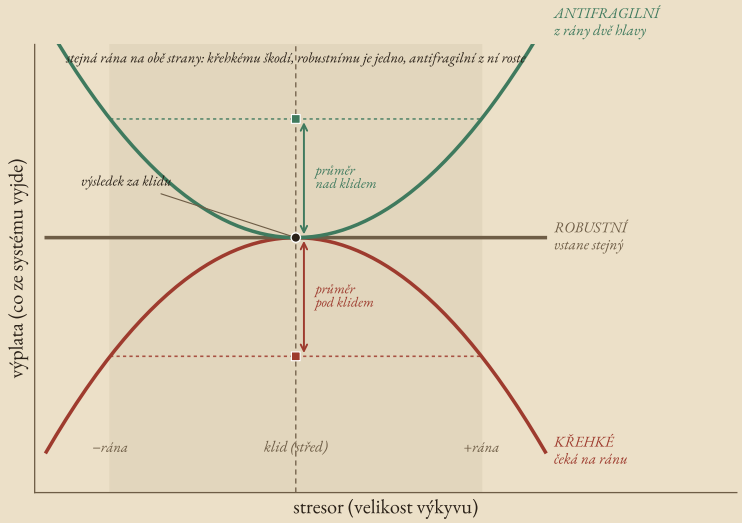
Tři tvary, tři osudy. Táž nerovnost dává celou Talebovu trojici jako tři případy jednoho vzorce:

$$f'' < 0: \quad \mathbb{E}[f(X)] < f(\mathbb{E}[X]) \quad \big(\text{konkávní} — \text{KŘEHKÉ}\big)$$

$$f'' = 0: \quad \mathbb{E}[f(X)] = f(\mathbb{E}[X]) \quad \big(\text{lineární} — \text{ROBUSTNÍ}\big)$$

$$f'' > 0: \quad \mathbb{E}[f(X)] > f(\mathbb{E}[X]) \quad \big(\text{konvexní} — \text{ANTIFRAGILNÍ}\big)$$

Křehký systém (Damoklés) má konkávní výplatu: kolísání mu v průměru ubírá, protože ztráta z velké rány převáží zisk z opačné. Robustní (Fénix) má lineární: na kolísání nezáleží. Antifragilní (Hydra) má konvexní: velké rány mu v průměru přidávají víc, než malé uberou. Antifragilita *je* konvexita — nic víc, nic míň.



Hrdinský graf kapitoly. Tři odezvy $f(x)$ na stresor, všechny procházejí klidem uprostřed. Stejná rána na obě strany ($\pm d$): u konkávní (křehké, sanguine) leží průměr výsledků pod *klidem — Jensenova mezera záporná; u konvexní (antifragilní, verdigris)* nad *klidem — mezera kladná; lineární (robustní) je na kolísání lhostejná. Čtvereček = průměr přes obě rány $\mathbb{E}[f]$, kolečko = klid $f(\mu)$.*

EINSTEIN · MATEMATIKA — přeskočitelné

Jak si vyrobit konvexitu. Konvexita není dar; dá se *postavit*, a všechny naše zvyky z dílny nedělají nic jiného. *Omezená ztráta, otevřený zisk* je konvexní tvar: chaosový protokol tě stojí odpoledne (malá, ohraničená ztráta), ale ušetří ti výpadek v produkci (velký, jinak neomezený zisk). *Zálohy* dělají spolehlivost konvexní: každá přidaná paralelní vrstva přidá devítku levněji, než kolik stojí ztráta z výpadku. *Levné experimenty s velkým možným výnosem* (commit před přepsáním z kap. 18, malá dávka místo velkého vydání) jsou konvexní sázka: prohraješ málo, vyhrát můžeš hodně. Recept na antifragilitu zní: usekni si ztrátu shora a nech zisku otevřená vrátka — a Jensen zbytek udělá za tebe. Přesně tohle je vítr, který místo svíčky rozdmýchá oheň.

Pozor na obrácenou past: konkávní *výplata je systém s omezeným ziskem a neomezenou ztrátou — „sbírej drobné před parním válcem". Tak vypadá spousta zdánlivě bezpečných strategií (šetřit na pojistkách, vypnout zálohy „nikdy je nepotřebujeme"): drobný zisk pořád, katastrofa jednou. Damoklés se převléká za spořivého hospodáře.*

Co si odnést z robustnosti

Poskládej to dohromady. Gimli, Mars i moje turbína nebyly příběhy hlupáků; byly to příběhy *děravých řetězů* — každý článek zvlášť přežitelný, teprve dohromady smrtící. Proto cíl není systém bez chyb; takový neexistuje a kdo ho slibuje, prodává ti Therac. Cíl je systém, který jednu chybu přežije — a v ideálním případě z ní zesílí.

A tři věci si odnes do ruky. *Nespoléhej na jednu vrstvu:* validuj na hranicích, kóduj jednotky do typů, zapojuj obranu paralelně (kde se násobí síly), ne sériově (kde se násobí slabiny) — a hlídej si skryté sériové články pod zdánlivými zálohami. *Předepiš si chaos sám:* vrstvu, kterou jsi neviděl zabrat, ve skutečnosti nemáš, tak si ji vyzkoušej za denního světla, dřív než tě usvědčí zákazník v noci. A *postav si konvexitu:* usekni ztrátu shora, nech zisku otevřená vrátka, a z každého pádu udělej regre-

sní pomník — pak z tebe každá rána udělá silnějšího, ne mrtvějšího. To je Hydra přeložená do čísel: antifragilita je konvexita výplaty, ne nálepka na PowerPointu.

Testy z minulé kapitoly byly první vrstva obrany — chytí, co umíš popsat předem. Robustní design je vrstva pro to, co popsat předem neumíš: pro chybu, kterou nikdo nečekal, protože „to přece nemůže nastat“. A příští kapitola vezme celý ten arzenál — git, testy, robustnost — a postaví z něj skutečnou produkční linku, která od commitu do provozu nepustí nic, co ránu neustojí. Most k ní vede přesně přes moji turbínu: kdyby existovalo standardní rozhraní, na němž palec a centimetr nejdou splést, nevešla by se do žádného příběhu o bolesti. O takových rozhraních — a o tom, jak z hračky, co běží, když se díváš, udělat produkci, co běží, když spíš — je kapitola dvacet jedna.

„Jak bych poznal, že se pletu?“

Konkrétní test k této kapitole máš pokaždé, když si o systému myslíš, že „to určitě nespadne“: *vypni mu to.* Vytáhni síťový kabel, zabij proces, zaplň disk, podstrč do vstupu „palivo = −500“ — schválně, za denního světla, s možností to vrátit. A pak se dívej. Když spadne *celý*, právě jsi levně našel křehké místo, které by tě jinak našlo samo, v noci a draho — postavil jsi Damokla. Když se zvedne *stejný*, máš Fénixe: slušný cíl. A když z pádu vyjde *odolnější* — protožes na tu chybu hned napsal test, který ji už podruhé nepustí —, postavil jsi Hydru. Kapitola se plete jedině tehdy, jestli mi ukážeš systém, kterému jsi tohle udělal, on to celé ustál, a přitom v něm *není žádná* paralelní vrstva, žádná validace na hranici, žádná omezená ztráta — čistá souhra bez pojistek, která přežila. Pak jsi buď našel zázrak, nebo — pravděpodobněji — jsi tu ránu ještě nezasadil dost silnou. Tak přidej. Vítr, který zhasne svíčku, rozdmýchá oheň; zjisti, které z těch dvou jsi postavil, dokud tě to stojí jen odpoledne.