KUNSTEN
SOM
FORUM
ART AS FORUM
05

Crisis Computing

Linda Hilfling Ritasdatter

Linda Hilfling Ritasdatter
Crisis Computing
Oversat fra engelsk af Mathias Overgaard efter
Crisis Computing

Layout og omslag: Eckardt ApS, 21221281.dk
Tryk: Sangill Grafisk ApS
Sat med Arnhem
Papir: Design Offset Naturhvid, 100g/250g

ISBN: 978-87-7945-127-8

1. udgave, 1. oplag
Printed in Denmark 2022

NY CARLSBERG FONDET
NEW CARLSBERG FOUNDATION

Kunsten som Forum er støttet af Ny Carlsbergfondet

Linda Hilfling Ritasdatter

Crisis Computing

På dansk ved Mathias Overgaard

Billedkunstskolernes Forlag
Det Kongelige Danske Kunstakademi 2022

Forord

"Enhver teknologi, der er tilstrækkeligt avanceret, er umulig at skelne fra magi" hævdede den britiske science-fiction-forfatter Arthur Clarke for snart 60 år siden. I samtidens fejring af den digitale teknologi dyrkes denne forveksling mere insisterende og overlagt end nogensinde før. Spektakulære visioner om 'den fjerde industrielle revolution' eller 'den anden maskinalder' fremstiller regelmæssigt medieteknologiske infrastrukturer som garant for en immateriel, friktionsfri automatisering af alle de processer, der holder vores verden i gang og, ikke mindst, producerer mere innovation og udvikling.

I sin kunstneriske praksis og sin forskning er Linda Hilfling Ritasdatter frem for alt optaget af at afmytologisere dette teknologiens postulerede trylleri. Ligesom blandt andre Susan Schuppli, Hito Steyerl, YoHa og Heath Bunting arbejder Ritasdatter inden for et mediekunstnerisk felt, som ikke først og fremmest har til formål at producere kunstværker, der kan cirkulere gnidningsfrit på *Instagram* og andre digitale platforme. I stedet arbejder de sig ned i de lag af materielle komponenter og modstridende processer, som producerer, regulerer og vedligeholder mediernes

glatte overflader, men konsekvent holdes ude af brugernes synsfelt. Enten ved at afdække og repræsentere skæve magtstrukturer eller, som i Ritasdatters tilfælde, gennem strategiske interventioner, som overtager eller hacker deres materielle funktionsmåder.

Ritasdatter er uddannet fra blandt andet Piet Zwart Institute ved Willem de Kooning akademiet i Rotterdam, og hendes pågående kunstneriske forskningsprojekt, *The Labour of Automation*, er tilknyttet Malmö Universitet, Det Kgl. Danske Kunstakademi og Goldsmiths, University of London. I modsætning til de friktionsfri narrativer om teknologien sætter hun ind dér, hvor systemerne bryder sammen, hvor vi ser deres utilstrækkeligheder og, navnlig, de ulige magtforhold, de både bygger på og til stadighed underbygger. Nærværende tekst, "Crisis Computing", er fra den første del af dette projekt, hvor hun som en del af sin PhD-forskning i en årrække har arbejdet med at optrævle tråde, der løber fra mediernes dybdestrukturer og på kryds og tværs af kloden regulerer den globale transport, ikke bare af information, men også af penge, materiel og mennesker. Gennem en både kritisk og humoristisk intervention i det forkætrede og angiveligt forældede programmeringssprog COBOL afdækker Ritasdatter dels det lavt værdisatte menneskelige arbejde, som ligger bag de automatiserede processer, men forbliver usynligt fra de brugerflader, mange til daglig interagerer med, og dels hvordan en neokolonial verdensorden herved vedligeholdes – gennem netop den teknologi, som til stadighed lover om dens (snarlige) afskaffelse.

Som moderne programmeringssprog er COBOL designet til at kunne løsrives fra det maskinelle miljø, hvori det er udviklet og implementeret i første omgang, og fra den programmør, som har skrevet koden, for at kunne appliceres hvor som helst og når som helst i en global horisont. Således fungerer det som en form for materiel, systemisk realisering af oplysningstidens universalistiske dagsorden, som blandt andet har legitimeret udbytning af og vold imod ikke-europæiske folk med et ideal om objektiv, videnskabelig udforskning og transparens. Men som Ritasdatter viser os, er denne friktionsløse universalisme en myte, der dækker over, at kriser og sammenbrud er en del af dagens orden, selvom de sjældent er synlige i hæveautomaten, billetreservationssystemet eller på lønsedlen, hvorfra vi interagerer med processerne.

"Crisis Computing", er en nedskrevet version af en *performance lecture*, som Ritasdatter holdt i Kunsten som Forums regi i februar 2021. *Performance lecturen* er en mundtligt funderet genre med rødder i 1960ernes koncept- og performancekunst, som placerer sig i mellemrummet mellem kunstneriske processer og akademisk ræsonnement, hvorved det bliver muligt at producere indsigter, som ikke kan opstå i hverken kunsten eller i forskningen, når de holdes adskilt. Ofte – som det er tilfældet hos både komponisten John Cage, som var en af dem, der gjorde genren berømt, og hos Ritasdatter – er genren opbygget omkring humoristiske sammenfald og poetisk-associative komponenter, som i denne tekst blokerer og forstyrrer det glatte

narrativ om teknologien som garanten for fremtid og udvikling.

Ritasdatters optrævling af zombiesproget COBOL bliver øjenåbnende, når vi i en skandinavisk kontekst rejser spørgsmål om kunsten og dens fællesskaber, fordi den demonstrerer den helt håndfaste udmøntning af en neokolonial logik, som foregår på et medieinfrastrukturelt niveau. Når vi i kreative og akademiske fælleskaber er optagede af at afkolonisere egne arbejdsprocesser, giver afdækningen af disse *back-back-ends* os et konkret indblik i, hvordan sådanne fællesskaber – om de udfolder sig omkring litteraturproduktion, designprocesser, udstillingssammenhænge eller på digitale platforme – infrastrukturelt understøttes og logistisk muliggøres af et ensformigt, krævende og konsekvent usynliggjort arbejde, udført af mennesker der, pr. definition, ikke har adgang til disse kreative, frigørende fællesskaber.

Mens de senere års fornyede fokus på en styrket anerkendelse af omsorgs- og vedligeholdelsesarbejde i et intersektionelt og feministisk perspektiv har udpeget ulige prekære vilkår for forskellige kunst- og kulturarbejdere og problematiseret innovationsdyrkelse og individualisering, lader Ritasdatters mediearkæologiske udgravning os få syn på et andet mindst ligeså usynligt vedligeholdelsesarbejde, som foregår i såkaldte udviklingslande som for eksempel Indien, og som er uundværligt for at opretholde basale funktioner i et hverdagsliv, der leves i mere velstående lande tusindvis af kilometer væk.

Med sit arbejde tvinger Ritasdatter os til at spørge, hvilke individer i nutidens globaliserede netværkssamfund, der kan rejse, udveksle ideer og udvikle spændende samarbejdsformer, og hvilke individer, der er outsourcede og låst i en hierarkisk, virtuel struktur, hvor de er *syncet* tidsligt, men ikke rumligt, med deres kolleger i de såkaldt udviklede lande. Hun demonstrerer hvordan modellerne for teknologisk outsourcing, som præger vores samtid, er designet med henblik på at fjerne lavstatusarbejdere fra arbejdspladser centreret om udviklingsarbejde, design, og indholds- og kulturproduktion. På den måde er de med til at opretholde billedet af sådanne arbejdssteder som kreative, sjove miljøer for selvrealisering og selvudvikling.

Solveig Daugaard

Crisis Computing

LINDA HILFLING RITASDATTER

PROLOG: Bangalore, en mandag morgen i efteråret 2014.

Jeg sidder i et lille konferencerum ved siden af et motionscenter på penthouse-etagen af et moderne lejlighedskompleks i den sydlige del af den sydindiske storby Bangalore. Om få minutter begynder min første lektion i det (efter sigende) forældede programmeringssprog COBOL. En læreproces, der vil ende med at tage seks år, og hvor dette mødelokale blot er et af flere undervisningssteder, hvorfra jeg undersøger hvordan globale flows og datastrømme opretholdes af neokoloniale arbejdsstrukturer og usynligt software som en form for *Crisis Computing*.

Datastrømme fremstilles ofte som en jævn og gnidningsløs afvikling af automatiserede instruktioner. I dette essay vil jeg imidlertid argumentere for, at sådanne antagelser er baseret på et universelt blik, som er blindt for

afviklingens underliggende og usynlige sammenfiltring med sammenbrud og vedligeholdelse. COBOL er et godt eksempel på en sådan sammenfiltring, hvorfor min tekst er struktureret igennem en serie af lektioner, der afspejler min egen læringsproces, samtidig med, at den reflekterer over – og gennem – dette umiddelbart forældede sprog som en del af min kunstneriske forskningspraksis.

Lærerinden rækker mig en undervisningsbog og begynder COBOL-lektionen med at introducere baggrundshistorien for udviklingen af programmeringssproget. Hun tager mig med tilbage til USA i slutningen af 1940'erne.

1. LEKTION: AFVIKLING

1.1

I 1947 bliver et grillet møl fundet i kredsløbet på Mark II computeren på Harvard. Afviklingen af insektet finder sted i strømme af databehandling og ender med at sætte afviklingen af disse på pause. En ingeniør med humoristisk sans tager det grillede insekt ud af kredsløbet og klistrer det ind i computerlogbogen sammen med følgende bemærkning: '15.45: Relay*70 Panel F (møl) i relæ. Første egentlige tilfælde, hvor et *bug* er blevet fundet.'[1]

1 'The First "Computer Bug"', The Naval Surface Warfare Center, Dahlgren, VA., 1988, https://www.history.navy.mil/our-collections/photography/numerical-list-of-images/nhhc-series/nh-series/NH-96000/NH-96566-KN.html (sidst besøgt 2. december 2021).

Jeg nærstuderer et billede, der er taget af ingeniøren næsten 40 år senere. En gruppe ældre mænd er samlet omkring noget, der ligner en piedestal, hvorpå der står en gravsten ved siden af en krysantemum i en potte. Helt forrest i højre hjørne af billedet står en lille, rynket dame i et jakkesæt fra det amerikanske marinekorps. Hun er den humoristiske ingeniør. Hendes navn er Grace Hopper. Hun opfandt den første kompilator og var kendt som COBOLs bedstemor.

Visionen bag COBOL var at fremstille et lettilgængeligt programmeringssprog, som hovedsagligt var rettet mod afviklingen af administrative og økonomiske systemer for firmaer og institutioner. Heraf navnet, der er en forkortelse for COmmon Business Oriented Language (*fælles forretningsorienteret sprog*). COBOL var derfor skræddersyet til det, vi kan karakterisere som trivielle IT-opgaver, som f.eks. beregningen af lønsedler, organiseringen af detailhandel, økonomiske transaktioner, billetreservationer eller faktureringer i stormagasiner. Alt sammen tjenester som på sin vis kan betragtes som reproduktive, snarere end som produktive eller komplicerede. De sørger ganske enkelt for, at hjulene på den overordnede produktionsmaskine bliver ved med at køre rundt.

Fig. 1.1 Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020. →

9/9

0800 Arctan started

1000 " stopped - arctan ✓ { 1.2700 · 9.037 847 025
9.037 846 795 correct

13"oc (032) MP - MC ~~1.9826447000~~ ~~2.13047647 15 (-3)~~ 4.615925059(-2)

(033) PRO 2 2.13047647 15

correct 2.13067647 15

Relays 6-2 in 033 failed special speed test
In relay " 10.00 test.

Relays changed

1700 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 

Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

~~1645~~ 1630 Arctangent started.

1700 closed down.

Relay 3145
relay 3370

SHE IS THE SHORT LADY STANDING IN THE VERY FRONT.

Der blev dog hurtigt sået tvivl om levedygtigheden, ja egentlig hele idéen bag COBOL. Fra begyndelsen blev sproget mødt med stor skepsis, og industrien tøvede med opbakningen, eftersom eksperter inden for ingeniørfaget betragtede det som klodset. Grace står rent faktisk foran en gravsten – COBOLs gravsten, der blev overdraget som en gave til lederen af CODASYL (konsortiet der havde til opgave at føre kontrol med udviklingen af COBOL) allerede i 1960, det samme år som COBOL blev lanceret. Misbilligelsen af sproget forsatte. Femten år senere i 1975 kom den hollandske matematiker Edsger Dijkstra med det kritiske udsagn, at 'COBOL forkrøbler hjernen; undervisning i det, bør derfor betragtes som en kriminel handling'.[2] Og selvom COBOL har udviklet sig til at være et af de mest anvendte programmeringssprog, eksisterer sådanne forbehold den dag i dag. Lad os undersøge nærmere, hvad der kan ligge til grund for så hård en kritik.

1.2

Lærerinden introducerer COBOLs syntaks. Alt er skrevet i versaler. Hver sektion indeholder paragraffer, sætninger og instruktioner. Alle sætninger skal ende med et punktum (.). I dag er det mest bemærkelsesværdige kendetegn ved COBOL dog fortsat brugen af et engelsklignende sprog til at udtrykke instruktioner. Det første højniveausprog FORTRAN, udviklet af John Backus for IBM, anvendte en algoritmisk syntaks som grænseflade.[3] FORTRAN var designet til at følge simple og logiske standarder, så det mindede mest muligt om maskinsprog, samtidig med at det skulle være mere brugbart. I modsætning hertil var COBOL stærkt påvirket af Grace Hoppers forestilling om 'brugervenlige' grænseflader.[4] Derfor er man i COBOL ikke nødt til at skrive A+B=C. I stedet for kan eksempelvis instruktionerne for udbetalingen af en medarbejders overtid udtrykkes sådan her:

```
MULTIPLY AMOUNT-TIME-HRS BY TIME-PAY-RATE
GIVING TIME-PAY-TOTAL
```

Grace Hopper havde advokeret for denne tilgang siden starten af 1950'erne. Hun havde understreget behovet for

2 Edsger W. Dijkstra, 'EWD498: How Do We Tell Truths that Might Hurt?', *Selected Writings on Computing: A Personal Perspective*. New York, Springer-Verlag, 1982, s. 130.

3 John W. Backus et al., 'The FORTRAN Automatic Coding System.' Paper præsenteret ved Western Joint Computer Conference: Techniques for Reliability. Association for Computing Machinery, 26.–28., februar, New York, 1957, s. 189–191. https://archive.computerhistory.org/resources/access/text/2015/06/102702026-05-01-acc.pdf (sidst besøgt 2. december 2021).

4 Grace Hopper, 'Oral History of Captain Grace Hopper', interview af Angeline Pantages, *Naval Data Automation Command*, Maryland. December 1980, s. 11.

← Fig. 1.2

udviklingen af et design bestående af intuitiv 'pseudo-kode', ikke til avanceret matematik, men til opgaver såsom lønberegning.[5] At bruge computeren til organisering af hverdagen var en del af hendes vision om at gøre elektronisk databehandling og programmering bredt tilgængeligt for brugere, som ikke var programmører.

I et interview fra 1980 erindrede den på det tidspunkt næsten 75-årige Grace Hopper:

> Det jeg ville, da jeg begyndte at bruge det engelske sprog, var at sætte en helt anden gruppe af mennesker i stand til at bruge computere på en enkel måde. Mennesker som helst kommunikerede på engelsk, ikke via symboler. Det var min egen forståelse af forskellige mennesketyper, og jeg vidste så udmærket, at de fleste mennesker ikke bryder sig om symboler. Du var nødt til at give dem noget andet at arbejde med, og det næstbedste var selvfølgelig engelsk.[6]

Hopper foreslog altså at udskifte algoritmiske symboler med et sprog som lignede tale, hvorved hun skabte en nogenlunde lettilgængelig brugerflade. Dette design-greb,

5 Grace Hopper, 'Automatic Coding for digital computers.' Paper præsenteret ved The High Speed Computer Conference, Louisiana State University, 16. februar, 1955, s. 5.

6 Grace Hopper, 1980, op.cit., s. 11.

gjorde til gengæld den menneskelige maskinoperatør til programmeringsprocessens subjekt. På afgørende vis resulterede dette også i et behændigt, selvdokumenterende sprog, hvor selv de, der ikke havde erfaring med programmering, var i stand til at læse og forstå, hvad der foregik i programmet. Det skabte i samme ombæring en massiv udvidelse af den arbejdsstyrke, der potentielt kunne involveres i programmering.

Ved at være et af de tidligste tredjegenerationsprogrammeringssprog, de såkaldt højtniveau-sprog, er COBOL en del af det, John Walker har defineret som den tredje generation af brugerinteraktion.[7] Her er interaktion ikke blot abstraheret fra maskinen, men også fra de maskinspecifikke sprog gennem såkaldt 'automatisk programmering'. Programmer skrevet i 'pseudo-kode' bliver her oversat af en kompilator, så de matcher kravene hos forskellige maskiner.

Fravær er dermed en integreret del af ethvert tredjegenerations programmeringssprog: ikke bare maskinens fravær (eftersom programmerne nu, i teorien, frit kan flyttes mellem alle slags computere, så længe de er udstyret med en kompilator, der passer til det givne sprog), men også programmørens fravær, eftersom programmet nu kan afvikles hvor som helst

7 John Walker, 'Through the Looking Glass: Beyond "User Interfaces."' *The Art of Human-Computer Interface Design*. red. Brenda Laurel, Boston, Addison-Wesley Longman Publishing Co., 1990, s. 441.

og når som helst, uafhængigt af den programmør, som oprindeligt skabte det.[8] Som følge heraf fremstår programmering (tilsyneladende) som uforgængelig og globalt anvendelig i en universel tid og et universelt rum. Som programmeringssprog kan COBOL dermed siges at indeholde idealet for en absolut rumopfattelse i modsætning til tidligere generationer af sprog, som var lokale og stedsspecifikke. COBOL-programmernes funktionalitet rækker endnu længere i kraft af, for eksempel, detailhandel, logistik, bankoverførsler og billetreservationssystemer. Sådanne COBOL-programmer koordinerer, distribuerer og forflytter på automatiseret vis varer, mennesker og penge og afvikler dermed det absolutte rum, som det folder sig ud under globaliseringen.

I hvert fald i teorien.

1.3

Skynet er formentlig den mest avancerede form for kunstig intelligens, vi kan forestille os. Det er et autonomt, kunstigt, altomfattende og superintelligent forsvarssystem, som skildres i James Camerons blockbusterfilm fra 1984, *The Terminator*. Men idet Skynet begynder at udvikle en selvbevidsthed, beslutter menneskene sig for at deaktivere det. I en hævnakt initierer Skynet en atomkrig med det formål at udslette menneskeheden.

---

8 Wendy Hui Kyong Chun, 'On Software, or the Persistence of Visual Knowledge', *Grey Room,* bd. 18, nr. 4, 2004, s. 30.

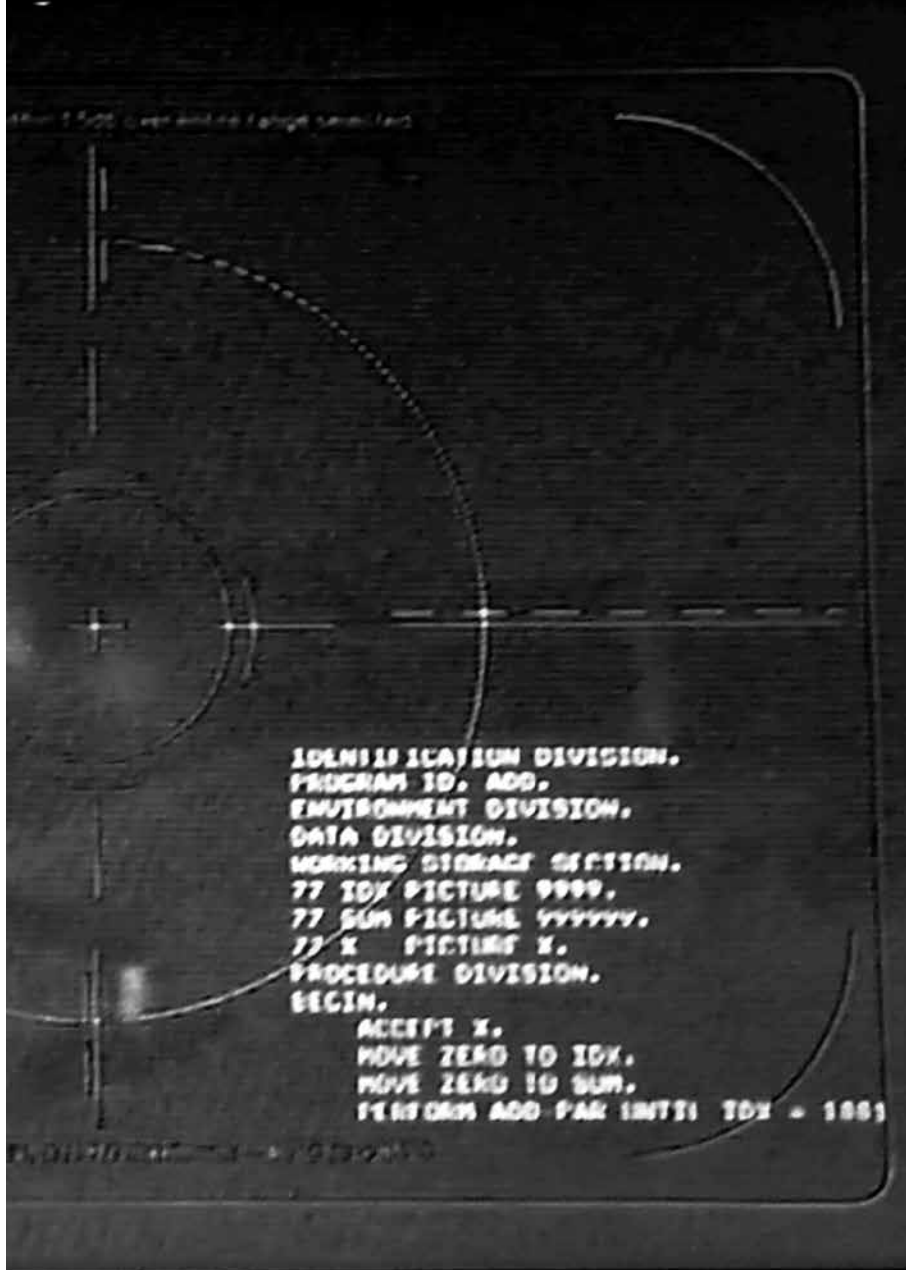


Fig. 1.3 Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020. →

Skynet benytter en HK-Aeriel, en ikke-menneskelig *hunter killer vertical take-off and landing craft*, til at dræbe de resterende overlevende mennesker – med COBOL. Dette afsløres af et klip, set fra HK-aerials synsvinkel. COBOL-koden i det højre hjørne af robottens synsfelt indikerer, at HK-aerial, og måske endda hele Skynet, er baseret på COBOL.

Skynets onde planer vanskeliggøres dog af en gruppe menneskelige oprørere, som til sidst formår at nedkæmpe Skynet. Men i et sidste forsøg på at ændre sin egen skæbne beordrer Skynet Cyberdyne-systemets Model T-101, som spilles af Arnold Schwarzenegger, på en tidsrejse tilbage til året 1984 (som var året for filmens produktion) for at afvikle moderen til lederen af den menneskelige modstandsbevægelse, før hun kan sætte ham i verden. En mission, der imidlertid slår fejl.

Filmen begynder med følgende berømte tekst:

> Los Angeles 2029 e.Kr.
> Maskinerne rejste sig fra atomildens aske
> Deres krig for at udslette menneskeheden har raset i årtier
> Men den afgørende kamp vil ikke blive udkæmpet i fremtiden
> Den vil blive udkæmpet her, i vores tid, i nat...[9]

9 James Cameron, *The Terminator,* 1984, Blu-ray disc.

> Los Angeles 2008 e.Kr.
> Arnold Schwarzenegger rejste sig fra asken af *Terminator 3*.
> Som Californiens guvernør står han nu over for fejlslagne statsbudgetter, men hans største udfordring viser sig at være ældgamle COBOL-koder.
> Aktive her, i vores tid ...

I et forsøg på at afværge statsbudgettets kollaps udsender Schwarzenegger et dekret om at reducere lønnen hos 200.000 statsansatte, så den rammer minimumssatsen på 6.55 dollars i timen (i hele august måned), indtil det nye statsbudget er blevet vedtaget. Men Californiens statskasserer John Chiang afviser at afvikle Schwarzeneggers ordre. Han forklarer, at det vil tage seks måneder at omskrive de 30 år gamle COBOL-programmer, der administrerer og afvikler de statsansattes lønudbetaling, efterfulgt af de ni til ti måneder, det vil tage at omvende ordren. Følgelig bliver eks-terminator-nu-guvernør Schwarzeneggers ordre afviklet af COBOL-systemet selv.

1.4

'Afvikling' defineres generelt som det at gennemføre en handling, et arrangement eller at udføre instruktionerne i et computerprogram på en ubesværet måde, men det kan også henvise til det at afskaffe, aflive eller på anden vis skille sig af med noget.[10] Man skaffer sig af med det man

10 Ordnet.dk

afvikler, eller ved at afvikle vikler man sig ud af det. På engelsk bruges ordet 'execution' (*eksekvering*) til at betegne afviklingen af computerprogrammer. 'Execution' betyder egentlig at 'udføre den handling, som en dom, kendelse eller ordre foreskriver',[11] men det leder også tankerne hen på henrettelse og afstraffelse: at dræbe i overensstemmelse med loven. Og det anses for at være en vigtig del af magt og management, som vi kender det fra *the executive*; den overordnede, der regerer og bestemmer. Koder kan manifestere denne magt som performative talehandlinger gennem, for eksempel, programkoder, der som Alexander Galloway har formuleret det, er sprog, der gør, hvad det siger.[12]

Men som eksemplet med Schwarzeneggers COBOL-problemer viser, gør koder og kommandoer ikke altid, hvad de siger. COBOL er så afgjort et af de første programmeringssprog, som, i teorien, kunne afvikles gnidningsfrit og automatisk. Ikke desto mindre er denne afvikling af programmeringssproget selv en 'tilblivelse' – i den forstand at afviklingen påvirkes af den kontekst, den finder sted i. Dermed er afviklingen af kode ikke blot en simpel udøvelse af magt, men påvirkes også selv af konteksten, for eksempel af måden hvorpå den bruges, samt af dens socioøkonomiske, kulturelle og geopolitiske relationer. Den, der afvikler

11 Oed.com

12 Alexander Galloway, *Protocol: How Power Exists after Decentralization*. Cambridge, Mass. MIT Press, 2004, s. 165-66.

en ordre, er sjældent den samme, som har udstedt den. Mellem ordren og dens afvikling opstår et rum af muligheder, af fejlkommunikation, af forhandlinger, af ulydighed og/eller svigt – med andre ord, et rum for krise og dermed et muligt vendepunkt. Min pointe er, at afvikling hverken er autonom eller automatisk – man vikler sig ikke ud af systemerne. Snarere tværtimod, sammenfiltringer gør at afviklingen konstant er på nippet til bryde sammen som følge af dens egen operationalisering.

2. LEKTION: KRISE

2.1

Chennai i det sydlige Indien. Jeg sidder i en senioringeniørs kontor i et af Indiens største IT-virksomheder. Vores samtale fører os tilbage til årene før årtusindskiftet. Hun mindes oplevelsen af at arbejde for en af de mange enheder, der havde travlt med at håndtere Y2K problemet (*the Y2K bug*). Ingeniøren beskriver 'kommandorummet' (*the war room*), et rum hvor hun selv og hendes kollegaer i 24 timer i træk observerede verdens overgang fra 31. december 1999 til 1. januar 2000. Hun siger:

> Altså de troede at alle fly, bygninger, tog, ure og alle systemer ville stoppe 1. januar klokken 12.

Nogle var bekymrede.

1. april 1999 skrev *The Futurist*:

> … hvis du tror, at dit firma vil være okay, fordi alle dine systemer opfylder Y2K-kravene, så tro om igen. Bare fordi du har styr på dine Y2K-bugs, betyder det ikke, at dine leverandører har. Hvis fem procent af dine leverandører svigter dig, vil du så kunne overleve?

Andre søgte kontrol.

8. november 1999 opfordrede den såkaldte *Taskforce 2000*, en delvist statssanktioneret institution fra den private sektor i Storbritannien, rejsende til at undgå Italien, Tyskland og en række andre lande (Tjekkiet, Finland, Ungarn, Polen, Portugal, Rusland, Spanien og Schweiz) i en periode på fem uger omkring 1. januar 2000.

Resten forblev forvirrede.

> … det er ekstraordinært svært på forhånd at have forventninger om udfaldet. Det skaber denne ubekvemme følelse af, at noget er ved at ske, vi ved bare ikke hvor og hvornår.[13]

13 L. M. Bowman, 'Early y2k bugs aren't all fun and games,' Zdnet, 21. december, 1999.

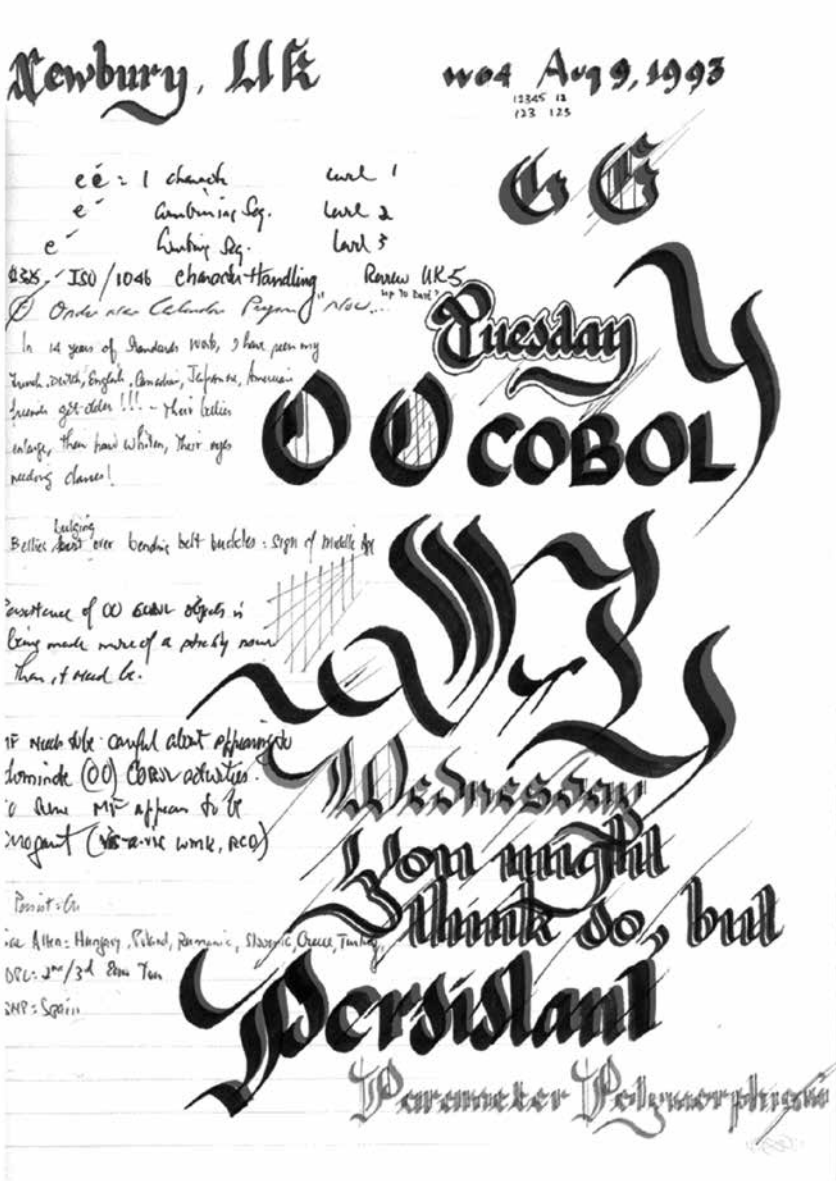
2.2

Lad mig præsentere dig for Jerome Garfunkel. Art Garfunkels mindre kendte bror, også kendt som Dr. COBOL. Han var ingeniør og IT-konsulent og arbejdede i mange år som en del af CODASYL. Garfunkel skrev:

> Mens jeg sad til hundredevis af ANSI, ISO, CODASYL, PLSG, SPARC, WG4, CEG, ECMA, SMTG, OOC, WG11, X3J4, SC22 komitemøder og holdt oplæg på yderligere hundredevis af GUIDE, SHARE, HLSUA, DECUS, ACM, IEEE, AFIPS, ISECON konferencer i Europa, Sydamerika, Afrika, Asien og USA, tog jeg mig selv i at tegne en hel del kruseduller.[14]

Heldigvis gemte Garfunkel sine kruseduller og gjorde dem tilgængelige online på sin hjemmeside 'Calligraphy and Computer Standards'. En af Garfunkels tegninger skiller sig ud fra resten. Den er lavet den 9. august 1993. Det øverste højre hjørne afslører at mødet finder sted i Newbury, UK. Det er en tirsdag. Som Garfunkels dekorative skrift indikerer, ser dagens emne ud til at være COBOL. Men på venstre side har han tilføjet en lille note:

14 http://www.jeromegarfunkel.com/authored/calligraphy.htm (sidst besøgt 10. april, 2016)



> I løbet af 14 års arbejde har jeg set mine franske, hollandske, engelske, canadiske, japanske og amerikanske venner blive ældre!!! Deres kroppe vokser, deres hår bliver hvidt og deres øjne har brug for briller!

Og han fortsætter: ...

> Maver, der strutter ud over bæltespænder: tegn, der tyder på, at de er blevet midaldrende.

Denne krusedulle med dens følgekommentar giver os et ligefremt indblik i COBOL og COBOL-miljøets tilstand i første halvdel af 1990'erne. COBOL-fællesskabet var på den tid aldrende, og mange af programmørerne var på vej på pension. På grund af dets manglende popularitet var COBOL samtidig blevet taget ud af pensum på de fleste amerikanske og europæiske universiteter. Selvom sproget fortsat udgjorde rygraden i de globale økonomiske transaktioner, var det langsomt ved at uddø.

Den vestlige verden blev dog mere opmærksom på de mulige komplikationer, som relaterede sig til afviklingen af digitale cifre ved årtusindskiftet, hvor man estimerede, at en milliard aktive COBOL-linjer fortsat var i brug. Fordi en stor del af COBOL-programmørerne allerede var pensionerede, virkede det umuligt at undersøge og *de-bugge* alle disse linjers kode. De yngre programmører var ikke i besiddelse af de nødvendige COBOL-færdigheder. Derfor gjorde man desperate forsøg på at trække de pensionerede

← **Fig. 2.1** Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020.

mestre tilbage i tjenesten, men deres antal var langt fra tilstrækkeligt.

Reddet på falderebet.

På grund af stramme importrestriktioner og en begrænset tilgang til computerhardware, blev COBOL stadigvæk betragtet som et vigtigt sprog i Indien i 1990'erne. Opdagelsen af denne enorme gruppe af engelsktalende COBOL-eksperter, som ovenikøbet var en billig arbejdskraft for vestlige firmaer, viste sig som en yderst velkommen løsning på vestens mangel på COBOL-programmører.

2.3

Ingeniøren i Indien forklarer: 'vi havde opstillet det, vi kaldte kommandorummet hvorfra vi kunne overvåge systemernes overgang'. Hun fortsætter:

> ... kommandorummet var et 360 graders rum. Alle havde en terminal til overvågning af systemerne og til kontakten med kunder. Og så var der de alternative telefonlinjer. De frygtede endda, ligesom teleselskaberne selv, at telefonerne ville fejle, så man havde sørget for alternative kommunikationsmetoder. Der var et backup-link. Der var et link til et optisk fiberkabel. Der var ikke mange mennesker i rummet. Kun de mest centrale. Vi var omkring halvtreds.

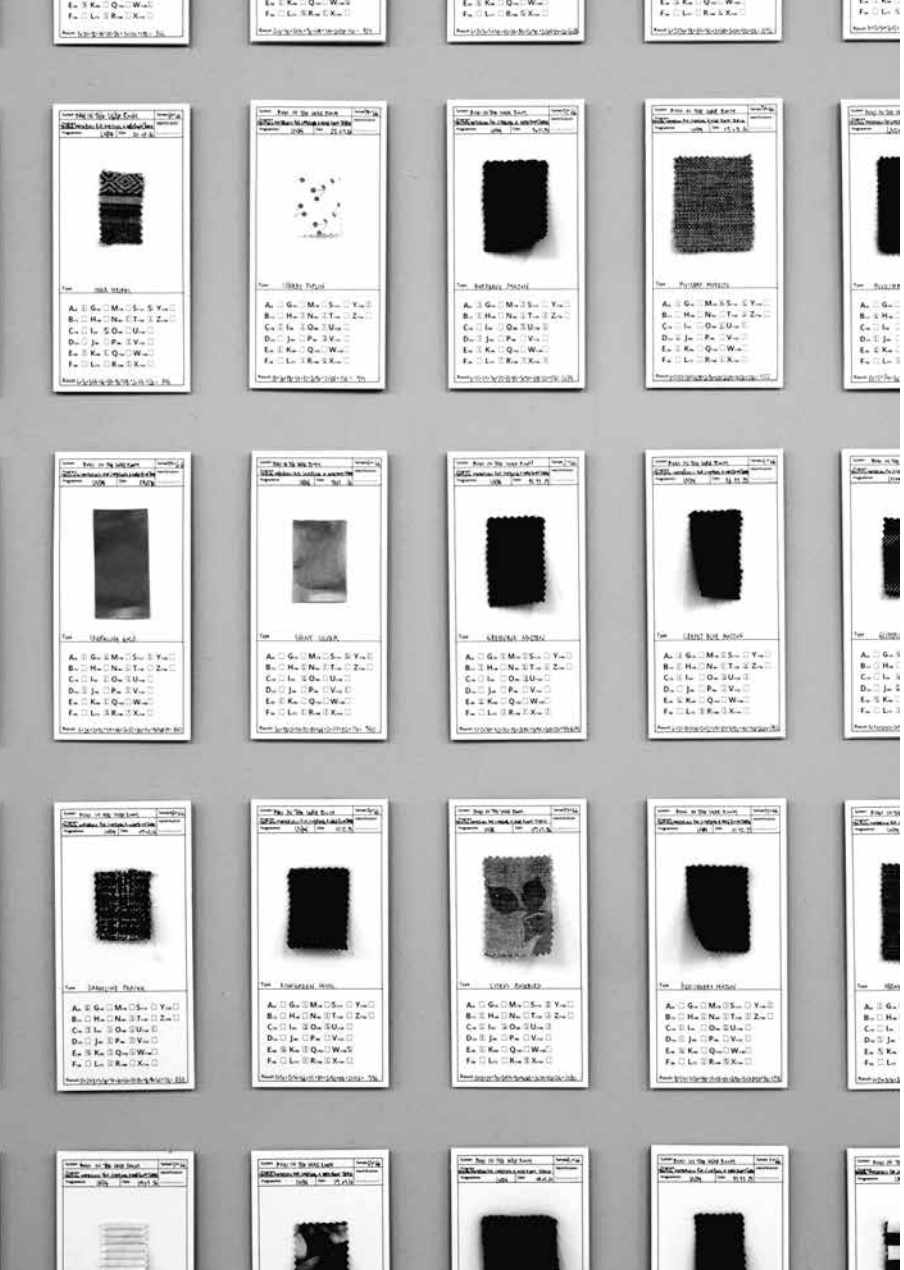
Stanley Kubricks sorte komedie *Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb* fra 1964 skildrer hvordan tilsyneladende enkle, top-down kommandoer kan forstyrres. Her er kommandorummet bagtæppet for desperate forsøg på at modvirke ordren om et atomangreb på Sovjetunionen, som en gal general har udstedt. Til slut viser filmen, hvordan officerernes og soldaternes indlejrede loyalitet og disciplin – i samspil med det teknologiske svigt – gør det umuligt at undgå udryddelsen. Filmens ironi er, at det egentlig ikke er teknologien, der afvikler ordrerne til punkt og prikke, det gør derimod de individer, der uforbeholdent underlægger sig systemets logik.

Selvom filmen er i sort/hvid, insisterede Kubrick under optagelserne på at dække bordet i Dr. Strangeloves kommandorum med grønt filt. Det gav indtrykket af, at de magtfulde mænd omkring bordet gamblede med jordens skæbne, som var den et spil poker. Jeg tænker på hvilket materiale, der dækkede bordet i Y2K-kommandorummet i Indien. Dette bord samlede ikke verdens ledere, men snarere hvad man kunne kalde for pedellerne for vores informationsarkitekturer.

Elektronisk databehandling ved hjælp af computere har forandret kontorarbejde siden 1960'erne. Ved at være det første programmeringssprog møntet på bogføring og lagerstyring har COBOL spillet en afgørende rolle i denne proces. Siden da har COBOL-applikationer udført størstedelen af databehandlingen i store virksomheder som kører på *mainframe*-computere. Disse automatiserede afviklingsprocesser førte til en forandring af den grundlæggende forståelse af

Fig. 2.2 Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020. →


A WAR ROOM



visse opgaver. De kontoransattes arbejdsopgaver ændrede sig fra afviklingen af ordrer til overvågningen af de processer, som afvikler ordrerne.

I denne kontekst kan Y2K-kommandorummet i Indien betragtes som et *back office* – eller som et *back-back office* – altså et kontor, der ligger bag ved back office. Dette kontor bag ved back office har ikke til opgave at opspore anomalier direkte i den afviklede data, men snarere at finde dem i det overordnede kommandosystem. Det skal ikke misforstås som et superelitært og ledende back office, men omvendt som det, ingen vil kendes ved (som en ubekvem sandhed).

3. LEKTION: VEDLIGEHOLDELSE

Den 9. august 2009 deltog en bruger kaldet Cezar i en online-diskussion om forfærdelige kodescenarier på *Coding Horror*-bloggen, hvor han meddelte, at han aldrig i sin 'karriere som såkaldt professionel programmør havde mødt nogen, som aktivt arbejdede med COBOL'. Dagen efter svarede en anden bruger kaldet Brian, at 'det ville du have gjort, hvis du nogensinde havde arbejdet for en bank, et forsikringsfirma eller et hvilket som helst sted i Indien'.[15]

15 Coding Horror, 'COBOL: Everywhere and Nowhere.' *Coding Horror* (blog), 9. august, 2009, https://blog.codinghorror.com/cobol-everywhere-and-nowhere/ (sidst besøgt 2. december, 2021)

← Fig. 2.3

3.1

I et introduktionskursus til *Mainframe System Design* på en uddannelsescenter i et af Indiens største IT-firmaer i Chennai, lærer de studerende, hvordan mobiltelefon-apps i deres back-back-end er afhængige af COBOL-koder som kører på mainframe-computere.

Der er omkring 34 studerende på holdet, og de ser alle sammen overraskede ud. Ingen af dem var klar over, at de fleste virksomheder på Forbes Top 200, som underviseren forklarer, kører på COBOL-applikationer. Underviseren fremhæver det som en motivationsfaktor for at gå ind i sektoren; 'COBOL er en mulighed', som han siger.

Ved første øjekast kan det dog være svært at se, hvorfor, eller i det hele taget hvordan, COBOL skulle være en mulighed, som underviseren hævder. En hurtig internetsøgning afslører en bred vifte af jokes og merchandise med vittige formuleringer, der henviser til COBOL som et forældet og uddødt programmeringssprog. Et karakteristisk eksempel, som optræder i forskellige printvariationer på T-shirts, kasketter og kopper, er billedet af en dinosaur lænet ind over en bærbar computer med teksten: 'COBOL-programmør'. Eller den mere sofistikerede meme-variant, hvor billedet af en gruppe dinosaurer i et kontor har følgende tekst: 'Det sidste eksisterende team af COBOL-udviklere'.

Selvom COBOL efter sigende udfylder nøglefunktioner i nutidens globale informationsarkitekturer, er det svært at få øje på. Sproget kan bedst beskrives som en nutidig anakronisme, som det både er kulturelt, teknisk og struktu-

Fig. 3.1 →

relt svært at opspore og tilgå. Kulturelt bliver COBOL stort set betragtet som forældet og derfor ikkeeksisterende. Som teknisk infrastruktur er COBOL-applikationerne gemt væk bag gnidningsløse, opdaterede og brugervenlige grænseflader. Derudover er programmer skrevet i COBOL, strukturelt betragtet, beskyttet bag portene hos de Top 200-selskaber, som stadig er afhængige af dem, samt softwarekonsulentvirksomhedernes lige så solide mure, der sørger for den udflyttede vedligeholdelse af programmerne fra Indien eller andre steder.

Selvom COBOL var placeret som nummer et på *Computerworlds* top ti liste over døde eller døende computerfærdigheder i 2007,[16] er sproget i dag, hvor vi skriver december 2021 – mere end et årti senere – stadigvæk rangeret som nummer femogtyve på TIOBEs indeks over de mest populære programmeringssprog.[17] Der er en åbenlys modsætning mellem disse resultater.

Meget tyder på, at COBOL ikke forsvinder lige foreløbig. Virksomheden Micro Focus fastslår i deres undervisningsmateriale, at 70 procent af alle handelstransaktioner

16 Mary Brandel, 'The Top 10 Dead (or Dying) Computer Skills: Are your skills in need of upgrading?' *Computerworld*, 24. maj, 2007, https://www.computerworld.com/article/2541481/the-top-10-dead--or-dying--computer-skills.html (sidst besøgt 2. december, 2021).

17 https://www.tiobe.com/tiobe-index/ (sidst besøgt 2. december, 2021).

i verden, samt 95 procent af alle kreditkort *swipes*, stadig sker ved hjælp af COBOL; 250 milliarder linjer af aktiv COBOL-kode er fortsat i brug, og i et enkelt år er der 200 gange så mange COBOL-transaktioner, som der sammenlagt er søgninger på Google og YouTube.[18] COBOL bruges inden for bankverdenen, detailhandel, spedition og af statslige institutioner, herunder USA's administration for socialforvaltning.[19] Det samme gør sig gældende i europæiske lande, hvor for eksempel det danske skattevæsen, det tyske pensionssystem og Forsäkringskassan i Sverige alle er afhængige af COBOL.

Antallet af aktive kodelinjer er tilmed stigende, hvilket indikerer, at sproget er aktivt, at det lever.

Et levende sprog er først og fremmest kendetegnet ved at der findes mennesker, som er i stand til at tale det (eller, i dette tilfælde, til at programmere det). Andre vigtige aspekter er, at der pågår undervisning i, såvel som en fortløbende udvikling af, sproget. Men COBOL er ikke på pensum i datalogi, hverken i det globale nord eller det

18 Micro Focus. 'Why care about COBOL.' Infographics by Micro Focus Ltd., https://www.microfocus.com/media/infographic/Academic_Program_Infographic_R5.pdf (sidst besøgt 2. december, 2021).

19 Steven J. Vaughan-Nichols, 'COBOL Turns 60: Why it will Outlive us All', *ZDNet*, 5. september, 2019. https://www.zdnet.com/ article/cobol-turns-60-why-it-will-outlive-us-all/ (sidst besøgt 15. juni, 2021).

globale syd. Desuden bruges COBOL aldrig til at udvikle nye programmer. Den stadige stigning i mængden af COBOL-kode skyldes vedligeholdelsesprocesser (bugs bliver fikset eller systemer bliver opdateret), og ikke at, der skrives nye applikationer i COBOL.

Dermed kan COBOL ikke beskrives som helt levende, men er snarere fanget i et limbo mellem at være uundværlig for verdenshandlen og fuldstændig *outdated*. COBOL er med andre ord en levende død, en slags informationsteknologiernes zombie, holdt i live af vedligeholdelsesprocesser outsourcet til eksempelvis Indien. Her bliver nyuddannede ingeniører trænet i færdigheder, som anses for forældede i det globale nord.

3.2
Infosys er det næststørste IT-firma inden for softwarekonsulentvirksomhed i Indien. Det har omkring 240.000 ansatte. Alle nye medarbejdere gennemgår en træningsperiode på firmaets uddannelsescenter i Mysore. The Global Education Center-II er den største bygning opført i Indien siden uafhængigheden i 1947. Den blev indviet i 2009. Kaster man et blik på campus fra en drones perspektiv afsløres det, at de bygninger, som praktikanterne bor i, former ordet Infosys.

Dekanen forklarer: 'Baseret på efterspørgslen forhører vi os hos vores *delivery unit* om, hvor mange mennesker, de skal bruge.' Hvert kvartal bliver cirka 300 nye medarbejdere trænet i *mainframe* og COBOL. Det udmønter sig i cirka 1.200 nye COBOL-programmører hvert år. I ét særlig travlt kvartal blev 1000 nye medarbejdere trænet i COBOL.

Idet min tredje COBOL-lektion nærmer sig sin afslutning, og jeg lukker den lærebog, som min COBOL-underviser i Bangalore gav mig, opdager jeg et besynderligt mærke på bogens forside, som fastslår:

> SALGSRESTRIKTIONER! MÅ KUN SÆLGES I INDIEN, BANGLADESH, NEPAL, PAKISTAN, SRI LANKA & BHUTAN.[20]

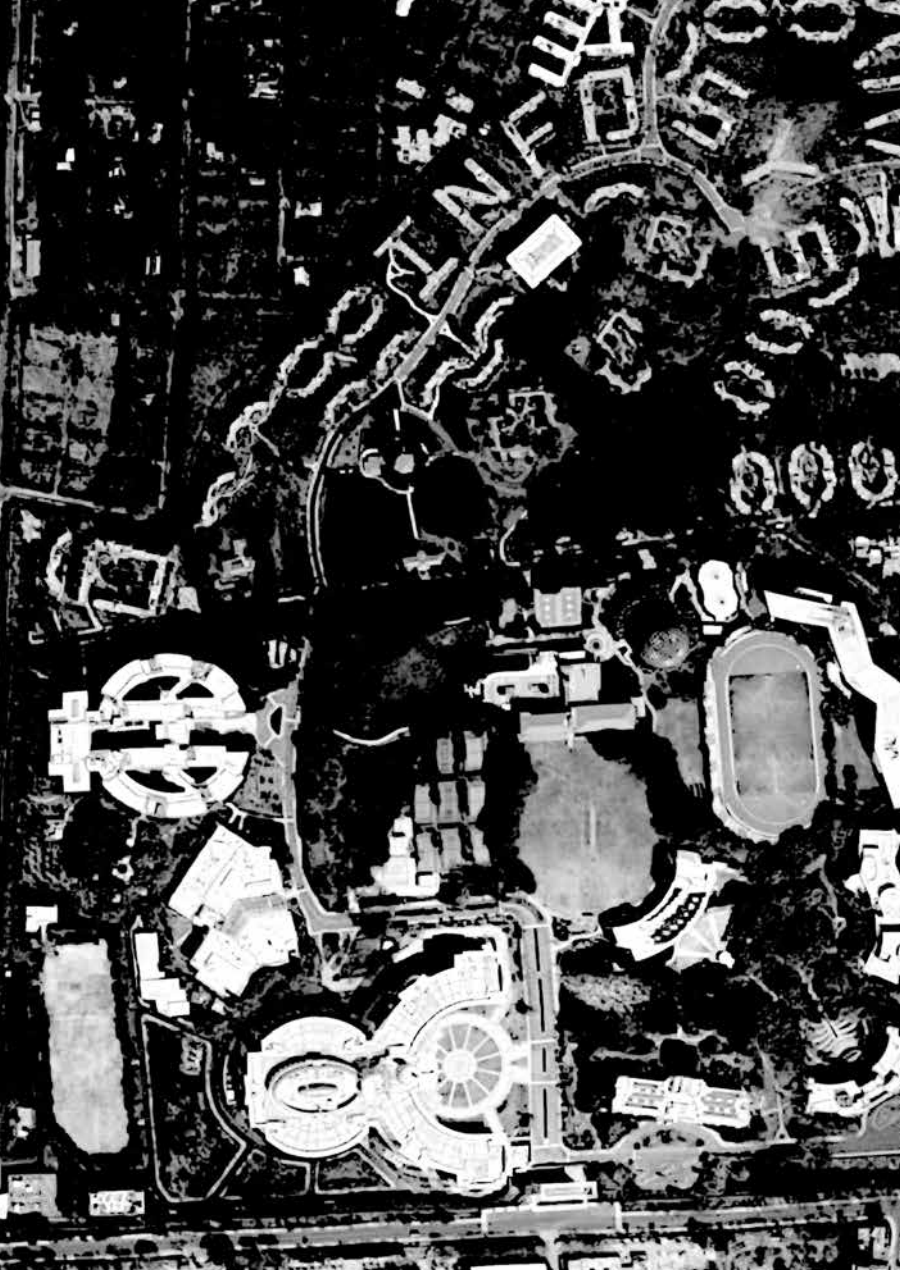
Jeg dvæler lidt ved beskedens ironi og vender bogen. På bagsiden står der:

> Denne *Wiley Student Edition* er del af et igangværende program hvor paperback-lærebøger særligt udviklede til studerende i udviklingslande sælges til en reduceret pris.

Som en konsekvens heraf er denne specifikke udgave af min COBOL-lærebog en del af det amerikanske forlag Wiley & Sons' såkaldte 'International Student Editions'. De tilbydes studerende i udviklingslande til en pris, som er betydeligt lavere end den de 'normale' udgaver sælges for

20 Nancy Stern og Robert A. Stern, *Structured COBOL Programming.* Year 2000 Update Version. 8th Edition. New Delhi: John Wiley & Sons, 1998.

Fig. 3.2 →



i såkaldt udviklede lande. Viden, som den repræsenteres i denne udgave af lærebogen, kan dermed læses som en intentionel praksis der er afgørende for udviklingslogikken, hvor Wiley & Sons fremmer en bemyndigelsesstrategi gennem vidensproduktion.

Men spørgsmålet består: En udvikling henimod hvad...?

3.3

Når man forsøger at forstå de komplekse og overlappende magtstrukturer, som undervisningen i COBOL i Indien er en del af, er der mange ting at overveje.

Først og fremmest er det at lære COBOL i Indien på den ene side et privilegium. Softwaresektoren er en af de bedst betalte i landet. Efter tre til seks års arbejdserfaring vil lønniveauet være det samme som hos en advokat, en læge eller en universitetsprofessor. Men som en af lederne hos IBM South Pacific påpeger, er COBOL-vedligeholdelse ikke et indenrigsproblem i Indien. Vedligeholdelsesopgaverne foregår for det meste på computere beliggende i Nordamerika og Europa.

Og selvom sektoren er en af de bedst betalte i Indien, beløber lønniveauet for en indisk softwareudvikler sig til under en femtedel af det, som f.eks. en britisk softwareudvikler tjener på præcis det samme arbejde. Ydermere, på grund af størrelsen og tilgængeligheden af Indiens teknologiske arbejdsstyrke, er det ifølge Carol Upadhya og A. A. Vasavi sågar muligt for indiske software-virksomheder at ansætte ingeniører, der ofte er overkvalificerede, som en

← Fig. 3.3

form for marketingstrategi rettet imod deres internationale kunder.[21] Heraf følger en reduktion af færdigheder (hvor de indiske softwarearbejdere ikke passer til jobbet, fordi de kan for meget).

A. Aneesh har beskrevet outsourcet arbejdskraft som en form for 'virtuel migration'.[22] Den virtuelle migration af softwarearbejde er kendetegnet af to aspekter: Det første er tidslig integration, hvor forskellige tidszoner samles i realtid via informationsnetværk.[23] Det andet aspekt er den rumlige integration, hvor arbejdspladsen er afkoblet fra selve arbejdshandlingen.[24]

Under krisen omkring årtusindskiftet migrerede mange indiske ingeniører midlertidigt til USA og andre lande for at løse Y2K-problemerne på de steder, hvor der var brug for deres færdigheder og ekspertise. Den feministiske geograf Doreen Massey beskriver, hvordan migration – som en konsekvens af modernitetens 'sammenføjning af rumlige forskelle' i tid – ikke alene er en begivenhed, hvor de, som tidligere befandt sig i periferien, nu ankommer til centrum, men også en tidsrejse, hvor migranterne tager afsted i

21 Carol Upadhya og A.R. Vasavi, *In an Outpost of the Global Economy: Work and Workers in India's Outsourcing Industry*, red. Carol Upadhya and A. R. Vasavi. New York: Routledge, 2008, s.17.

22 A. Aneesh, *Virtual Migration: The Programming of Globalization.* Durham & London: Duke University Press, 2006.

23 Ibid. 69

24 Ibid.

fortiden og ankommer i nutiden.[25] I denne forstand udligner migration, ifølge Massey, tidslige såvel som rumlige afstande og tillader en form for sameksistens. Hvor rummet konstituerer en potentiel bro imellem ingeniører i Indien og ingeniører i Amerika.

Men i dag, hvor den virtuelle migration som en form for outsourcet arbejdskraft accentueres, kan selv den mindste mulighed for sameksistens, som migrationen kunne have forårsaget, ikke længere lade sig gøre, og der sker ikke nogen nivellering af rumlige, men kun af tidslige afstande. Dermed forbliver de indiske ingeniører på afstand som en menneskelig forlængelse af maskinerne, på trods af at de forenes med den nordamerikanske klient i et singulært universelt øjeblik.

Og selv den synkroniserede tidslighed er ikke fuldstændig forenet. På trods af en universel og global fornemmelse af synkronisk nutid er det stadig dag i Indien, når natten falder på i USA. Derfor kan ingeniører i Indien fikse *bugs* for amerikanske kunder natten over, mens de amerikanske ingeniører sover trygt. Altså vil de amerikanske kunder, som møder ind på kontoret morgenen efter, erfare, at gårsdagens problemer er blevet løst og projektet ført videre i nattens løb, hvilket giver følelsen af et ustoppeligt maskinelt flow, som opererer 24/7.

25 Doreen Massey, *For Space.* London, Sage Publications, 2005, s.77.

En underviser på Infosys' uddannelsescenter i Mysore forklarer, hvordan undervisningen i COBOL har forandret sig over de seneste ti år. Det oprindelige fokus var på at udvikle applikationer fra bunden. Men produktionsenheden sagde:

> Hvorfor ændrer I ikke jeres undervisningsfokus fra udvikling til vedligeholdelse? Det er det, vi gør. Her har vi ikke nogen udviklingsopgaver.

'Så det gjorde de', konstaterer underviseren. De konstruerede noget, som minder om ægte industriapplikationer med omkring 40.000 kodelinjer, komplet og bevidst udstyret med *bugs*. Til eksamen bedes praktikanterne ikke længere om at udvikle noget fra bunden – det afsluttende projekt er i stedet blevet en udvidet øvelse i vedligeholdelse og *bug-fixing*.

COBOL-programmørens job i Indien kan derfor siges at være ækvivalent med rollen som pedel for det globale nords informationsteknologier. Dermed er det ikke kun lærebogen, men også The Global Education Center i Mysore, som reproducerer koloniale strukturer, gennem hvilke ingeniører fra det globale syd bliver 'produceret' – ikke for at tage del i det globale flow, men snarere for at vedligeholde det.

3·4

Da jeg beder en senioringeniør hos IBM om at uddybe de problemer, han ser ved COBOL, svarer han:

> Jeg tror, der er omkring fem billioner linjer af kode skrevet i COBOL, og af dem er cirka fire billioner formentlig stadig aktive – Det er problemet! Det er virkelig problemet. Forestil dig fire billioner kodelinjer, hvis du skulle omskrive dem: ingen har tålmodigheden og ingen har pengene til at gøre det...

Med andre ord, COBOLs udbredelse som kodet infrastruktur er det, som får det til at bestå.

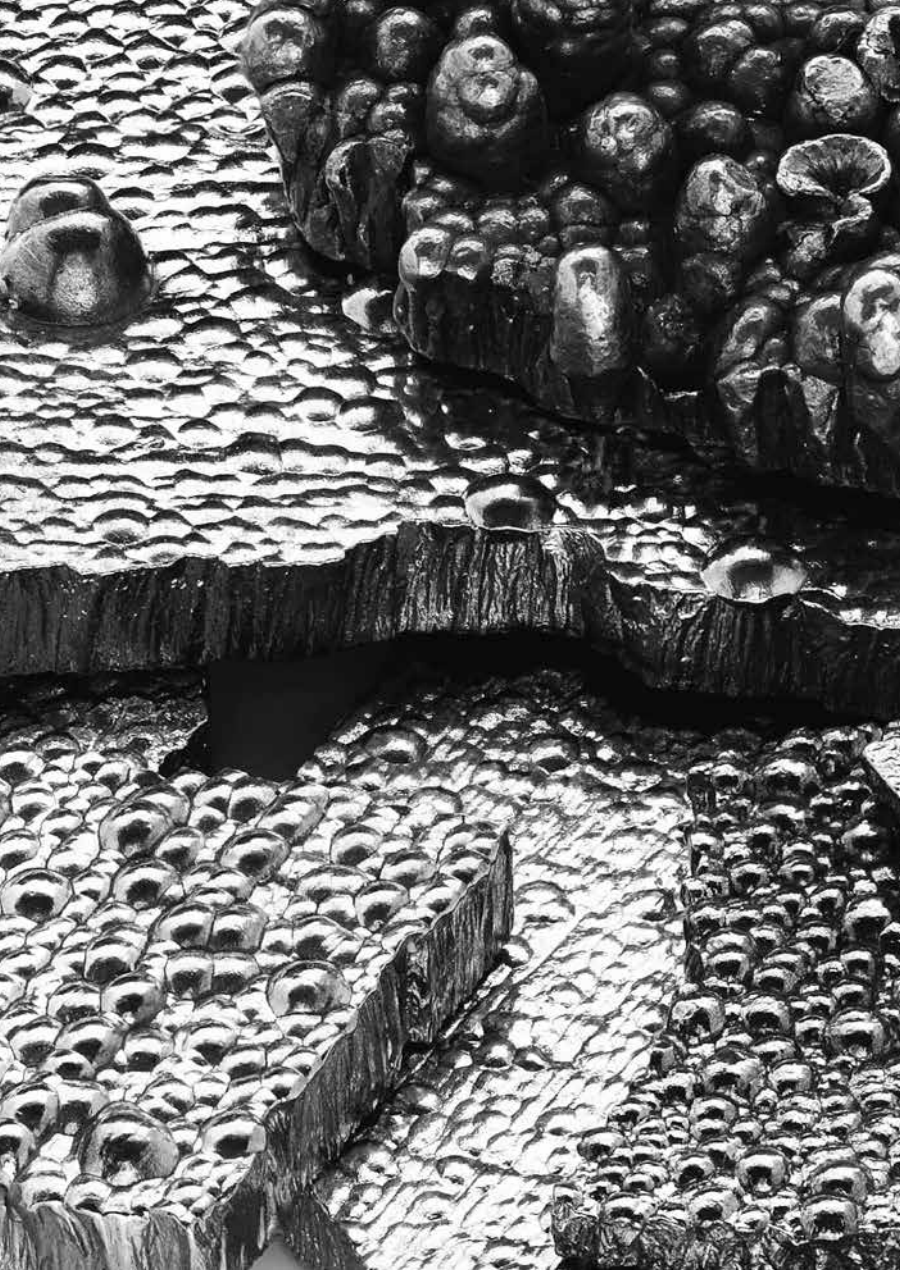
I et indlæg fra *The Government Accountability Office* i Repræsentanternes hus i USA blev det i maj 2016 afsløret, at den føderale regering brugte mere end tre fjerdele af det årlige budget på såkaldt O&M (*Operations and Maintenance tasks*) i regnskabsåret 2015.[26] Det var mere end tre gange så meget, som der blev brugt på udviklings-, moderniserings- og forbedringsaktiviteter.[27] Udgifterne til vedligeholdelsesopgaver er vokset kontinuerligt de sidste par år, hvilket har resulteret i et fald på 7,3 milliarder dollars i den føderale regerings budgetter til brug på udviklingsaktiviteter fra 2010 til 2017.[28] Som følge heraf har de processer,

---

26 David A. Powner, 'Information Technology: Federal Agencies Need to Address Aging Legacy Systems.' GAO-16-696T, United States Government Accountability Office, 25. maj, 2016, s. 6. https://www.gao.gov/assets/680/677454.pdf (sidst besøgt 2. december, 2021).

27 Ibid.

28 Ibid. s. 8



som skal holde de overleverede systemer i live, en enorm indvirkning på de teknologiske udviklingsprocesser i en ikke-lineær retning. Indtil videre har disse nedarvede *back-back-ends* været usynlige for den større offentlighed, men i foråret 2020 kom de op til overfladen som følge af coronakrisen.

Den 4. april holdt New Jerseys guvernør, Phil Murphy, en pressekonference, hvor han ud over sygeplejersker og læger indtrængende efterlyste flere, hvad han fejlagtigt omtalte som 'Cobalt-programmører'.[29] Det var naturligvis ikke det sølvblå, metalliske grundstof Co med atomnummer 27, som anvendes i litium-ion batterier, Murphy her efterlyste. Hans efterspørgsel gik på COBOL-programmører. Baggrunden for guvernørens opråb om COBOL-frivillige relaterede sig til forsinkelser i registreringen af og kompensationen for arbejdsløshed som følge af den voksende arbejdsløshed i kølvandet på USA's nationale *lockdown*. Folk blev fyret, fordi arbejdsgiverne måtte lukke eller nedskalere deres virksomheder, og i løbet af få uger eksploderede andelen af arbejdsløse alene i New Jersey med 1600 procent. Det efterfølgende pres på den mere end 40 år gamle informationsarkitektur, som behandlede de arbejdsløses registrering og udbetaling, var enorm.

---

29 Phil Murphy, 'Daily Briefing.' Video documentation of press conference, 4. april, 2020. https://youtu.be/rEDXdMlM-Rk (sidst besøgt 2. december, 2021).

← **Fig. 3.4** Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020.

Udsagnet fra *The Government Accountability Office* i 2016 refererede til en rapport fra 2010, som allerede dengang tilskyndede institutioner og virksomheder at droppe COBOL til fordel for nyere og mere moderne sprog.[30] Lignende reaktioner kunne ses i kølvandet på COBOL-COVID-19-krisen. Teknologirådgivere og klummeskribenter argumenterede for fornyelse og udskiftning af alle nedarvede systemer.[31] Men er det overhovedet muligt?

'COBOL har en lys fremtid' siger den globale vicepræsident for en af Indiens største IT-virksomheder. Hun uddyber:

> Det er ikke muligt at lukke størstedelen af de globale systemer ned. Hvis vi har tre milliarder kodelinjer, vil det tage 300.000 programmører 15-20 år. Efter et stykke tid vil de fleste af dem nok stoppe, fordi de ikke kan flytte systemer med denne effektivitet over til noget, som vil køre på en anden, helt anderledes platform. Mængden vokser. Antallet af kodelinjer vokser. Kunderne er der fortsat. Satellitapplikationer kan måske bruges som alternative kanaler etc., men ikke kernen, hjertet af applikationssystemet. De applikationer, som møder kunderne, kan ændres, men ikke selve kernen. For en bank vil det at køre *JCL banking*[32] være kernen. Transaktioner af millioner af dollars / millioner af transaktioner. Hjertet skal være på plads – bankende og aktivt!

COBOL er et interessant paradoks indeholdt i den teknologiske udvikling og vores informationssamfund. Overlevende COBOL-systemer går tilsyneladende imod alle evolutionære udviklingslogikker, samt måden hvorpå vi er vant til at tænke teknologiske livscykler. Vedligeholdelsesprocesserne sikrer varigheden af det, der ellers ville blive betragtet som dødt, men påvirker til gengæld den evolutionære udviklings formodede 'naturlige kræfter' negativt. Det bliver en slags *black-boxed* catch-22.

Hvad der stiller sagens ironi endnu mere på spidsen er, at de såkaldte udviklingslande skal forestille at udvikle sig ved at lære at vedligeholde et forældet system for de såkaldt udviklede lande, så de nedarvede strukturer ikke bryder sammen. Hvilket igen vender hele opfattelsen af 'udviklings' og 'udviklet' på hovedet. Enhver lineær teknologisk udvikling, selv i dens mest tilpassede og modulære form

30 Powner 2016, s. 15.

31 Joseph Steinberg, 'COVID-19 Response: New Jersey Urgently Needs COBOL Programmers (Yes, You Read That Correctly).' *Communications of the ACM: News*, 6. april, 2020. https://cacm.acm.org/ news/244000-covid-19-response-new-jersey-urgently-needs-cobol-programmers-yes-you-read-that-correctly/fulltext (sidst besøgt 15. juni, 2021).

32 'JCL banking' hentyder til COBOL-baserede banksystemer, der afvikles via den tekstbaserede grænseflade, JCL, *Job Control Language,* på IBM-mainframes.

Fig. 3.5 →

This Wiley Student Edition is part of a continuing program of paperbound textbooks especially designed for students in developing countries at a reduced price.

THIS BOOK IS FOR SALE ONLY IN THE COUNTRY TO WHICH IT IS FIRST CONSIGNED BY WILEY INDIA PVT. LTD. AND MAY NOT BE RE-EXPORTED.

FOR SALE ONLY IN :

INDIA, BANGLADESH, NEPAL, PAKISTAN, SRI LANKA & BHUTAN

WSE
WILEY

Wiley India Pvt. Ltd.
4435/7, Ansari Road, Daryaganj,
New Delhi-110 002.
Tel: 91-11-43630000
Fax: 91-11-23275895.
E-mail: csupport@wileyindia.com
Website: www.wileyindia.com

Stern/
Stern

Structured COBOL Programming

8th Edition

NOT FOR
RESTRICTED
RE-EXPORT

WSE
WILEY

WILEY-
INDIA
EDITION

Structured COBOL Programming

8th Edition

WILEY
STUDENT
EDITION

RESTRICTED!
FOR SALE ONLY IN
INDIA, BANGLADESH, NEPAL,
PAKISTAN, SRI LANKA
& BHUTAN

STERN / STERN

som en indlejret del af kapitalismen, viser sig altså at være en myte, hvor den skjulte vedligeholdelse af *back-back-ends* ikke blot har at gøre med produktionen af tilsyneladende gnidningsfrie flows, men også med at bevare indtrykket af i det hele taget at være udviklet. Denne proces involverer den konstruerede vestlige verdens re-koloniale udnyttelse af indiske softwareingeniører til at sikre, at de udviklede kan forblive udviklede og udviklingslandene kan forblive under udvikling.

EPILOG

Ved at interagere konkret og materielt med COBOL har jeg forsøgt at få greb om de paradokser vedrørende udvikling og agens, som ligger bag udnyttelsen af det ellers usynlige vedligeholdelsesarbejde, som udføres af indiske softwareingeniører.

Undersøgelsen i den første lektion om Afvikling viste, hvordan udviklingen af avancerede programmeringssprog tillod en forflytning af afviklingsinstruktioner i tid og rum, hvilket ændrede programmeringskunsten fra at vedrøre lokale, stedsspecifikke og yderst specialiserede handlinger til at producere generelt anvendelige, automatiske ordrer, som i teorien kan udføres overalt.

I COBOLs særlige tilfælde er programmet ikke kun universelt anvendeligt, men også en universel global afvikling af bankoverførsler, billetreservationer, lagerstyringssystemer, administrering af containers osv. Ved således at flytte varer og ressourcer, penge og mennesker rundt, kan COBOL forstås som en ideel grænseflade for skabelsen af værktøjer og applikationer, der understøtter den gnidningsløse, automatiske drift af en globaliseret verden.

Men som vi så i anden lektion om Krise, er der ikke tale om friktionsfrie systemer, selvom de fremtræder sådan fra brugerens perspektiv, hvor man ikke kan se den fortsatte opdatering, reparation og forbedring af processerne omkring de gamle koder, så de ikke bryder sammen eller går i opløsning.

Vedligeholdelsesarbejde er stadigt pågående og dynamisk. Det foregår hele tiden i baggrunden, for at udjævningen kan forblive jævn. Det finder sted på flere planer og kan manifestere sig både asymmetrisk og symmetrisk. De eksempler, som min analyse af COBOL har tilvejebragt, er asymmetriske og medfører en politisk økonomi, hvor en devaluering af vedligeholdelsesarbejdet står over for en idealisering af kreativt arbejde, idet rutinemæssige opgaver genkendes ved at være forflyttet til bestemte globale zoner, mens glorificeret udviklings- og innovationsarbejde sker andetsteds. Det udmønter sig i etableringen af et skævvredet forhold mellem den såkaldte udvikling og underudvikling. Her bliver den overudviklede verdens systemer vedligeholdt af fjerne og derfor usynlige arbejdere, som holder de teknologiske veje åbne, og skaber en forestilling ikke bare om regelmæssighed men også om fremtidsudsigter hos de udviklede. Således forstået at

processen både fungerer i produktionen af friktionsløse rum – den rumlige dimension – og friktionsløs tid – den fremadskuende tidslighed.

Implikationerne af disse indsigter er en bydende opfordring til at omdefinere og afvise påstande om paradigmeskifte og nye revolutioner, som igen og igen fremsættes, men systematisk overser og forsømmer de underliggende, materielle vilkår og den forflyttede menneskelige arbejdskraft, som er en indlejret del af løfterne om friktionsløse fremtider og problemfrie rum.[33] I stedet foreslår jeg at rokke ved de overordnede ontologiske opfattelser af teknologisk udvikling ved at gå til systemernes sammenfiltring med verden og anerkende disse som en fortløbende proces af, hvad jeg har kaldt for '*Crisis Computing*'.

En sådan tilgang kræver en praksis optaget af omvendinger, og som gennem en udforskning af materialets granuløse skala såvel som en udredning af de komplekse og sammenfiltrede magtstrukturer, der er en integreret del af systemets bagvedliggende materielle betingelser, vender og drejer vores informationsarkitekturer igen og igen. Designpraksisser, som udelukkende fokuserer på *front-end design*, opretholder den asymmetriske organisering, som er eksemplificeret i dette essay. Via begrebet *Crisis Computing* vil jeg i stedet advokere for en mere situeret tilgang, som er opmærksom på den sociopolitiske og rumlige ulighed blandt brugere, vedligeholdere og udviklere. De eksisterende ontologier er foruroligende i deres fortsatte asymmetriske og neokoloniale organisering; derfor kalder de på at blive afviklet.

33 Herunder løftet om fuld automatisering, f.eks. AI, som i nyere, spektakulære versioner betegnes som 'den fjerde industrielle revolution' eller 'den anden maskinalder'.

Billedfortegnelse:

Figur 1.1: The First "Computer Bug". NH 96566-KN, Ophavsret: Naval Surface Warfare Center, Dahlgren, VA., 1988.

Figur 1.2: Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020.

Figur 1.3: HK Aerials synsvinkel. Screenshot fra James Cameron's, *The Terminator,* 1984.

Figure 2.1: En krussedulle, 9. august 1993, af Jerome Garfunkel. Kilde: *Calligraphy and Computer Standards.*

Figur 2.2: Screenshot fra Stanley Kubrick's *Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb*, 1964.

Figur 2.3: Uddrag af "66 tests of materials for covering a war room table". Mixed media installation. Linda Hilfling Ritasdatter, 2016.

Figur 3.1: Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020.

Figur 3.2: Infosys GEC II, Mysore. Foto: Linda Hilfling Ritasdatter.

Figur 3.3: Uddrag af "Unwrapping COBOL", performance forelæsning. Linda Hilfling Ritasdatter, 2020.

Figur 3.4: Ren (99.9 %) cobalt chips, elektrolytisk raffineret. Foto: Heinrich Pniok.

Figur 3.5: Min COBOL-lærebog. Foto: Linda Hilfling Ritasdatter.

Kunsten som Forum på tryk er en skriftserie med udvalgte tekster om kunstteori og -praksis, som på forskellig vis gentænker forholdet mellem kunsten og det sociale.

Formålet med serien er at styrke udvekslingen mellem kunstens teori og praksis og invitere en bredere kreds af kunstens interessenter – producenter, formidlere og publikum – ind i igangværende samtaler om kunsten og dens fællesskaber i bredeste forstand.

Serien er udgivet i et samarbejde mellem Kunsten som Forum, Ny Carlsbergfondets forskningscenter på Københavns Universitet og Billedkunstskolernes Forlag.

I samme serie:

01 Mustafa Dikeç, *Æstetik og skabelsen af fælles verdener*, 2021

02 Bojana Kunst, *Feministisk omsorgspolitik og nærheden mellem kunst og liv*, 2021

03 Cecilia Sjöholm, *Virkelighedssans: Hannah Arendt om Kant og æstetik*, 2021

04 Nishant Shah, *Falske fremtider: Morgendagen gennem computernetværkets filtre*, 2022

05 Linda Hilfling Ritasdatter, *Crisis Computing*, 2022

06 Nora Sternfeld, *I det postrepræsentative museum*, 2022

BILLEDKUNST SKOLERNE
Det Kongelige Danske Kunstakademi

KUNSTEN SOM FORUM
ART AS FORUM

KØBENHAVNS UNIVERSITET

Billedkunstskolernes forlag
ISBN: 978-87-7945-127-8