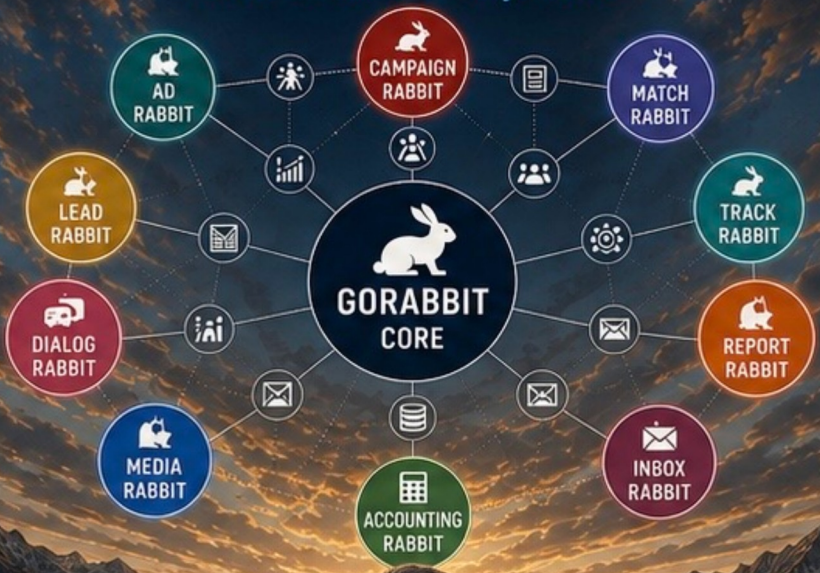
JEG KAN IKKE KODE

UTDRAG

Hvordan jeg bygget et helt forretningssystem med AI – uten å skrive en linje kode



En historie om ChatGPT, Claude, 1 407 PHP-filer, Python-motorer, noen kraftige krangler – og hvorfor 40 års erfaring fortsatt betyr noe.

MORTEN VON HAFENBRÄDL

Til deg som leser dette

Dette er et utdrag fra en bok om å bygge en komplett markedsføringsplattform uten å kunne programmere.

Ikke fordi jeg lærte meg det underveis. Jeg kan fortsatt ikke kode. Fem hundre linjer Python ser fremdeles ut som koblingsskjemaet til en varmepumpe. Men jeg fant en arbeidsform sammen med kunstig intelligens som gjorde det mulig — og som viste seg å handle om noe helt annet enn teknologi.

Boka er skrevet på samme måte som systemet den handler om: av et menneske i tett samarbeid med maskiner. Første del ble til sammen med ChatGPT, andre del sammen med Claude, og byttet mellom dem er beskrevet inne i boka selv — av verktøyet som overtok. En bok om å bygge med AI uten å kode, skrevet for hånd, hadde vært en selvmotsigelse.

De tre delene du får her er satt sammen og skrevet om for anledningen. De handler om der det begynte, om hvordan arbeidet faktisk artet seg, og om det jeg til slutt satt igjen med.

DEL 1

Jeg trodde jeg skulle kjøpe et system

Det begynte ikke med kunstig intelligens. Det begynte med irritasjon.

Den helt spesielle typen irritasjon som oppstår når du har sittet gjennom enda en presentasjon av et «komplett system» som visstnok løser alt — helt til du stiller spørsmålet om akkurat det du faktisk trenger. Da blir det stille i rommet. Ikke lenge, bare akkurat lenge nok til at du skjønner at løsningen likevel ikke var så komplett som brosjyren lovet.

Jeg hadde gjennom årene samlet en rekke ulike virksomheter under samme paraply: fotografering, formidling, utleie, og flere til. Felles for dem alle var at de trengte markedsføring, og at jeg brukte enormt mye tid på å skrive tekster jeg trodde var gode — som viste seg å ikke være så gode likevel, og som uansett tok altfor lang tid å lage.

Her kommer mønsteret alle som driver flere ting samtidig vil kjenne igjen: når jeg ikke hadde tid, brukte jeg ikke tiden på markedsføring. Da ble markedsføringen lidende, og da ble virksomhetene lidende, og da satt jeg der med dårlig samvittighet og en huskeliste som bare vokste.

Det var dette jeg ville løse. Ikke fordi jeg drømte om å bygge programvare, men fordi jeg ville ha tiden min tilbake.

Så jeg lette i markedet, slik fornuftige mennesker gjør. Alle hadde pene dashboards. Alle hadde grafer. Alle hadde en knapp som het noe med «integrasjon». Og alle hadde et lite

område, gjerne langt nede på prislisten eller dypt inne i dokumentasjonen, hvor virkeligheten lå begravet — den som sa at akkurat det jeg trengte krevde en konsulent, et tillegg, eller en versjon de skulle lansere neste år.

> *«Hvis noen sa 'dette er veldig fleksibelt', betydde det som regel: 'Dette kan ikke brukes uten konsulent.'»*

Det tok meg lang tid å forstå at jeg stilte feil spørsmål. Jeg lette etter ett system som kunne løse problemene mine. I dag vet jeg at det spørsmålet aldri kunne gi et godt svar. En virksomhet består ikke av ett problem som kan løses av ett program. Den består av hundrevis av små prosesser, beslutninger og arbeidsflyter som må fungere sammen.

Jeg trengte ikke et system. Jeg trengte et økosystem. Jeg visste det bare ikke ennå.

Den første forelskelsen, og hverdagen etterpå

De første kveldene med AI var magiske. Jeg beskrev noe, og det kom kode tilbake. Det virket. Etter tjue år med å høre at «det er nok mulig, men det blir dyrt», satt jeg plutselig og fikk ting gjort på en kveld.

Så kom hverdagen, slik den gjør i alle forhold.

Verktøyet var fortsatt fantastisk på sitt beste, men det kunne også svare med en selvtillit som burde vært regulert av Finanstilsynet. Det foreslo biblioteker som ikke fantes. Funksjoner som hadde byttet navn for tre versjoner siden. Løsninger som så ut som arkitektur helt til de møtte en ekte server og falt sammen som et korthus i trekk.

Jeg skal være ærlig om hvor langt frustrasjonen gikk, for det hører med. Jeg kunne fylt hundre sider bare med utdrag fra samtaler der jeg — unnskyld uttrykket — forbannet maskinen oppover og nedover.

Og her er poenget som gjør dette til mer enn en klagesang: hadde jeg ikke hatt bakgrunnen min, hadde jeg godkjent rotet.

For det var systemarkitekten i meg som reagerte. Svarene hang ikke sammen. De var ikke konsistente fra dag til dag. Navnekonvensjoner skled, strukturer ble glemt, og løsninger som ble foreslått på tirsdag motsa dem fra mandag. En som ikke hadde sett systemutvikling fra innsiden, ville antagelig tatt imot hver bit med takknemlighet — og bygget videre på sand.

Dermed måtte jeg ta en mye større rolle enn jeg hadde planlagt: ikke bare bestiller, men systemarkitekt med krav til hvordan ting skulle bygges, og prosjektleder som styrte hver eneste leveranse.

> *«AI er ikke sannheten. AI er en ekstremt rask praktikant med god språkføring.»*

Gjettearbeidet

Etter hvert sluttet jeg å se på AI-en som et verktøy og begynte å se på den som en ansatt. En merkelig ansatt, riktignok — en som kunne skrive kode, forklare arkitektur, lage tekster og foreslå løsninger døgnet rundt, men som aldri burde få nøklene til banken.

Men det var én uvane hos denne ansatte jeg aldri har akseptert, og som er kanskje det mest praktiske rådet jeg har å gi. Jeg kaller det gjettearbeid.

AI-verktøyene har en tendens til å gjette i stedet for å analysere, og til å kjøre prøving og feiling i stedet for å undersøke hvorfor noe skjer. I stedet for å slå opp hva feltet i databasen faktisk heter, gjetter de på et navn som høres sannsynlig ut. I stedet for å analysere feilmeldingen, prøver de en variant til. Hver gjetterunde koster tid — og i en verden der man betaler for AI-bruk, koster den bokstavelig talt penger også.

Det fine er at gjettingen er lett å kjenne igjen når du først vet hva du ser etter. Det er noe med måten svaret er formulert på: en viss ullenhet, et «la oss prøve», en løsning som kommer litt for fort etter den forrige som feilet.

Da har jeg én fast respons, hver gang, uten unntak: jeg stanser umiddelbart og ber om analyse. Ikke prøv noe nytt. Analyser hvorfor det feilet, og kom tilbake med et forslag basert på analysen.

Og vet du hva? Da gjør de det. Og analysen er nesten alltid riktig på første forsøk. Det som irriterer meg, er at det er jeg som må be om det.

> *«Mennesket er CEO og tester. AI er utvikler, rådgiver og sparringspartner. Ansvaret blir aldri outsourcet.»*

DEL 2

Når systemet svarer «OK»

Betaling skulle være det trygge hjørnet av prosjektet. Stripe er voksent, gjennomtestet og kjedelig på den gode måten. Etter alt jeg hadde vært gjennom med plattformregler og juridiske gjennomganger, gledet jeg meg nesten til å jobbe med noe der reglene var klare og dokumentasjonen god.

Så jeg koblet opp en webhook — meldingen betalingsleverandøren sender til systemet ditt når noen har betalt, slik at kunden kan aktiveres automatisk — og lente meg tilbake for å se på at pengene og aktiveringene trillet inn.

Dashbordet lyste grønt. «200 OK», sto det, om og om igjen, ved hver eneste leveranse. Alt så perfekt ut.

Problemet var bare at ingenting skjedde i databasen min. Ingen kunder ble aktivert, ingen felter ble oppdatert, ingenting. Grønt lys hos leverandøren, tom database hos meg — og jeg brukte altfor lang tid på å forstå hvordan begge deler kunne være sanne samtidig.

Jeg lette i koden. Jeg lette i databasetilgangene. Jeg lette alle stedene en systemarkitekt leter, helt til tiøringen falt et sted ingen av oss hadde sett på: i selve plasseringen av filen.

Webhook-filen min lå inne i en WordPress-mappe, og WordPress har en egen logikk som sender alt den ikke kjenner igjen, videre til forsiden. Så når betalingsleverandøren banket på adressen min, svarte WordPress høflig med å servere forsiden til nettstedet mitt

— som selvfølgelig lastet helt fint, og dermed ga et blidt «200 OK».

Koden min hadde aldri kjørt. Ikke én gang. I ukevis hadde leverandøren fått beskjed om at alt var i orden, av en forside som ikke ante hva en webhook var.

> *«'200 OK' er ikke det samme som 'det gikk bra'. Noen ganger betyr det 'forsiden lastet fint, og koden din sov.'»*

Løsningen var nesten fornærmende enkel: flytt filen ut av WordPress sitt rike. Men lærdommen var stor, og den har fulgt meg siden som en av de viktigste kontrollreglene i hele driftsopplegget: en vellykket statuskode betyr bare at noe svarte — ikke at din kode kjørte, og slett ikke at den gjorde det den skulle.

Det eneste som teller, er sideeffekten. Ble raden faktisk skrevet til databasen? Ble e-posten sendt? Ble kunden aktivert? Status kan lyve. Resultatet kan ikke.

Maskinen som myste på en skjerm

En av de tidlige løsningene mine leste Facebook ved å ta skjermbilder og kjøre tekstgjenkjenning på dem — i praksis å lære maskinen å myse på en skjerm og gjette hva som sto der. Det høres absurd ut når jeg skriver det nå, men det fantes en logikk i det den gangen, og løsningen virket akkurat nok til å lure meg og akkurat dårlig nok til å gjøre meg gal.

Symptomet var nemlig så merkelig spesifikt: uansett hvor aktiv gruppen var, kom det nøyaktig to treff ut av hver

skanning. To. Hver gang. En aktiv gruppe med femti innlegg: to treff. En død gruppe med tre innlegg: to treff.

Jeg holdt lenge fast ved metoden, og grunnen er like menneskelig som den er farlig: jeg hadde bygget den selv, og den virket nesten. Det er en kombinasjon som har holdt liv i flere dårlige systemer enn noen annen kraft i IT-historien — eierskap pluss nesten.

Til slutt forkastet jeg hele bilde-lesingen og lot maskinen i stedet lese teksten som faktisk lå strukturert i kilden, der den hele tiden hadde ligget. Plutselig kom det åtte, ti, tolv ekte treff per gruppe, og kvaliteten var en helt annen — for maskinen sluttet å gjette på et fotografi og begynte å lese innholdet.

> *«Vi hadde lært maskinen å skvise øynene sammen og gjette. Det var bedre å la den lese.»*

Lærdommen er en av dem jeg oftest deler når jeg snakker med andre som bygger med AI: når strukturen finnes — i koden bak siden, i et grensesnitt, i en datastrøm — så les den direkte. Visuell tolkning er siste utvei, ikke første. Du skal ikke fotografere svaret når du kan be om det.

Da jeg nesten bygde en konsernsjef

Den neste ideen var uimotståelig på den måten store feil ofte er uimotståelige: Hva om hele systemet hadde én intelligens som satt over modulene og bestemte hvem som skulle gjøre hva? Én kunne være salgsdirektør, en annen markedsdirektør, en tredje økonomidirektør — og over dem en slags digital konsernsjef som prioriterte, fordelte ressurser og koordinerte alt.

Vi tegnet til og med organisasjonskartet. Og det så flott ut. Nettopp derfor måtte jeg bli mistenksom.

For modulene var bygget for å kunne leve selvstendig, og hele poenget med arkitekturen var at én modul ikke skulle trekke resten ned. En sentral «konsernsjef» ville gjort det motsatte. Den ville blitt en ny feilflate, en ny avhengighet og en grunn til å åpne kode som allerede var ferdig. Enda verre: den ville gitt meg et nytt stort prosjekt akkurat da jeg trengte å avslutte det jeg allerede hadde.

Så vi gjorde det vanskeligste man kan gjøre med en god idé: vi tok livet av den store versjonen og beholdt prinsippene.

Det er lett å bruke AI til å få flere ideer. Det er mye vanskeligere å bruke vanlig intelligens til å si nei til dem. AI er den dårligste vennen du kan ha i det øyeblikket, fordi den aldri blir sliten av ambisjon. Den sier gjerne ja til den større ideen og kan på sekunder beskrive hvor fantastisk den blir.

Min jobb var plutselig ikke å finne flere muligheter. Den var å holde igjen.

> *«Arkitektur handler like mye om hva du nekter å koble sammen som om hva du kobler sammen.»*

DEL 3

Erfaring kan ikke lastes ned

Det er vanskelig å lese en avis om kunstig intelligens uten å få inntrykk av at alle yrker står på en brygge og venter på å bli erstattet. Programmerere først, så konsulenter, markedsførere, økonomer, jurister og resten av oss i en høflig kø.

Etter å ha brukt noen år på å bygge et helt system med AI som nærmeste medarbeider, kjenner jeg meg dårlig igjen i den historien.

La meg bruke mitt eget prosjekt som bevis, ikke som fasit for hele verden.

Jeg hadde aldri kunnet bygge dette uten AI. Det er helt sant. Jeg kunne ikke skrevet koden, integrasjonene, databaselogikken eller noen av de moderne komponentene som måtte til. Maskinene har produsert arbeid jeg ellers måtte kjøpt fra mange mennesker, og de har gjort det i et tempo jeg fortsatt synes er absurd.

Men den motsatte setningen er like sann: AI hadde aldri kunnet bygge dette uten meg.

Ikke fordi jeg er flinkere enn en språkmodell til å huske syntaks, men fordi systemet er bygget av erfaring som ligger et helt annet sted. Tretti–førti år med systemarkitektur, integrasjoner, drift, kundereiser, forretningsmodeller, ledelse — og feil som har kostet penger.

AI kunne foreslå en funksjon. Jeg måtte vite om funksjonen burde finnes.

AI kunne bygge et dashbord. Jeg måtte vite hvilket spørsmål dashbordet skulle besvare før morgenkaffen.

AI kunne lage en annonsemotor. Jeg måtte stoppe og spørre om kunden i det hele tatt burde annonsere.

AI kunne produsere 240 sider dokumentasjon. Jeg måtte oppdage at en senere versjon hadde mistet substans selv om den så penere ut.

Der knappheten flytter seg

Dette er ikke et forsvar for at alt skal fortsette som før. Tvert imot. Jobber kommer til å endres dramatisk, akkurat som min gjorde. Oppgaver flyttes mellom menneske og teknologi, tempoet øker, og én person kan levere det som tidligere krevde et helt team.

Men nettopp derfor melder det seg et spørsmål jeg ikke klarte å holde innenfor mitt eget prosjekt.

Når produksjonskapasiteten blir billig, flytter knappheten seg. Fra å kunne lage ting, til å vite hva som bør lages — i hvilken rekkefølge, innenfor hvilke grenser, og hvordan du kan se at resultatet faktisk skaper verdi.

Norge kommer neppe til å mangle folk som kan få AI til å skrive kode. Det vi risikerer å mangle, er mennesker som forstår koblingen mellom teknologi, forretning og mennesker godt nok til å ta de riktige beslutningene når ingen fasit finnes.

For hvor kommer den forståelsen fra?

Den lastes ikke ned. Ingen lærer betydningen av datakvalitet før de har stått i konsekvensene av dårlig

datakvalitet. Ingen forstår verdien av en sammenhengende kundereise før de har sett kunder falle ut av den. Slik innsikt bygges nedenfra — gjennom de små, rutinepregede oppgavene en fersk medarbeider fikk bryne seg på mens noen mer erfarne holdt et øye med resultatet.

Men det er nettopp disse oppgavene AI nå overtar.

Vi står altså i fare for å fjerne de nederste trinnene på stigen, samtidig som vi forventer at folk fortsatt skal nå toppen.

> *«En bransje som spiser opp sine egne læringsarenaer, vil før eller siden oppdage at den har sluttet å produsere erfarne folk.»*

Tigeren og kaninen

De første eksperimentene med AI var små, nesten lekne — en liten funksjon her, en test der — og så begynte det plutselig å ligne et system. Oppgaver som tidligere ville tatt uker, kanskje måneder, lot seg løse på noen timer.

Det var som å slippe løs en tiger: rå kraft, enorm fart, og en følelse av nesten ubegrensede muligheter. Og det er lett å la seg rive med av en tiger, helt til du oppdager at den ikke er bygget for å lytte. Den er bygget for å ta ting.

Det var her navnet på systemet mitt begynte å bety mer enn en logo. Kaniner er ikke tigere. De er små, raske og fullstendig avhengige av å høre etter, og de overlever nettopp fordi de ikke prøver å være størst. De tar små steg, men mange av dem, og de kan snu.

Jo lenger jeg holdt på, desto tydeligere ble det at det var kaninen jeg trengte å bygge — ikke fordi den imponerer mest ved første øyekast, men fordi den tåler tid.

> *«Rå kraft imponerer i noen uker. Evnen til å snu er det som avgjør om systemet fortsatt finnes om to år.»*

Så hva er det egentlig som har endret seg, hvis jeg fortsatt ikke kan kode? Dette: jeg vet hvilke spørsmål jeg skal stille, jeg tester alt, og jeg beslutter. AI-en foreslår, utvikler og dokumenterer — men den signerer ikke, og den får aldri siste ord når noe skal i produksjon.

Grensen for hva jeg kan, har ikke flyttet seg nevneverdig. Grensen for hva jeg kan ta ansvar for, har flyttet seg enormt. Og det er to forskjellige grenser, selv om de forveksles hver eneste dag i debatten om AI.

RESTEN AV BOKA

Dette utdraget viser problemet, håndverket og innsikten. Det som ligger mellom — hvordan det faktisk ble gjort, dagen drømmen møtte GDPR, den store oppryddingen, den første kunden som ikke var meg, og hva jeg ville gjort annerledes — er i boka.

Den er skrevet for folk som driver noe selv. Ikke for utviklere.

Underveis: 35 kapitler med konkrete lærdommer, feil som kostet, og en arbeidsform som lot seg gjenta.

FOREDRAGET

Jeg holder også foredrag om det samme. Da blir det anledning til å stille spørsmålene som gjelder din egen situasjon, og ikke bare min.

Foredraget passer for ledergrupper, bransjeforeninger, nettverk og samlinger der spørsmålet «hva betyr AI for oss?» er mer aktuelt enn «hvordan skriver vi prompts?». Det handler om arbeidsdeling, ansvar og dømmekraft — ikke om verktøy som er utdatert til neste år.

Morten von Hafenbrädl

NiceTalks

«Keep it simple. Even the complex things.»

JEG KAN IKKE KODE.

LIKEVEL BYGGET JEG GORABBIT.

Et helt forretningssystem med titalls moduler, over tusen PHP-filer, Python-motorer, databaser, automatisering, markedsføring, analyse og AI.

Jeg gjorde det sammen med ChatGPT og Claude.

Det høres ut som historien om hvordan AI har gjort programmerere, arkitekter og andre kunnskapsarbeidere overflødige.

Det er det ikke.

Underveis gjorde AI fantastiske ting. Den skrev kode jeg aldri kunne skrevet selv. Den analyserte systemer på minutter. Den produserte dokumentasjon det tidligere ville tatt uker å lage.

Den tok også feil. Den glemte. Den overkompliserte. Den forenklet ting som ikke skulle forenkles. Den kunne bruke timer på å forklare hva den skulle gjøre i stedet for å gjøre det. Og enkelte dager kranglet Claude og jeg som et gammelt ektepar.

GoRabbit kunne aldri blitt bygget uten AI.

Men etter denne reisen er konklusjonen min like klar:

AI kunne heller aldri bygget GoRabbit uten meg.

Denne boka handler om hvorfor.

1 407 PHP-FILER
101 PYTHON-SCRIPTS
53 GR-TABELLER

TITALLS MODULER
I EN SAMMENHENGENDE
PLATTFORM

DATA SOM JOBBER
AUTOMATISERT OG
MÅLBART

BYGGET MED AI
– DESIGNET MED
ERFARING

GORABBIT
ECOSYSTEM

> AI ER EN EKSTRAORDINÆR MEDARBEIDER. MEN DET ER ARKITEKTUREN SOM AVGJØR OM RESULTATET BLIR ET HUS ELLER ET KAOS.

ISBN 978-82-303-7845-8

MER OM GORABBIT
gorabbit.no