TEORIBLOKK #3: "Hvilken innflytelse har UNIX filosofi på dagens databehandling?"

UNIX filosofien som ble etablert på slutten av 1960-tallet ble utviklet av Ken Thompson, og filosofien gikk ut på at en utvikler skulle lage enkle, korte, klare, modulære og utvidbare kodeeksempler som andre utviklere kan utvikle videre (Wikipedia, 2018). Filosofien ble først dokumentert av Doug McIlroy i Bell System Technical Journal i 1978. Filosofien gikk ut på at (1) hvert program skal ha en funksjon og gjøre den bra, (2) forvent at hvert program skal ha en output som er en input i et annet, design og lag software som skal prøves ut tidlig, bruk verktøy i stedet for ufaglært hjelp for å forenkle programmeringsoppgaver.

Senere forenklet han denne til å lyde som følger; skriv programmer som gjør en ting og gjør det bra. Skriv programmer for å jobbe sammen. Skriv programmer for å håndtere tekststrømmer, fordi det er et universelt grensesnitt.

Flere har i ettertid kommet med egne versjoner av filosofien. Eric Raymond lister i 2003 opp 17 UNIX regler i boken “The Art of Unix Programming”:

Rule of Modularity

Rule of Clarity

Rule of Composition

Rule of Separation

Rule of Simplicity

Rule of Parsimony

Rule of Transparency

Rule of Robustness

Rule of Representation

Rule of Least Surprise

Rule of Silence

Rule of Repair

Rule of Economy

Rule of Generation

Rule of Optimization

Rule of Diversity

Rule of Extensibility

(Wikipedia(2), 2018)

Filosofien sier noe om hvordan en skal utvikle programvare og systemer. Det er viktig å påpeke at denne filosofien ikke er noe som ledere har utviklet for at utviklerne skal lage bedre programvare, men den har rot i erfaringer hos utviklerne selv. Det er heller ikke regler en må følge, men en filosofi rundt hvordan en utvikler programmer og skriver god kode. Dette er prinsipper som også har påvirket dagens databehandling, og spesielt innenfor Open Source er disse reglene nyttige som retningslinjer for hvordan en utvikler programmer. Når en jobber i et Open Source miljø er det mange mennesker som bidrar i miljøet og det kan være krevende å sette seg inn i koden til andre. Ved å følge regler som modularity og transparency gjør en det lettere for andre å sette seg inn i og forstå koden. En forstår også lettere hvordan strukturen i koden er om man er tydelig på at man har hatt UNIX filosofien som grunnlag for hvordan koden er bygd opp.

Mikroservicer er også påvirket av UNIX filosofien. Mikroservicer er små og spesifikke, er laget for å utføre en enkelt funksjon, og vi finner dem overalt på internett. Nettsteder som Facebook, Google og Amazon bruker det og for eksempel i et enkelt googlesøk blir mer enn 70 mikroservicer brukt (LeanIX, 2017).

UNIX filosofien kjenner vi også igjen i objektorientert programmering. Vi har begreper som cohesion, encapsulation, coupling og localizing change som har mange likhetstrekk med UNIX filosofien. Disse begrepene dreier seg om å programmere med tanke på spesifikke ansvarsområder, ikke ha for mange koblinger, og at endringer skal holdes lokalt. Dette er sentrale begreper i responsibility driven design, som er en teknikk for å designe et program basert på at objekter har sine ansvarsområder og deler en viss type informasjon.

Vi har nevnt eksempler på at UNIX filosofien har påvirket dagens databehandling, men samtidig er det også ulikheter. UNIX filosofien var ment å gi makt og ansvar til brukeren slik at en selv kan bruke de modulene som ligger der og lage skript som gjør det en ønsker å gjøre. Dette er det imidlertid få brukere som gjør, da de fleste brukerne ikke har kompetansen til å skrive et slikt skript. Moderne software har lagt lista lavere og har forenklet bruken slik at flere brukere skjønner hvordan den skal brukes. Ulempen med å forenkle softwaren er at dette går ut over funksjonaliteten, og de mer

profesjonelle brukerne får begrensede muligheter. Det er derfor viktig å balansere enkelhet med funksjonalitet (Bytebaker, 2009). Windows ble lansert for å beskytte brukerne fra alt som kunne få dem i trøbbel, det vil si at en ikke skulle behøve gode kunnskaper innen IT for å bruke operativsystemet. Filosofien bak operativsystemet var at brukeren er redd for datamaskiner og trenger å bli beskyttet fra kompleksiteten i dem. Dette førte også til at noe av kontrollen og mulighetene til brukeren ble borte da Windows ble designet for å unngå at brukeren gjør noe galt ved et uhell. Et eksempel på dette er at både Windows og UNIX har et Command Line Interface (CLI), men der UNIX sin CLI har mye funksjonalitet for brukeren, har CLI i Windows lite funksjonalitet (opensource.com, 2014).

En sentral del av UNIX filosofien er at en skal lage små programmer som gir en output til et annet program. Dersom en skal gjøre større og mer komplekse oppgaver blir dette et problem. En må da gjøre en liten del av oppgaven i et program, før en går videre til neste program for å gjøre neste steg. Dermed blir dette fot tidkrevende og komplekst. Dette fører til at en lager programmer med flere funksjoner, og ofte vil funksjonene til programmene overlappe hverandre. Ta for eksempel Microsoft Word og Excel; Word hovedansvar for tekstbehandling mens Excel har hovedansvar for databehandling i regneark. Likevel har Word regnearksfunksjoner og Excel har tekstbehandlingsfunksjoner. Dette er logisk da en ikke ønsker å bytte program om en bare skal gjøre enkle oppgaver utenfor det programmet en jobbe i sin hovedfunksjon. Dersom en skriver et dokument i Word og trenger å lage en enkel tabell med et par regnearkfunksjoner behøver en ikke gå inn i Excel for å gjøre dette (Cook, 2010).

Om vi ser på blockchainteknologien presentert i artikkelen i pensum (Blockchain Technology: Principals and Applications), ser vi ingen klar sammenheng med UNIX filosofien. Det kan likevel være visse likheter. En blockchain består av mange blokker i som er lenket sammen. Dersom en node ønsker å endre informasjonen som ligger i kjeden, gjøres dette ved at andre verifiserer endringen og en ny blokk legges til i kjeden. I denne prosessen har enkelte deler av nettverket en spesifikk oppgave, nemlig å verifisere en denne endringen. Bitcoin er den mest kjente blockchain i dag, og her er det utvinnerne som gjør denne verifikasjonen. Dette er spesialiserte og svært kraftige maskiner som kjører avanserte algoritmer.

Det er ikke bare innen kryptovaluta at blockchain blir brukt. Også andre tjenester kan ha nytte av teknologien. For eksempel er Guardtime et firma som selv sier at de bruker UNIX filosofien i utviklingen av produktet deres. De bruker abstraksjon og innkapsling for å lage ulike lag og hvert av lagene gjør en ting og gjør den bra (e-Estonia, 2017). Videre er det i *Blockchain Technology: Principals and Applications* presentert flere ulike blockchain som ikke dreier seg om kryptovaluta. Gridcoin er

for så vidt en kryptovaluta, men denne fungerer slik at en blir kompansert for å delta med datakraft inn i forskningsprosjekter, Etherum er en blockchain som skal behandle kontrakter. Ripple Labs utvikler blockchainteknologi for betaling direkte mellom selskaper. Blockchain kan også brukes til stemmesystemer og oversikt over personnummer. Dette kan relateres til UNiX filosofien med at alle har spesialiserte oppgaver og ansvarsområder. Pilkington,

Kilder

Bytebaker. (2009). *The unix philosophy and the common man*. Hentet fra: https://bytebaker.com/2009/09/22/the-unix-philosophy-and-the-common-man

Cook, J. (2010). *Where the Unix philosophy breakes down.* Hentet fra: https://www.johndcook.com/blog/2010/06/30/where-the-unix-philosophy-breaks-down/9

e-Estonia. (2017). *Guardtime awarded best government emerging technology at WGS 2017*. Hentet fra: https://e-estonia.com/guardtime-awarded-best-government-emerging-technology-at-wgs-2017

LeanIX. (2018). *The philosophy behind microservices in 7 minutes.* Hentet fra: https://blog.leanix.net/en/the-philosophy-behind-microservices-in-7-minutes

oensource.com. (2014). *Linux philosophy*. Hentet fra: https://opensource.com/business/14/12/linux-philosophy

Pilkington, M. 2016. *Blockchain Technology: Principles and Applications*. Hentet fra: https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=2662660

Wikipedia. (2018). *Unix-filosofien.* Hentet fra: https://no.wikipedia.org/wiki/Unix-filosofien

Wikipedia(2). (2018). *Unix philosophy.* Hentet fra: https://en.wikipedia.org/wiki/Unix_philosophy