

Tietojärjestelmien luotettavuus

Aleksandr Koltsoff
czr@iki.fi
<http://koltsoff.com/>

Aiheet

- Luotettavuuden merkitys
- Musta laatikko on rikki
- Kuka rikkoi mustan laatikon?
- Nopeammin ja edullisemmin
- Merkitys ohjelmistotuotannolle
- Entä sitten?

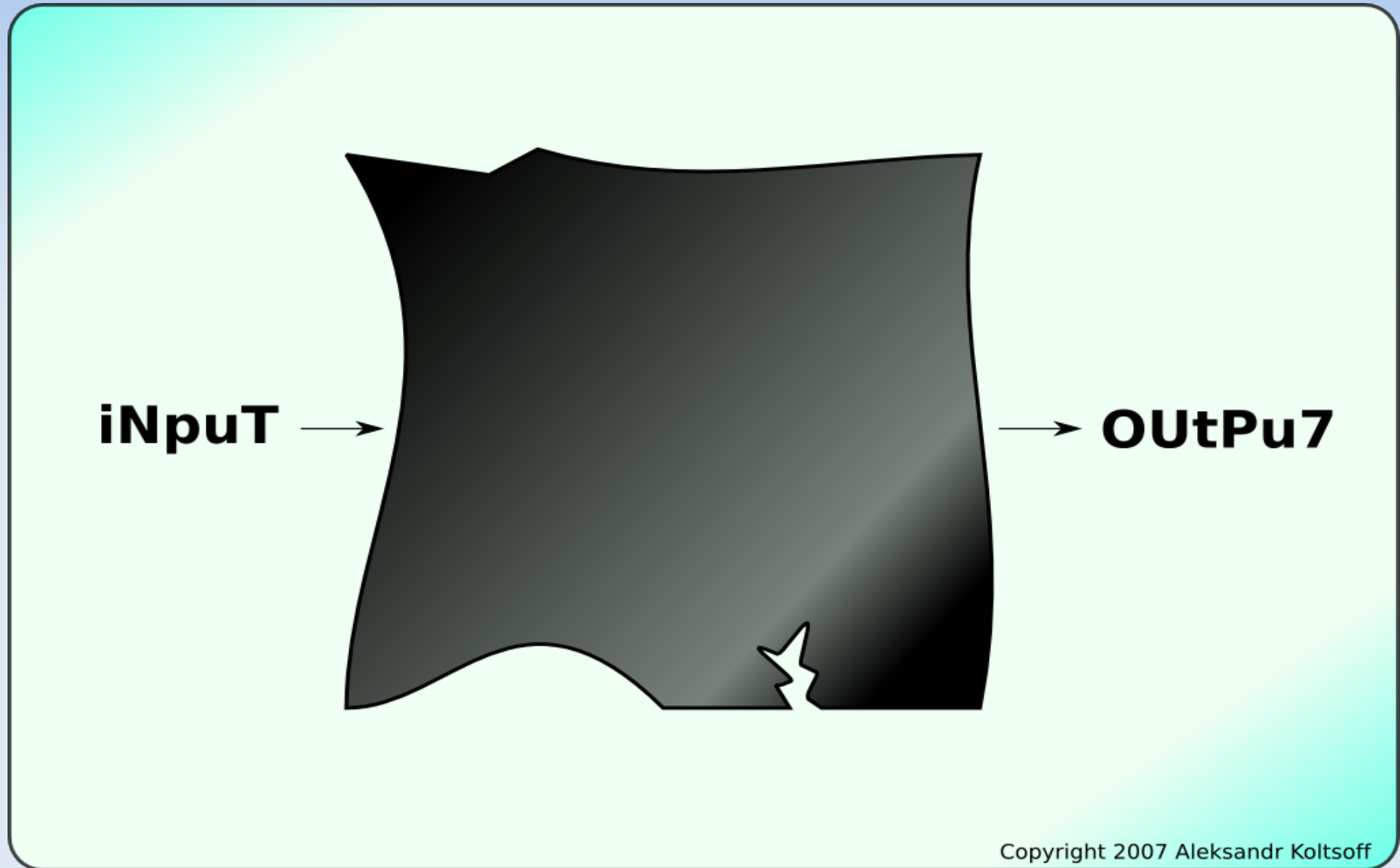
Luotettavuuden merkitys 1/2

- Merkitystä vain tärkeissä järjestelmissä
- Kaikki järjestelmät eivät ole tärkeitä
- SOA/WS/InsertHypeHere
 - Erillisten järjestelmien keskinäiset riippuvuudet?
 - Yksi “mätä” järjestelmä voi helposti pilata kokonaisuuden
 - Hajautetut järjestelmät vaativat toimivan verkon allensa joka myös pitää käsittää lopullisen järjestelmän osana

Luotettavuuden merkitys 2/2

- Esimerkkejä vääristä odotuksista:
 - Virtualisointi säästää rahaa
 - VoIP:llä korvataan kallis lankaverkko ja kännykät
 - UPS hoitaa kaiken (entä ilmastointi?)
 - “Onhan meillä RAID” (entä bare-metal palautus?)
 - Päivitys korjaa asian (ja parin viikon päästä huomataan kriittinen vika?)
 - Automatisointi on aina parempi vaihtoehto
 - Active Directory (vaatii toimivan verkon mutta verkko ei toimi jos AD ei toimi)

Musta laatikko on rikki



Musta laatikko loittonee ideaalista jatkuvasti

Musta laatikko 1/4

- Järjestelmien luotettavuus rakentuu kahdesta tekijästä:
 - Järjestelmä toteuttaa halutut toiminnot ideaalisessa mustassa laatikossa oikein, ennustettavasti ja toistettavasti (ei sw-bugeja)
 - Järjestelmän suoritusympäristö on tarpeeksi lähellä ideaalia mustaa laatikkoa (esityksen aihe)

Musta laatikko 2/4

Deterministisyys

```
extern volatile int tili;

if (nosto < tili) {
    tili -= nosto;
    return tili;
} else {
    error("ei katetta");
    return tili;
}
```

Musta laatikko 3/4

- Järjestelmän tila
 - Sisältää toiminnallisuuden (koodi) sekä muistin (RAM ja/tai taltioidut)
- Järjestelmän tilan poikkeustilanteet
 - Tila lakkaa olemasta (hard-failure)
 - Tila tai siirtymiset tilojen välillä muuttuvat sisäisesti lopullisesti tai virtasyklin ajaksi
 - Järjestelmä muuttaa sisäistä toimintaansa rajoitetuksi ajaksi

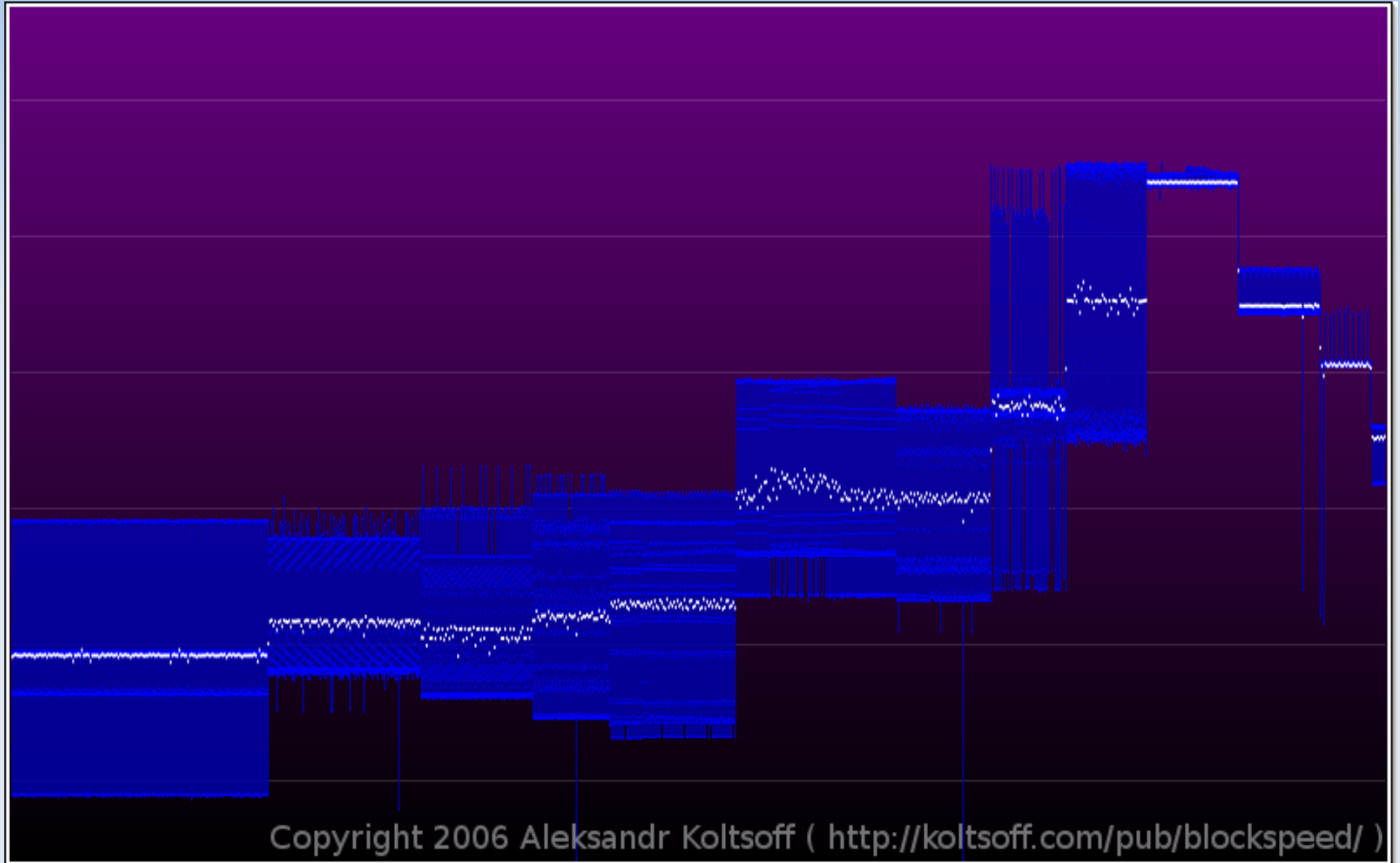
Musta laatikko 4/4

- Tieto: järjestelmän osan prosessoinnin ja muunnosten tulos (output)
- Miten tiedosta voi päätellä sen oikeellisuuden?
- Esimerkkinä järjestelmä joka summaa kaksi numeroa yhteen
 - Input: 1, 2
 - Tieto: 3
- Voiko numerosta 3 päätellä että se “pitää paikkansa”? Entä jos sen alin bitti vaihtuu? Jos + operaatio vaihtuu - :ksi?

Mustan laatikon ongelmat

- Nopeuden kasvatus vaatii pienemmät signaalointijännitteet (häiriöalttius)
- Siirrettävän datan ja tiedon määrä jatkuvassa kasvussa
- Viat muuttumassa enenevässä määrin väliaikaisiksi:
 - Jos rauta rikkoutuu täysin, se on “hyvä”.
 - Jos rauta muuttaa toimintaansa pysyvästi, se ei ole “hyvä”, mutta sen voi vielä ehkä huomata.
 - Jos rauta muuttaa toimintaansa väliaikaisesti...

Kuka rikkoi mustan laatikon?

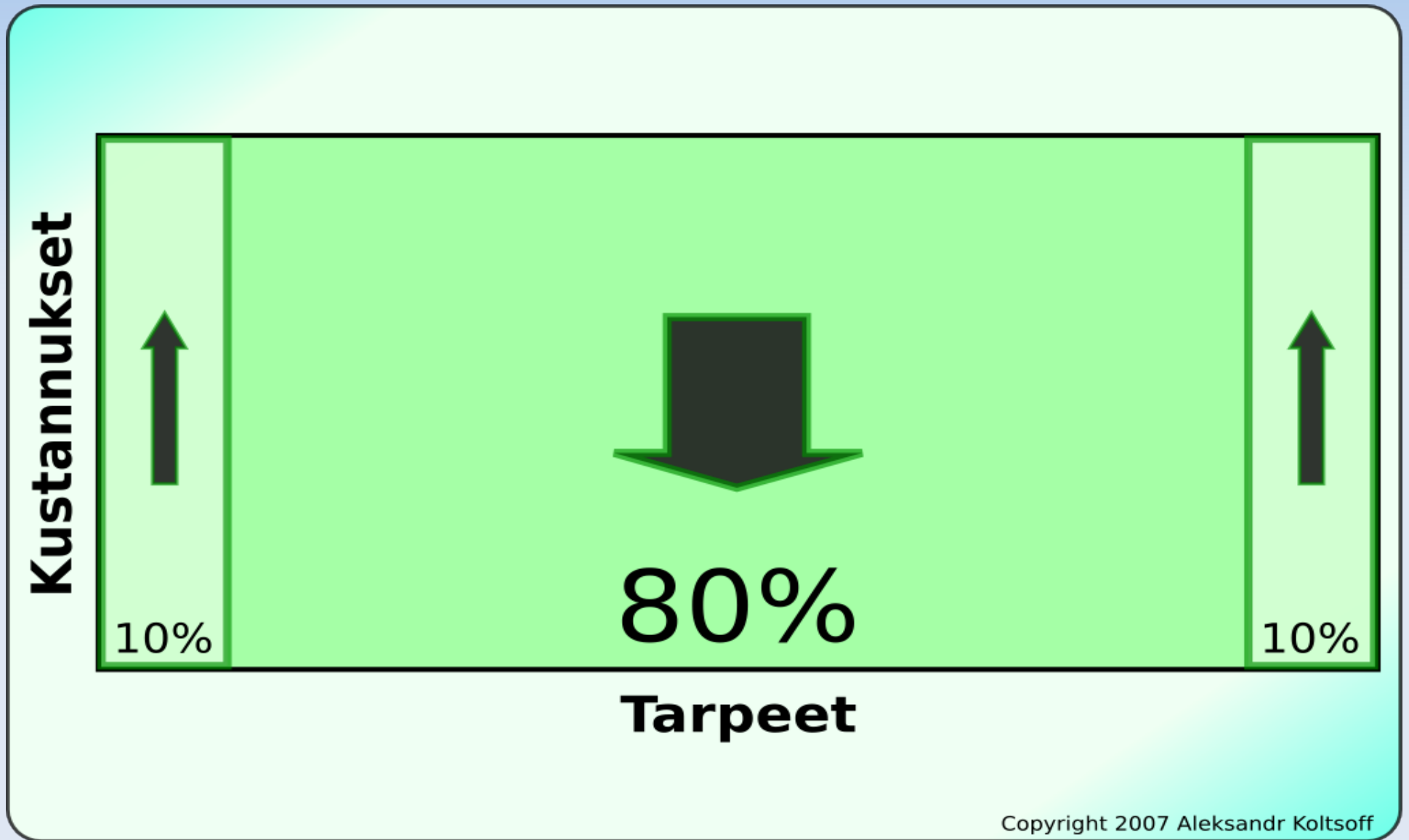


Yhden kovalevyn lukunopeus luettaessa sen sisältö alusta loppuun (epätyypillinen)

Moderni markkinatalous 1/2

- Komponenttimarkkinoiden toiminta:
 - Tuotteiden suunnittelukustannukset kasvavat per tuote
 - Tuotteiden valmistuskustannukset pienenevät per tuote
 - Tuotteet ovat halvempia kun niitä myydään enemmän
 - Halvat tuotteet myyvät enemmän (mikäli täyttävät tarpeen näennäisesti)
- Täten on halvempaa valmistaa tuotteet samoista komponenteista joita muutkin käyttävät

Moderni markkinatalous 2/2



Hintapaineet eri tarvealueilla. 80%-alueen ulkopuolella suhteelliset hinnat kasvavat.

Monokulttuuri

- “Turhat” ominaisuudet jotka jäävät 80% tarveikkunan ulkopuolelle rasittavat tuotantoa, joten ne karsitaan pois.
- Uusia ratkaisuja ja tekniikoita on vaikea tuoda markkinoille koska ne eivät ole 80% ikkunassa.
- Yksinkertaisimmat ratkaisut joiden valmistuskustannus on potentiaalisesti pienin valtaa 80% ikkunan (tekniikka etabloituu / “Ethernet”-efekti).

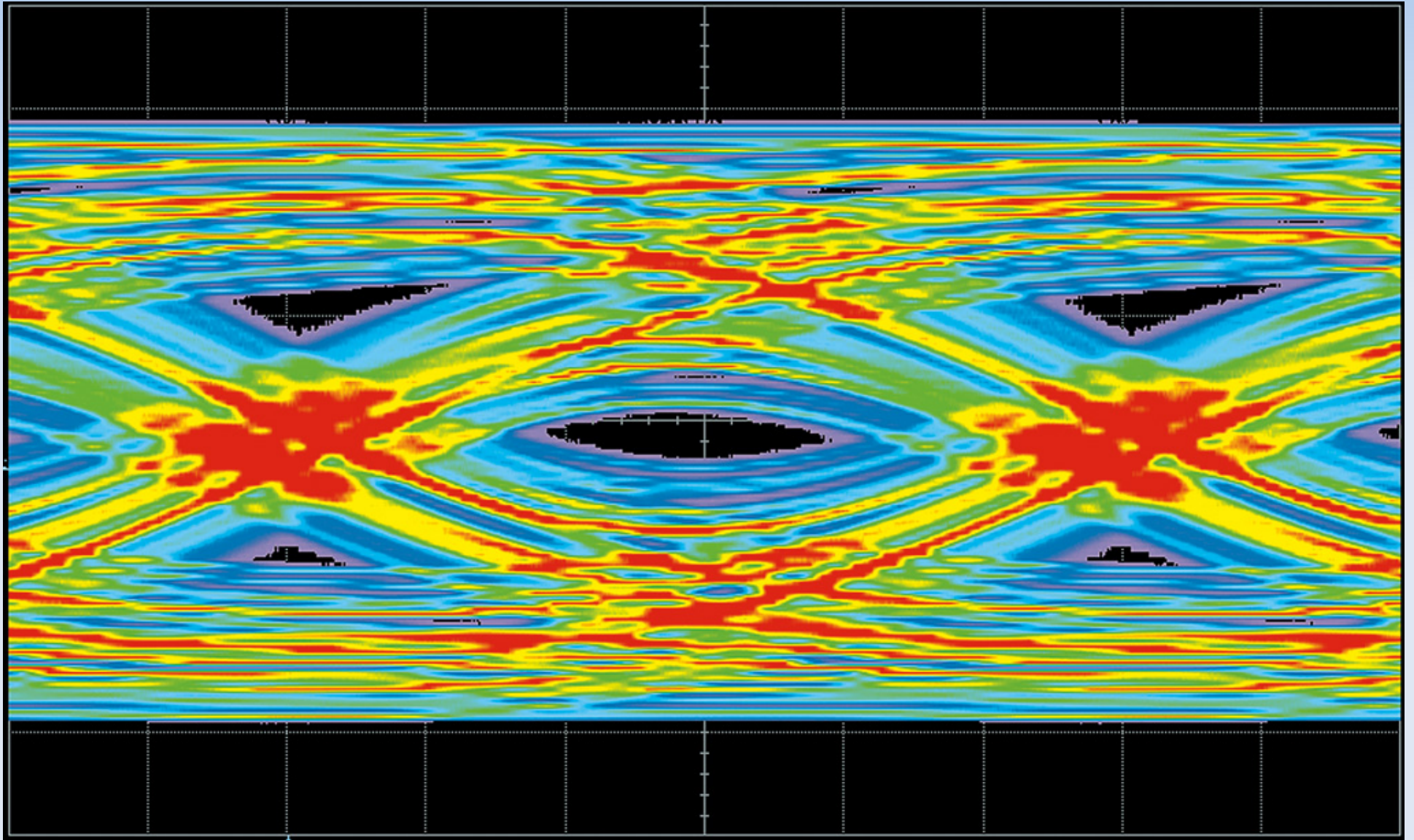
Odotukset ja niiden kohtaaminen

- Mikäli luotettavuus ja laatu olisivat aitoja valintakriteerejä, kasvaisivat ne molemmat “rautapuolella” (80%-sääntö).
- Ihmiset eivät kuitenkaan valitse aidosti luotettavia ja laadukkaita tuotteita.
 - Järjestelmät ja tuotteet ovat liian monimutkaisia ostajien ymmärtää.
 - Markkinointitekstiin on helpompi uskoa kun se toistetaan tarpeeksi monta kertaa.
 - “Uusi on parempi”
- Näinollen laatu ei kannata valmistajalle.

Monokulttuuri elektronikassa

- Rinnakkaisväylien tilalle sarjaväylät koska ne tarvitsevat vähemmän polkuja ja siten halvempia valmistaa.
- Galvaanista erotusta ei käytetä enää tallennuspuolella koska se maksaa liikaa (noin 0,5 USD per kovalevy).
 - Aiheuttaa lisää harmaan signaalin alueita kun sähkönsyötössä ilmenee ongelmia.
- Valitettavasti esimerkkejä on paljon lisää.

Nopeammin ja halvemmin



10Gbps “kanttiaalto” (kuva Le Croy:n WaveExpert-sivuilta, ilman lupaa)

Sarjaliikenne on pop

- Matalammat jännitteet (energiankulutus pysyy kurissa taajuuden noustessa)
- Differentiaalisignaalointi käytössä (ainoa häiriösuoja)
- Huomattavasti korkeammat taajuudet (helpompi myydä GHz-ihmisille)
- Vähemmän johtimia (halvempi)
- Osa ratkaisuista toteutettu suljetuissa ad-hoc työryhmissä (SATA, TDMI), osa taas standardoitu (PCIe, IB, 10GBASE-CX4)
- Rinnakkaisliikenne poistumassa.

“Vikasietoinen” muisti (ECC)

- Vikasietoisuus RAM:issa toteutetaan lisäämällä muistimoduuleihin yksi kapeampi muistipiiri.
- Ylimääräiseen muistipiiriin tallennetaan aina kirjoitusvaiheessa redundantit bitit (ECC).
- Lukuvaiheessa muistinohjain (yleensä piirisarja) laskee ECC:n uusiksi normaalibiteille ja vertaa tallennettuihin ECC-bitteihin. Poikkeavuudet aiheuttavat lokiviestin. Tietyissä tapauksissa yhden bitin poikkeavuus voidaan korjata.
- Tekninen ratkaisu on yksinkertainen, mutta toteutus tippuu 80% ikkunan ulkopuolelle joten kallis (perus PC:ssä piirisarja ei tee ECC:tä).
- Kuinka moni on lukenut MCE-lokeja koskaan?

Paremmat mustat laatikot

- Signaalointipolkujen EMF-suojaaminen
- Signaalointipolkujen monistus
- Ohjelmistotason eheystarkistus
- Asynkroninen logiikka/signalointi (esim. AMULET)
- Uudet VLSI-prosessit joilla dynaamiset vuotovirrat (dynaamiset kellot mahdollisia)
- Optinen signalointi & prosessointi
- Evoluutionäärinen elektroniikka (GA:t VLSI-tasolla)
- Ohjelmistotasolla geneettiset algoritmit replikoituna (tietyissä määrin myös neuroverkot).

Merkitys ohjelmistotuotannolle 1/2

- Ole paranoidi ja kyseenalaista aina.
- Halvalla ei saa hyvää, mutta ei aina kalliillakaan.
- Varaa aikaa ympäristösi ymmärtämiseen.
- Kaikki menee rikki.
- Älä luota mihinkään yksittäiseen tietoon liikaa.
- Jos olet epävarma niin mittaa ja testaa
 - Yhdistelmätestaus kuormituksen alaisena suositeltavaa.

Merkitys ohjelmistotuotannolle 2/2

- Käytä kehityksen rinnalla “fault-injection”-tekniikoita, mieluiten siten että se on aina testeissä päällä ja toistettavissa.
- Lisäksi rautatason fault-injection on suositeltavaa
- Tiettyjen vikojen tuottaminen on mahdollista myös virtualisoidussa tai emuloidussa ympäristössä.
- Myös formaaleista menetelmistä (esim Z) voi olla apua.

Summa summarum 1/2

- Luotettavuuden lisääminen:
 - Nk. Don Quijote-tekniikka, eli lisätään vikasietoisuutta rahalla (RAID, RAM-RAID, sähkönsyötön varmennukset, yms)
 - Suunnitellaan tietojärjestelmä lähtökohtaisesti operoimaan rikkiinäisissä ja epäluotettavissa mustissa laatikoissa.
 - Jos tuote on halpa, unohda koko juttu ja jätä markkinoinnin hoidettavaksi.

Summa summarum 2/2

- Ostajana arvioi arvot seuraaville:
 - Tiedon saatavuus
 - Tiedon oikeellisuus
 - Järjestelmän saatavuus
 - Järjestelmän oikeellisuus
- Miten hoitaa prosessit silloin kun järjestelmä on täysin toimintakelvoton?
- Onko saatavuudelle kriteerejä laajempien rikkojen jälkeen? (laitetilassa tulipalo)
- Luotettavia järjestelmiä saa vain niitä vaatimalla ja niistä maksalla.

$$1 + 2 = 3$$

$$3 - 2 = 2$$

Kysymyksiä?