ALEXANDRA INSTITUTTET

AI-assisteret udvikling

Hvad har egentlig ændret sig — og hvad betyder det for dit team?

Hvem er Alexandra Instituttet?

- Etableret I 1999
- GTS-institut specialiseret i IT- og digitalisering
- Ejet af Aarhus Universitets forskningsfond
- Kontorer I Aarhus og København
- Faglige labs: AI, Security, Insights og Digital Experience & Solutions

Hvem vi er, og hvorfor vi står her

Mads Darø Kristensen

Principal Application Architect, PhD

Kaspar Rosengreen Nielsen

Principal Software Architect

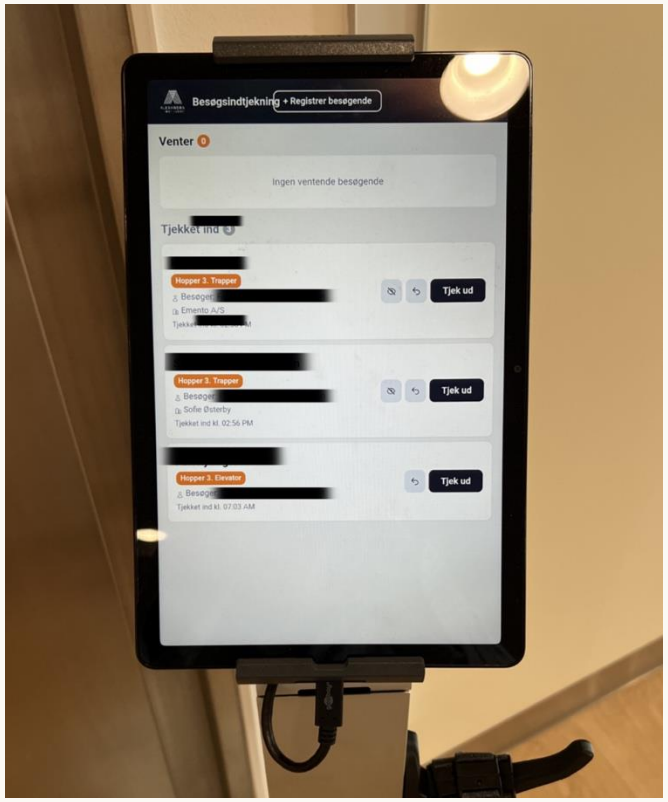
Hvad det her oplæg er

Et praktisk perspektiv fra et hold, der er i fuld gang med at gøre det her i virkeligheden – ikke kun for os selv, men også for vores kunder.

- **Overfladisk** på det basale — I har ikke brug for endnu en LLM 101.
- **Dybere** på det, der faktisk er svært: at få det til at fungere på tværs af et hold.
- **Ærligt** om det, vi endnu ikke har fundet ud af.

Ikke et produktpitch. Ikke et hype-show.

En kort historie



Vi havde brug for et besøgsregistreringssystem.

Noget vi selv skulle bruge hver dag på kontoret.

Vi byggede det på

70 timer

Stabilt. Vedligeholdbart. Nu i drift.

Hvorfor jeg fortæller den historie

For et år siden ville det her projekt have taget os et par måneder.

Noget har ændret sig — for nylig — og de fleste baserer stadig deres holdning på indtryk fra før den ændring.

Det er det, oplægget handler om.

ALEXANDRA INSTITUTTET

AI-assisteret udvikling er ikke længere valgfrit

Hvad "ikke længere valgfrit" betyder

Det betyder ***ikke***, at AI erstatter udviklere.

Det betyder:

- Hold, der adopterer det her, kommer til at levere markant hurtigere.
- De udviklere, I gerne vil ansætte, forventer at I bruger værktøjerne.
- Jeres bedste udviklere bruger dem allerede — med eller uden jeres velsignelse.
- Forskellen mellem dem, der har taget det til sig, og dem der ikke har, vokser hele tiden.

Den indvending vi oftest hører er

> *“Den genererer alt for meget kode, og kvaliteten er dårlig.”*

Hvis I har prøvet værktøjerne og er gået derfra med det indtryk, så havde I ret.

I havde ret — indtil for nylig

De tidligere værktøjer havde reelle problemer:

- Modeller, der selvsikkert producerede kode, der ikke virkede.
- Agenter, der gik i loops og brændte tokens af uden resultat.
- Output, der så plausibelt ud, men ikke virkede.
- Mere oprydning end værktøjet sparede jer for.

De fleste, der har forsøgt sig med AI udvikling, dannede deres indtryk i den periode.

Hvad har ændret sig

Tre ting har rykket sig — nogenlunde samtidig:

01

Modellerne

er blevet markant bedre til at ræsonnere om kode.

02

Værktøjerne

er fulgt med — agenter kan nu køre tests, læse fejlbeskeder og navigere kodebaser.

03

Workflows

er modnet — multi-agent setups er holdt op med at gå i loops og er begyndt at levere.

Det her er ikke en jævn kurve. Det er et komplet faseskift.

Skiftet

DEN MENTALE MODEL DE FLESTE STADIG HAR

"AI hjælper dig med at skrive din kode hurtigere."

DEN NUVÆRENDE VIRKELIGHED

"AI planlægger, skriver, tester og itererer på kode — med et menneske i loopet."

Det er en helt anden måde at tænke på.

Tilbage til de 70 timer

- **Hvad appen rent faktisk gør:** registrering af besøgende på Alexandra Instituttets Aarhus kontor.
- **Hvem byggede den:** Et lille hold på tre / designer, udvikler, arkitekt
- **Hvad de 70 timer dækkede:** planlægning, design, udvikling, hardware-indkøb, test og idriftsættelse
- **Hvad det ville have taget før:** Måske ~400 timer?
- **Hvad gik i stykker efter launch:** Uidentificerede krav dukkede op – det hele var gået så hurtigt

Pointen er ikke hastigheden. Et lille team byggede noget rigtigt, som de selv ejer og vedligeholder, på en brøkdel af tiden.

Vores nye workflow

I grove træk:

01	**Holdet & agent**	— finder på ny ønsket feature sammen med interessenter
02	**Udvikler & agent**	— planlægger implementeringen og foreslår en tilgang i tæt samarbejde
03	**Agenter**	— skriver kode, kører tests, itererer på fejl
04	**Udvikler**	— reviewer, presser tilbage, herefter kører skridt 3 og 4 igen og igen
05	**Udvikler**	— laver PR, reviewer selv inden overlevering
06	**Reviewer**	— reviewer, kan tage processen helt tilbage til skridt 2

Der spares tid i skridt 2 og 3 – der bruges mere tid i skridt 4 og 5.

Hvad der stadig ikke virker

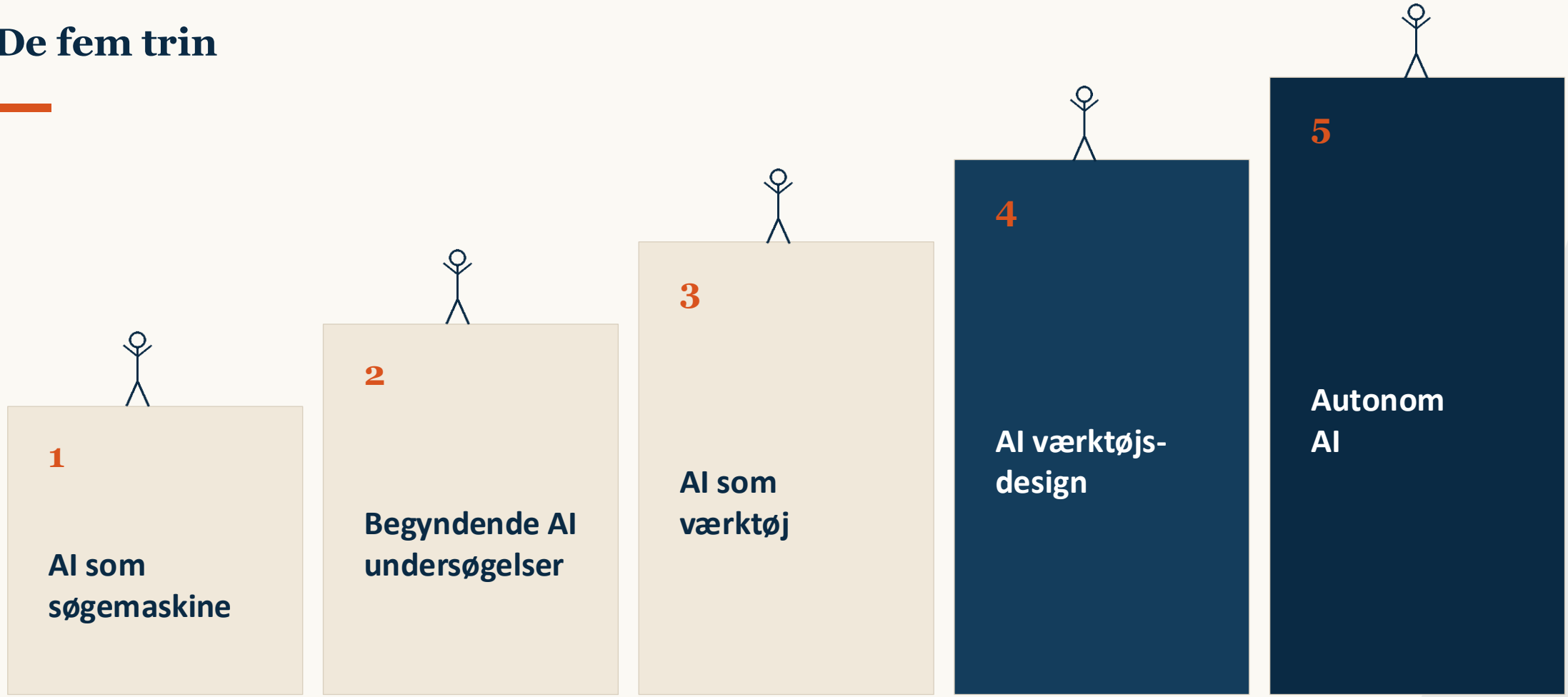
Den ærlige liste:

- Opgaver med dyb, udokumenteret kontekst — agenten kan ikke læse jeres holds tanker.
- Alt hvor det svære er selve spec'en.
- Alt hvor "ser plausibelt ud" er farligt — sikkerhedskritisk kode, nye algoritmer.
- Store refactorings i ukendte kodebaser.

Værktøjet forstærker klar tænkning. Det erstatter den ikke.

Det organisatoriske

De fem trin


1
AI som søgemaskine
2
Begyndende AI undersøgelser
3
AI som værktøj
4
AI værktøjs-design
5
Autonom AI

Tre lejre

I de hold vi har talt med — inklusive vores eget — fordeler udviklerne sig i tre grupper.

Entusiasterne

Bruger det allerede. Finder nye workflows. Evangeliserer.

De forsigtigt nysgerrige

Åbne, men usikre på hvor de skal starte, eller om det er besværet værd.

Skeptikerne

Bekymrede for, at det tager håndværket og glæden ud af jobbet.

Det her er ikke et problem, der skal løses. Det er en fordeling, der skal håndteres.

Hvad vi gjorde

Vi købte licenser til de dyre værktøjer til alle udviklere. Ingen undtagelser. Læring er forventet. Brug er op til den enkelte.

Hvorfor den form:

- Påbud giver ikke reel adoption.
- Når adgang ikke længere er en undskyldning, tvinger man en ærlig samtale.
- Frivillig brug respekterer, at gode udviklere kender deres arbejde.
- Sæt udviklerne fri til at gå for langt, eksperimentere, vurdere hvad der virker og ikke virker lige nu – og lad dem endelig bruge det i fritiden.

Hvad politikken ikke løser

Lad os være ærlige: det her er vi stadig ved at finde ud.

- "Læring" gør meget arbejde i den sætning — overfladisk eksponering er ikke reel engagement.
- På et tidspunkt løber frivillig brug ind i hårde afvejninger: opgavefordeling, reviews, forventninger.
- Produktivitetsforskellen mellem lejrene vokser over tid.
- Jeg ved endnu ikke, hvordan den svære samtale skal se ud.

Hvis I ruller det her ud, rammer I den samme væg. Planlæg efter det.

> ***"Læring er forventet. Brug er op til den enkelte."***

Flaskehalsen er flyttet

Når agenter kan producere god kode i stor skala, er det ikke længere det at skrive kode, der er begrænsningen.

De nye problematikker er:

- **Ejerskab** — hvem forstår faktisk koden om seks måneder?
- **Review** — hvad er et meningsfuldt niveau af code review for et AI-genereret PR?
- **Viden** — hvordan bygger holdet fælles mentale modeller af systemet?
- **Juniorer** — hvordan udvikler de sig, hvis de læner sig op ad værktøjet fra dag ét?
- **Konsistens** — fem PR'er, fem forskellige stilarter. Hvad nu?

VORES BUD (1/2)

Vores bud

Ejerskab	I sidste ende er koden lavet af den, der har styret agenten og løbende rettet ind og lavet reviews.
Review	Man skal for alt i verden undgå, at den der ejer koden smider ansvaret for den. Så review af en anden kommer efter grundigt review fra ejeren.
Viden	Håbet er, at grundige reviews på begge sider (ejer og reviewer) gør at dette ikke rykker sig alt for meget. Men der laves godt nok hurtigt ny kode…

VORES BUD (2/2)

Vores bud

Juniorer

Det er vores største bekymring lige nu. De skal have lov til at lære selv, ellers kan de ikke give ordentlige reviews, guide agenten osv.

Konsistens

"Skills", tests og rigid CI/CD er vejen frem her. Det skal I have eksperimenteret med og have opbygget en proces omkring.

Hvad vi stadig prøver at finde ud af

De reelle åbne spørgsmål:

- Hvordan måler man, om AI-assisteret udvikling rent faktisk virker — ud over "bruger folk det?"
- Hvordan bevarer man dyb forståelse af sine systemer, når mindre kode skrives i hånden?
- Hvordan udvikler man den næste generation af seniorudviklere?
- Hvordan ser organisationen ud om to år, hvis kurven fortsætter?
- Hvordan estimerer vi opgaver?

Hvis I har svar, vil vi gerne høre dem.

Hvad I kan gøre på mandag

Fire konkrete ting:

01 Prøv den nuværende generation selv. Ikke den version I prøvede sidste år. Den der bliver udgivet og udviklet på lige nu.

02 Spørg jeres hold, hvad de faktisk bruger i dag. Sanktioneret eller ej. I lærer noget.

03 Vælg ét workflow at teste. Noget rigtigt, med en klar definition af "færdig".

04 Begynd at diskutere det med holdet nu. Før produktivitetsforskellen tvinger samtalen frem.

ALEXANDRA INSTITUTTET

Begrænsningen for, hvad jeres team kan bygge, har flyttet sig.

ALEXANDRA INSTITUTTET

Spørgsmål?

Kaspar Rosengreen Nielsen
kaspar.rosengreen@alexandra.dk

Mads Darø Kristensen
mads.kristensen@alexandra.dk